Año: 2024. Volumen: 1
Numero:10. ISSN: 2448-623X

Determinación de los Flujos de Potencia en una Red Eléctrica Utilizando el Método de Desacoplado Rápido en Python

Muñoz-Luna, E.J.[1] ✉; Luna-Aguilera, C.[1]; Hernández-Flores, C. [1]; Arjona-López, M. A. [1]

Datos de Adscripción:
[1] Tecnológico Nacional de México/ Instituto Tecnológico de La Laguna, División de Estudios de Posgrado e Investigación, Blvd. Revolución y Av. Instituto Tecnológico de La Laguna, C.P. 27000, Torreón, Coahuila, México.
✉ johnamn97@gmail.com

Resumen - **El método de desacoplado rápido se emplea en los sistemas eléctricos de potencia como una técnica numérica para aumentar la velocidad y eficiencia en el cálculo de flujos de potencia en la red. Este artículo presenta el análisis y la implementación del método de desacoplado rápido mediante una herramienta computacional desarrollada en Python. Se describe el modelo matemático utilizado para simular, calcular y evaluar el comportamiento de sistemas de buses establecidos por la IEEE. Se analiza la convergencia del método considerando los efectos de acoplamiento entre las variables del sistema, como la tensión y la potencia. Además, se detalla la metodología de desarrollo del código utilizando módulos y librerías como NumPy, SciPy y Pandas. Los resultados obtenidos se comparan con el software OpenDSS, mostrando la precisión y eficiencia del algoritmo implementado en Python. Finalmente, se destacan las ventajas del uso de Python en términos de flexibilidad, adaptabilidad y eficiencia en la simulación y análisis de sistemas eléctricos de potencia.**

Palabras Clave: **Algoritmo de flujos de potencia, Método de Desacoplado Rápido, Python, Simulador de redes eléctricas, Sistemas Eléctricos de Potencia.**

Abstract **- The fast decoupled method is used in power systems as a numerical technique to increase the speed and efficiency of power flow calculations in the network. This article presents the analysis and implementation of the fast decoupled method using a computational tool developed in Python. The mathematical model used to simulate, calculate, and evaluate the behavior of bus systems established by the IEEE is described. The convergence of the method is analyzed considering the coupling effects between system variables, such as voltage and power. Additionally, the methodology for developing the code using modules and libraries such as NumPy, SciPy, and Pandas is detailed. The results obtained are compared with the OpenDSS software, demonstrating the accuracy and efficiency of the method implemented in Python. Finally, the advantages of using Python in terms of flexibility, adaptability, and efficiency in the simulation and analysis of power systems are highlighted.**

Keywords: Algorithm for Power Flows, Decoupled Load Flow Method, Electric Network Simulator, Electric Power Systems, Python.

I. INTRODUCCIÓN

El análisis de los flujos de potencia es uno de los análisis más importantes en el estudio de sistemas de potencia, ya que ofrece las condiciones iniciales necesarias para la evaluación de la estabilidad, el análisis de fallas, la calidad de energía y el análisis de contingencias (Karimi et al., 2019). El objetivo del análisis es calcular magnitudes y ángulos de voltajes para una carga, generación y condiciones de red dadas (Volkov, 2020). Para realizar lo anterior se han implementado métodos diferentes, dentro de los cuales, algunos de los más implementados son: Newton Raphson, Gauss Seidel y Desacoplado Rápido. Este último es un método mejorado basado en la simplificación del método de Newton-Raphson, implementado por Stott y Alsac en 1974. Al igual que el método de Newton-Raphson, ofrece simplificaciones en los cálculos, converge de forma rápida y sus resultados son confiables (Afolabi et al., 2015). Sin embargo, el método de Desacoplado Rápido tiene un problema al momento de converger, el cual se presenta en redes con elementos R/X y G/B altos. (Yanez & Rolando, 2019)

Por otro lado, con el paso del tiempo las herramientas computacionales han pasado a ser un aspecto importante para solucionar todo tipo de problemas, dentro de los cuales se encuentran los problemas de flujos de potencia. En la actualidad, “Python”, es uno de los lenguajes más empleados y ha ganado popularidad. Este lenguaje se ha vuelto ideal para solucionar problemas complejos, gracias a sus bibliotecas, sintaxis intuitiva y por ser una herramienta de uso libre (Jaclolov, 2023). Actualmente es posible encontrar herramientas computacionales para sistemas de potencia creadas con este lenguaje. Un ejemplo de lo anterior es PyPSA, un software utilizado para análisis de flujos de carga en sistemas eléctricos. PyPSA destaca por ser una herramienta líder para simulación y optimización de sistemas de energía (Sharma & Dhillon, 2021). PyPSA ofrece varias ventajas, como su flexibilidad para adaptar modelos, un amplio número de herramientas de análisis, respuesta dinámica y simplicidad (Phongtrakul et al., 2018). LTB ANDES es una herramienta que fue desarrollada intentado cubrir el análisis tradicional de sistemas de potencia y la necesidad de estudios de ciberseguridad (Cui & Li, 2018), da soporte al cálculo de flujos de potencia, simulación de estabilidad transitoria y análisis de estabilidad de pequeña señal para sistemas de transmisión (Cui et al., 2021). PandaPower es una herramienta diseñada para automatizar el análisis y optimizar sistemas de potencia balanceados (Thurner et al., 2018), es sencillo de entender y manipular, lo que lo hace adecuado para investigación y educación, además posee funciones para resolver flujos de potencia, estimación de estados y cálculos de corto circuito, entre otros (Huang et al., 2020).

En este artículo se describe una herramienta computacional desarrollada en Python para la solución de flujos de potencia mediante el método de desacoplado rápido. Los módulos desarrollados utilizan librerías ya existentes como NumPy, Scipy y Pandas. La herramienta desarrollada utiliza el paradigma de programación orientada a objetos, por lo cual, además de las librerías anteriormente mencionadas, se realizaron módulos para tareas específicas. La creación de módulos ayudará a cumplir con el correcto funcionamiento del algoritmo. La contribución que se busca por medio de este trabajo es la implementación de una herramienta computacional programada en Python, la cual utiliza el método de Desacoplado Rápido para solucionar flujos de potencia, mejorando la accesibilidad y eficiencia de estas metodologías en el ámbito de la ingeniería eléctrica. Al estar desarrollado mediante la programación orientada a objetos, este trabajo facilita la expansión y adaptación futura del código, permitiendo su aplicación a una variedad más amplia de problemas complejos en sistemas de potencia. Se busca no solo mejorar la optimización de recursos computacionales, sino también fomentar la reproducibilidad y colaboración en futuras investigaciones, tanto académicas como aplicadas.

La forma en que se realizó este artículo, se presenta a continuación: en la sección II, se detallan los módulos que componen el código del Desacoplado Rápido, junto con la estructura aplicada y las ecuaciones empleadas en cada uno de ellos, la extracción de datos de los sistemas de la IEEE de 14, 30 y 57 Buses, la formación de la matriz Y-Bus y el método de Desacoplado Rápido. En sección III, se muestran los resultados de convergencia, tiempo de cómputo, voltajes y ángulos realizados por el algoritmo. Así mismo, para validar los resultados, se hace una comparativa con el software OPENDSS. Para finalizar, en la sección IV se encuentran las conclusiones a las que se llegaron en este artículo.

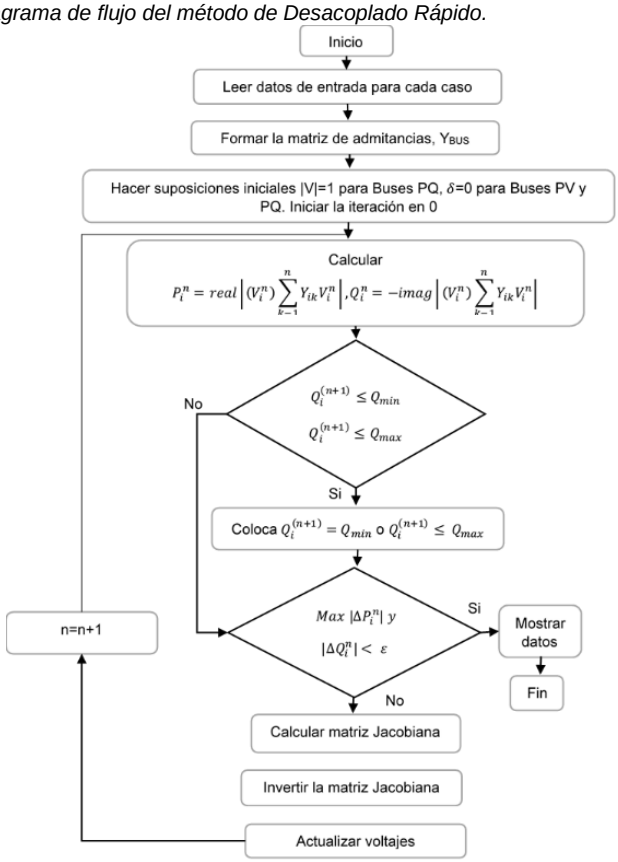
II. PARTE TÉCNICA DEL ARTÍCULO

La herramienta computacional que utiliza el método de Desacoplado Rápido se programó en Python. La Figura 1 muestra el proceso seguido para su implementación. Este método se seleccionó debido a su capacidad para ofrecer una solución rápida y eficiente en el análisis de flujos de potencia en sistemas eléctricos complejos. Para optimizar la implementación y facilitar futuras modificaciones, el código se dividió en 6 módulos. Esta modularización permite un manejo más limpio y organizado del código, facilitando su mantenimiento y mejora continua.

Cada módulo fue diseñado con una tarea específica, asegurando así una separación clara de las funciones y promoviendo la reutilización de código. Los módulos desarrollados incluyen funciones para la lectura y almacenamiento de datos, la definición de los elementos que conforman los buses y las líneas, el cálculo de la matriz de admitancias, y la implementación del algoritmo de Desacoplado Rápido.

La estructura modular no solo mejora la legibilidad del código, sino que también permite una fácil integración de nuevas características y la realización de ajustes específicos sin afectar el funcionamiento general del método.

Figura 1
Diagrama de flujo del método de Desacoplado Rápido.



Los módulos desarrollados son los siguientes:

- excelpy.py
- Line.py
- Bus.py
- Y_matrix.py
- FDPF.py
- main.py

Los cuáles serán explicados a detalle a lo largo de esta sección.

2.1 Formulación del algoritmo de Desacoplado Rápido

Los algoritmos para solución de flujos de potencia se han desarrollado para proporcionar soluciones en tiempos del orden de segundos o menos. Estos algoritmos están basados en la simplificación de la matriz Jacobiana, al omitir J_2 y J_3, la ecuación (1) se reduce a (Saadat, 2010):

$$\begin{bmatrix}\Delta P\\ \Delta Q\end{bmatrix} = \begin{bmatrix}J_1 & 0\\ 0 & J_4\end{bmatrix}\begin{bmatrix}\Delta\delta\\ \Delta|V|\end{bmatrix} \quad (1)$$

donde *P, Q, J, δ,* y *V* representan potencia activa, potencia reactiva, Jacobiano, ángulo y voltaje, respectivamente. Las ecuaciones (2) y (3) muestran cómo la ecuación matricial es separada en dos ecuaciones desacopladas, lo cual se traduce en un menor tiempo de solución.

$$\Delta P = J_1\Delta\delta = \left[\frac{\partial P}{\partial\delta}\right]\Delta\delta \quad (2)$$

$$\Delta Q = J_4 \Delta|V| = \left[\frac{\partial Q}{\partial |V|}\right] \Delta|V| \quad (3)$$

Además, se puede hacer una simplificación, para eliminar la necesidad de recalcular J_1 y J_4. Este procedimiento resulta en ecuaciones de potencia desacopladas por Stoot y Alsac. Los elementos diagonales de J_1, se puede describir como la ecuación (4) (Saadat, 2010).

$$\frac{\partial P_i}{\partial \delta_i} = \sum_{j=1}^{n} |V_i||V_j||V_{ij}|sen(\theta_i - \delta_i + \delta_j) - |V_i|^2 |Y_{ii}| sen\theta_{ii} \quad (4)$$

Reemplazando el primer término de la ecuación anterior con $-Q_i$, resulta en la ecuación (5) (Saadat, 2010):

$$\frac{\partial P_i}{\partial \delta_i} = -Q_i - |V_i|^2 |Y_{ii}| sen\theta_{ii} = -Q_i - |V_i|^2 B_{ii} \quad (5)$$

donde $B_{ii} = |Y_{ii}| sen\theta_{ii}$ es la parte imaginaria de la diagonal de la matriz Y-Bus, la suma de las susceptancias de todos los elementos que inciden en el bus i. En los sistemas típicos de potencia, la susceptancia propia $B_{ii} \gg Q_i$, de tal forma que se puede omitir Q_i.Se obtiene una mayor simplificación al suponer que $|V_i|^2 \approx |V_i|$, lo cual da la ecuación (5) (Saadat, 2010).

$$\frac{\partial P_i}{\partial \delta_i} = -|V_i| B_{ii} \quad (6)$$

En condiciones normales de operación $\delta_j - \delta_i$ es relativamente pequeño. Así que, suponiendo que $\theta_i - \delta_i + \delta_j \approx \theta_{ii}$, por lo que los elementos fuera de la diagonal de J_1 toman la forma de la ecuación (7) y asumiendo que $|V_j| \approx 1$ se convierten en la ecuación (8) (Saadat, 2010).

$$\frac{\partial P_i}{\partial \delta_i} = -|V_i||V_j| B_{ij} \quad (7)$$

$$\frac{\partial P_i}{\partial \delta_i} = -|V_i| B_{ij} \quad (8)$$

De forma similar, los elementos en la diagonal de J_4 pueden aproximarse a partir de la ecuación (9).

$$\frac{\partial Q_i}{\partial |V|_i} = -|V_i|^2 |Y_{ii}| sen\theta_{ii} \sum_{j=1}^{n} |V_i||V_j||V_{ij}| sen(\theta_i - \delta_i + \delta_j) \quad (9)$$

dando como resultado la ecuación (10).

$$\frac{\partial Q_i}{\partial |V|_i} = -|V_i| B_{ij} \quad (10)$$

Con esto, las ecuaciones (2) y (3) toman la forma de las ecuaciones (11) y (12):

$$\frac{\Delta P}{|V_i|} = B' \Delta\delta \quad (11)$$

$$\frac{\Delta Q}{|V_i|} = B'' \Delta|V| \quad (12)$$

donde B' y B'' son la parte imaginaria de la matriz Y-Bus. Como los elementos de la matriz son constantes, deben descomponerse de forma triangular e invertirse una sola vez al comienzo de la iteración. Además B' es del orden de (n-1). Para los Buses PV donde $|V_i|$ y P_i están especificados y Q_i no, se elimina la fila y la columna correspondientes a la Y-Bus. Para B'' el orden es de (n-1-m) siendo m es el número de buses PV. Por lo tanto, en el algoritmo de desacoplado rápido, los cambios sucesivos de voltajes y ángulos se realizan por medio de las ecuaciones (13) y (14) (Saadat, 2010).

$$\Delta\delta = -[B']^{-1} \frac{\Delta P}{|V_i|} \quad (13)$$

$$\Delta|V| = -[B'']^{-1} \frac{\Delta Q}{|V_i|} \quad (14)$$

El algoritmo de este método requiere un mayor número iteraciones que los métodos como el de Newton-Raphson, pero su gran ventaja es el menor tiempo requerido en cada iteración, lo que permite una solución más rápida.

2.2 Módulo excelpy.py

La función principal de este módulo es extraer y almacenar los datos de los archivos de Excel donde se tienen guardados los datos de los buses y líneas. Inicialmente se utilizó un formato texto estándar proporcionado por UW Electrical Engineering. Sin embargo, al usar este tipo de estructura y archivos, hay problemas al editar alguno de los valores ya establecidos. Lo anterior se debe a que, en este tipo de formato, solo se puede extraer datos separados por espacios específicos. Eso quiere decir que al mover un espacio la estructura cambia y dificulta la modificación de la información. Así mismo, representa un problema a futuro, ya que generar nuevos archivos con condiciones específicas, sería un problema. Otra ventaja de manejar datos de esta forma es que se puede separar en hojas de cálculo diferentes la información aportada sobre los buses y las líneas, teniendo un mayor control de la información.

Al tener los datos en archivos bien establecidos, se generó la función "`obtener_datos()`" dentro del módulo. Está función utiliza la librería "`os`" y su función "`listdir`" con la finalidad de poder hacer la lectura de un tipo de archivo por su extensión, dentro de una dirección específica. De esta forma se pueden tener todos los archivos enlistados para poder elegir alguno de los sistemas predeterminados, así como se presenta en la Figura 4.

Figura 4
Lista de archivos con extensión .xlsx.

```
In [1]: runfile('C:/Users/Johnatan/Desktop/SIMULADOR/FDPF/
main.py', wdir='C:/Users/Johnatan/Desktop/SIMULADOR/FDPF')
1.14-Bus.xlsx
2.30-Bus.xlsx
3.57-Bus.xlsx
4.Datos Y Bus.xls
5.Prueba.xlsx

Selección:
```

El último paso que realiza este módulo es eliminar las columnas que no son requeridas para los siguientes procesos, almacenando todo en una matriz.

2.3 Módulos Bus.py y Line.py

La función principal del módulo Bus.py es definir y gestionar los datos para estructurar los buses. Lo anterior permite representar características eléctricas de cada bus como su tipo (por ejemplo, PV, PQ o Slack), magnitudes y ángulos de voltaje, generación y carga, tanto de potencia activa como reactiva, incluyendo los límites de potencia. Este módulo proporciona una estructura clara y accesible para representar los datos de las ecuaciones y para solucionar los flujos de potencia. También permite la inicialización, actualización y recuperación de los atributos de cada bus.

Para encapsular las características de cada bus del sistema se definió una clase con los siguientes atributos: número de identificador del bus, tipo de bus, magnitud del voltaje (V), el ángulo de fase (°), potencia activa (P_g) y reactiva (Q_g) generadas, potencia activa (P_l) y reactiva (Q_l) consumida, límite inferior (Q_{min}) y superior (Q_{max}) de la potencia reactiva.

Por otro lado, la función principal del módulo Line.py es definir y gestionar los datos para estructurar las líneas en el algoritmo. También se utilizó una clase para representar las líneas que conectan los buses, la cual contiene la siguiente información: identificador del bus de origen, identificador del bus de destino, resistencia (R), reactancia (X), susceptancia (B) y relación de transformación de la línea (a).

2.4 Módulo Ymatrix.py

Este módulo tiene como propósito realizar cálculos y gestionar la matriz de admitancias (Y-Bus). La matriz de admitancias es esencial para la solución de flujos de potencia y otros estudios afines. Este módulo inicializa con dos parámetros: **"**`buses`**"** y **"**`lines`**"** los cuales son datos enlazados de los módulos anteriormente mencionados.

El método principal construye la matriz de admitancias, la cual inicia como una matriz de ceros de un tamaño dependiente del número de buses en el sistema. Esta matriz es una matriz compleja porque tanto las admitancias como las suceptancias tienen componentes reales e imaginarios. La inicialización se hace con una matriz de ceros para asegurar que cualquier bus que no esté conectado a otro tenga una admitancia cero. Esto ayuda a evitar un paso más en el proceso, centrándonos solo en los buses interconectados.

Para cada línea se calcula "Y" con la ecuación (15):

$$Y = \frac{1}{r+jx} \qquad (15)$$

donde r es la resistencia y x es la reactancia.

Posteriormente se calcula la susceptancia de la línea utilizando el parámetro "b" de la línea. La susceptancia total es dividida entre dos, porque se distribuye entre los extremos de la línea, lo cual se expresa con la ecuación (16).

$$B = \frac{b}{2} \qquad (16)$$

A continuación, la matriz Y-bus se actualiza en las posiciones diagonales correspondientes a los buses de origen y destino. Incluyendo la admitancia de la línea y la susceptancia, ecuación (17).

$$Y_{bus}[i,i] = Y + jB \qquad (17)$$

Finalmente, los elementos de la Y-bus también se actualizan, los cuales corresponden a la conexión del bus de origen y el bus destino. Estos elementos se actualizan restando la admitancia "Y", tal como se muestra en la ecuación (18)

$$Y_{bus}[i,j] -= Y \qquad (18)$$

Como resultado, se crea una matriz de admitancias que tiene la característica de ser dispersa.

De forma sencilla, este módulo se podría resumir en:

1. Inicializar la matriz Y-bus
2. Calcular los valores de la admitancia y susceptancia para cada línea.
3. Actualizar los elementos dentro y fuera de la diagonal de la matriz Y-bus con las admitancias y susceptancias calculadas.

Todo este proceso es importante, ya que esta matriz es un componente clave para los cálculos y simulaciones relacionadas a los sistemas eléctricos de potencia.

2.5 Módulo FDPF.py

Este módulo implementa el método numérico de desacoplado rápido, para resolver flujos de potencia. Este método permite una convergencia rápida y eficiente en el análisis y solución de flujos de potencia en sistemas eléctricos. A continuación, se detalla la estructura utilizada.

El módulo utiliza la clase **"**`FastDecoupledPowerFlow`**"**. Esta clase utiliza la librería **"**`Numpy`**"** para operaciones numéricas y matriciales, así como los módulos **"**`Bus`**"** y **"**`Line`**"**, creados para representar buses y líneas, así como las librerías **"**`Time`**"** para medir los tiempos de ejecución y **"**`Pyplot`**"**, para visualizar datos de forma ordenada.

La clase se inicializa con los parámetros **"**`buses`**"**, **"**`lines`**"** y **"**Y**"** (la Y-bus o matriz de admitancias). También se utiliza un método para ayudar a inicializar los voltajes y ángulos de los buses, de acuerdo con la información en la Tabla 1

Tabla 1
Condiciones para los tipos de buses.

Tipo de bus	*Inicialización del Voltaje*	*Inicialización del ángulo*
Slack	Bus.voltage_mag	Bus.voltage_ang
PV	Bus.voltage_mag	0
PQ	1	0

Para el Bus Slack, la magnitud del voltaje se inicializa con el valor especificado en el atributo **"**`voltaje_mag`**"** del objeto **"**`bus`**"**. Mientras que el ángulo de voltaje se inicializa con el valor específico en el atributo **"**`voltaje_ang`**"** del objeto **"**`bus`**"**

Para el Bus PV, la magnitud de voltaje se inicializa de la misma forma que en el Slack, mientras que el ángulo de voltaje se inicializa en 0 radianes, ya que no está especificado en los buses PV para la inicialización. Para el Bus PQ, la magnitud de voltaje se inicializa con un valor por defecto de1 p.u., mientras que el ángulo de voltaje se inicia en 0 radianes. Esta inicialización es vital para los cálculos, ya que establece valores iniciales desde los cuales se harán las iteraciones para la convergencia del sistema.

La esencia del módulo FDPF es realizar los cálculos iterativos para resolver problemas de flujo de potencia. Los parámetros utilizados son: la parte real de la Y-bus, la parte imaginaria de la Y-bus, potencia base en MVA, vector que contiene los tipos de buses, los índices de los buses tipo Slack, los índices de los buses tipo PQ y los índices de los buses tipo PV.

Las submatrices B_1 y B_2 se derivan de la parte imaginaria de la Y-bus, conocida como B. Su finalidad es resolver ecuaciones desacopladas para el incremento de las magnitudes de voltaje y los incrementos de ángulos.

Donde B1 es una submatriz de B que excluye las filas y columnas correspondientes a los buses Slack y B2 es una submatriz de B que excluye filas y columnas correspondientes a los buses PV.
La submatriz B1 es usada para resolver los sistemas de ecuaciones lineales desacopladas para los incrementos de ángulo ($\Delta\theta$). La eliminación de las columnas y filas que corresponden a los buses Slack ayudan a simplificar el sistema de ecuaciones, debido a que estos buses tienen ángulos de voltajes fijos.

La submatriz B2 es utilizada para resolver los sistemas de ecuaciones lineales desacopladas para los incrementos de magnitud de voltaje (ΔV). La eliminación de las columnas y las filas correspondientes a los buses PV, permite simplificar el sistema de ecuaciones, porque los buses PC tienen magnitudes de voltajes fijas.

La importancia en la construcción de estas submatrices asegura que el algoritmo pueda converger rápidamente y ayuda a proporcionar resultados precisos para este tipo de análisis.

Los voltajes y potencias se definen los parámetros P_g, potencias activas generadas, Q_g, potencias reactivas generadas, P_l, potencias activas consumidas, Q_l, potencias reactivas consumidas, P_{sp}, potencias activas especifica ($P_g - P_l$) y Q_{sp}, potencias reactivas generadas ($Q_g - Q_l$)

El proceso de iteración para la convergencia es esencial para garantizar que los resultados sean estables y precisos. Para este caso, las magnitudes de voltaje y sus ángulos se ajustan iterativamente, hasta que las diferencias entre las potencias calculadas y las especificas sean menores al límite predefinido. En la iteración se considera la tolerancia inicial para la convergencia, el contador de iteraciones y el factor de relajación para controlar la magnitud de cambios en cada iteración.

Para este caso el ciclo utilizado para realizar las iteraciones se repite hasta que la tolerancia sea menor que un valor umbral predeterminado (1e-5).

En cada iteración, ambas potencias (P y Q) son calculadas para cada bus, utilizando las ecuaciones (19)(20).

$$P_i = \sum_{j=1}^{n} V_i V_j \left(G_{ij} \cos\left(\theta_i - \theta_j\right) + B_{ij} \operatorname{sen}\left(\theta_i - \theta_j\right)\right) \qquad (19)$$

$$Q_i = \sum_{j=1}^{n} V_i V_j \left(G_{ij} \operatorname{sen}\left(\theta_i - \theta_j\right) - B_{ij} \operatorname{sen}\left(\theta_i - \theta_j\right)\right) \qquad (20)$$

donde V_i y V_j son las magnitudes de voltaje en los buses i y j, θ_i y θ_j son los ángulos de voltaje en los buses i y j, G_{ij} y B_{ij} son las partes real e imaginaria de la Y-bus.

Posteriormente, las diferencias entre las potencias calculadas y las específicas se utilizan para determinar el ajuste que se hará en las magnitudes de voltaje y los ángulos de fase, ecuaciones (21)(22).

$$\Delta P_i = P_{sp,i} - P_i \qquad (21)$$

$$\Delta Q_i = Q_{sp,i} - Q_i \qquad (22)$$

Estas diferencias se normalizan dividiendo entre las magnitudes de voltaje y los ángulos.

Los sistemas de ecuaciones lineales generados durante este proceso se resuelven para obtener los incrementos de magnitud de voltaje (ΔV) y los incrementos de ángulo($\Delta\theta$). Estos incrementos se aplican a las magnitudes de voltaje y los ángulos, a esto se le agrega un factor de relajación (α) para mejorar la estabilidad de la convergencia.

La verificación de la convergencia se realiza calculando la norma máxima de los vectores combinados de los incrementos de magnitud de voltaje (ΔV) y los incrementos de ángulo($\Delta\theta$). Todo este proceso se repite hasta que la norma sea mayor o igual que la tolerancia de error (1e-5).

La importancia de este método está en su eficiencia en el tiempo de cada iteración. Este método FDPF es más rápido y menos exigente en el cómputo que otros métodos, como el Newton-Raphson. Debido a su naturaleza desacoplada este método permite una convergencia rápida al separar la potencia reactiva en una ecuación y la potencia activa en otra.

A continuación, se presenta el pseudo código del módulo de desacoplado rápido:

```
ALGORITMO Desacoplado_Rápido
Entrada:
    buses: lista de objetos Bus
    lines: lista de objetos Line
    Y: matriz de admitancia nodal

Inicializar variables:
    nbus: número de buses
    V: vector de voltajes inicializado a 1 p.u.
    del_vec: vector de ángulos inicializado a 0 radianes

Inicializar voltajes:
    PARA cada bus EN buses HACER
       SI bus es slack ENTONCES
          Ajustar magnitud y ángulo del voltaje
       SI bus es PV ENTONCES
          Ajustar magnitud del voltaje
```

```
19. Preparar matrices B1 y B2:
20.    G: parte real de Y
21.    B: parte imaginaria de Y
22.    Eliminar filas y columnas de buses slack de B para B1
23.    Eliminar filas y columnas de buses PV de B para B2
24.
25. Iterar hasta convergencia:
26.    MIENTRAS Tol > 1e-5 HACER
27.       Calcular potencias P y Q para cada bus
28.       Calcular diferencias de potencia delp y delq
29.         Resolver sistemas lineales para obtener delta_theta y
delta_V
30.       Actualizar ángulos y magnitudes de voltaje
31.       Calcular y actualizar la norma de convergencia
32.       Guardar voltajes actuales y norma de convergencia
33.
34.    Calcular resultados finales:
35.       Convertir ángulos de radianes a grados
36.       Guardar resultados finales
37.
38. Funciones adicionales:
39.    plot_voltages: graficar voltajes por iteración para cada bus
40.    save_results_to_database: guardar resultados en base de datos
41.   plot_convergence: graficar norma de convergencia por iteración
42.
43. Salida:
44.    Voltajes y ángulos finales de cada bus
45.
46. FIN ALGORITMO
```

2.6 Módulo main.py

Este módulo es el punto principal para ejecutar el cálculo de flujo de potencia mediante el desacoplado rápido. Además, es el módulo que permite enlazar todos los módulos anteriores y permite que trabajen en conjunto. Es decir, integra varios componentes y coordina la ejecución completa del algoritmo. El código inicia importando las librerías necesarias y definiendo las variables iniciales.

Inicialmente se obtienen los datos de buses y líneas para después ser convertidos en arreglos. Posteriormente, los datos obtenidos se utilizan para crear las listas de objetos **"`Bus`"** y **"`Line`"**. Se crea el objeto **"`YMatrix`"** para almacenar los datos de la matriz de admitancia. Finalmente se crea el objeto **"`FasDecoupledPowerFlow`"**, el cual permite ejecutar el algoritmo de análisis.

La importancia de main.py está en que sirve para integrar los componentes necesarios, los cuales fueron desarrollados con el paradigma de programación orientada a objetos. Desde la importación de datos, hasta la impresión de los resultados, este módulo coordina cada acción del proceso, asegurando una ejecución fluida y eficiente. Esto permite que, en caso de existir un error, éste pueda ser corregido de la mejor forma posible, lo cual ayuda a darle un mejor mantenimiento al software desarrollado.

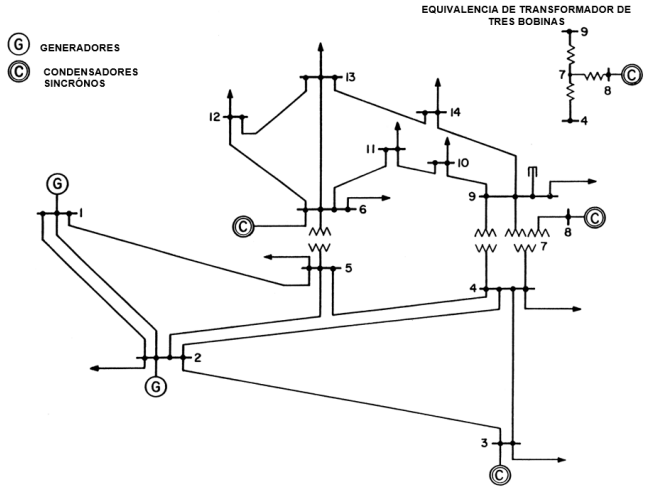
III. RESULTADOS Y DISCUSIÓN

El algoritmo de desacoplado rápido se llevó a cabo para los sistemas de 14 y 30 Buses de la IEEE. La implementación se realizó utilizando Spyder en su versión 5.4.3, el cual es un entorno para utilizar Python, para este caso, en su versión 3.11. Las validaciones de resultados se realizaron comparando los resultados obtenidos en el software OpenDSS en su versión 9.8.0.1. Ambos se ejecutaron en un ordenador portátil con Windows 10, equipado con un procesador Intel Core i7-8650U a 1.90 GHz y 16 GB de RAM.

La Figura 10 muestra el diagrama unifilar de la IEEE que se tomó en cuenta para el primer caso de estudio. Cada uno de los buses está etiquetado y conectado a través de líneas de transmisión. Algunas de estas líneas incluyen trasformadores con una relación de transformación específica. Para este caso, los tipos de buses están distribuidos de la siguiente forma: Bus Slack (Bus 1), Bus PV (Buses 2,3,6 y 8) y Bus PQ (Resto de Buses).

Figura 10

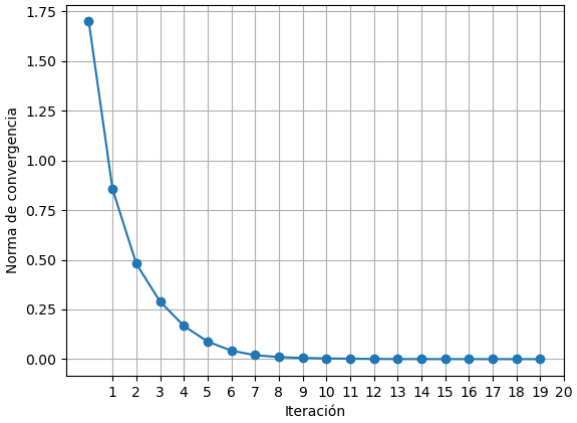
Diagrama unifilar del sistema de 14 Buses de la IEEE (IEEE 14 Bus Test System :Dr. Francisco M. Gonzalez-Longatt, s. f.).



La Figura 11 representa la cantidad de iteraciones para el sistema de 14 Buses, el cual converge después de 0.36 segundos y 19 iteraciones.

Figura 11

Iteraciones para el sistema de 14 Buses de la IEEE.



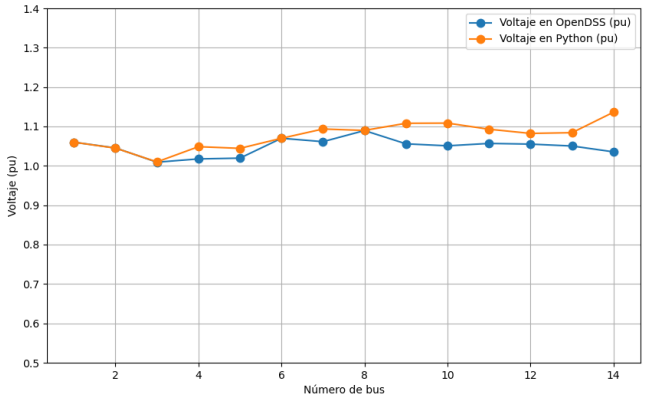
Se realizó la Tabla 2 para comparar los resultados obtenidos en el software libre OpenDSS y con la herramienta computacional mediante el método de Desacoplado Rápido realizada en Python. Lo anterior se realizó con el fin de verificar que los datos generados sean iguales o próximos a los resultados que se esperan. De la Tabla 2 se obtiene que el mayor porcentaje de error para los voltajes es de 9.73% en el Bus 14, mientras que el menor es de 0% para el Bus 1 y el Bus 6.

Tabla 2
Comparativa de Voltajes en OpenDSS y Python para el sistema de 14 Buses de la IEEE.

Bus	Voltaje en OpenDSS (pu)	Voltaje en Python (pu)	Error (%)
1	1.06	1.06	0
2	1.0455	1.045	0.047824
3	1.0093	1.01	0.069355
4	1.0176	1.048643	3.05057
5	1.0196	1.044434	2.435632
6	1.07	1.07	0
7	1.0614	1.093535	3.027567
8	1.0896	1.09	0.036711
9	1.0559	1.108103	4.943934
10	1.0509	1.108522	5.483081
11	1.0569	1.092965	3.412357
12	1.0552	1.082388	2.576621
13	1.0504	1.084224	3.220107
14	1.0355	1.136267	9.731202

Con los datos obtenidos, se realizó, también en Python, la gráfica “Comparación de Voltaje: Python vs OpenDSS”, la cual se observa en la Figura 12.

Figura 12
Gráfica que ilustra los resultados de la Tabla 2.



Los errores observados en los voltajes se deben a la sensibilidad de ciertos buses a los cambios de los parámetros de la red. Esto en particular con los Buses PQ, que pueden mostrar mayores errores debido a su solución iterativa. Además, pueden variar significativamente, debido a la ubicación de los generadores y cargas. Las variaciones de potencia reactiva y la necesidad de mantener límites de voltaje pueden provocar que existan mayores diferencias.

La Tabla 3 se realizó de igual manera, para realizar la comparativa de los ángulos obtenidos. En este caso el mayor error fue de -5.82% para el Bus 3 y el menor error, omitiendo al Bus 1, fue de -0.96% para el Bus 2.

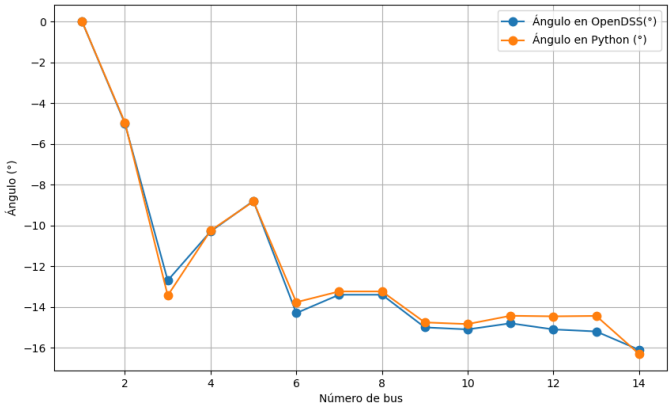
Tabla 3
Comparativa de Ángulos en OpenDSS y Python para el sistema de 14 Buses de la IEEE.

Bus	Ángulo en OpenDSS (°)	Ángulo en Python (°)	Error (%)
1	0	0	0
2	-5	-4.951777	-0.96446
3	-12.7	-13.439596	-5.823591
4	-10.3	-10.25405	-0.446117
5	-8.8	-8.826184	-0.297545
6	-14.3	-13.767115	-3.726469
7	-13.4	-13.238703	-1.203709
8	-13.4	-13.238703	-1.203709
9	-15	-14.757478	-1.616813
10	-15.1	-14.837872	-1.735947
11	-14.8	-14.431585	-2.489291
12	-15.1	-14.458875	-4.245861
13	-15.2	-14.438458	-5.010148
14	-16.1	-16.301714	-1.252882

Al igual que con la tabla anterior, con los datos obtenidos se realizó la gráfica “Comparación de ángulos: Python vs OpenDSS”, la cual está representada en la Figura 13.

Se puede observar que, en este caso, la variación de los ángulos no es significativa, esto debido a que, en una red bien interconectada, los ángulos tienden a ajustarse de manera más uniforme a través de las iteraciones, resultando en menos errores. Además, la distribución de ángulos normalmente es más homogénea y menos susceptible a variaciones abruptas, en especial en sistemas con configuraciones de red bien balanceadas.

Figura 13
Gráfica que ilustra los resultados obtenidos en la Tabla 3.



El Bus Slack mantiene un voltaje fijo de 1.06 p.u. y un ángulo de fase 0°. Este actúa como referencia para todos los buses del sistema. Las variables de ambas potencias en este nodo se ajustan durante el proceso de cálculo del flujo de potencia para equilibrar el sistema.

Los buses PV tienen un voltaje y potencia especificado, sin embargo, su ángulo varía. Los generadores conectados a estos buses ajustan el ángulo para mantener el voltaje deseado. Esta variación se da debido a la generación y carga de estos Buses, se ajustan para equilibrar las potencias activa y reactiva.

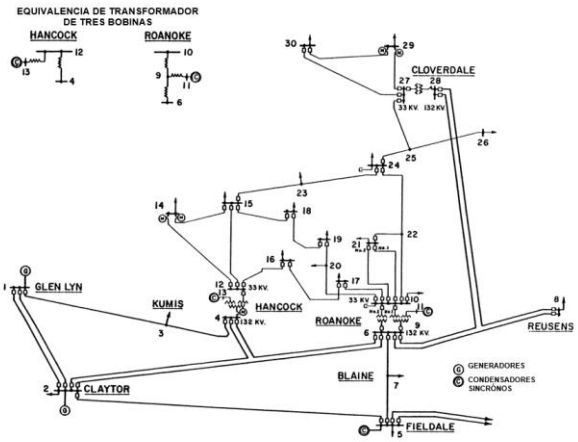
Los buses o nodos PQ tienen voltajes y ángulos variables. Las cargas que están conectadas a estos Buses pueden consumir tanto potencia activa como reactiva, sin embargo, el algoritmo se encarga de satisfacer las demandas de carga. Para estos buses, el voltaje es sensible a cambios, debido a la ausencia de control directo de generación que hay en ellos.

La implementación en Python puede introducir diferencias debido a la precisión numérica y las simplificaciones del método de desacoplado rápido.

El diagrama unifilar de la IEEE (Figura 14) fue el sistema que se tomó en cuenta para el segundo caso de estudio. Cada uno de los buses está etiquetado y conectado a través de líneas de transmisión, algunas de ellas incluyen trasformadores con una relación de transformación específica. Consiste en 3 Buses, 6 generadores y múltiples líneas de transmisión. Para este caso, los tipos de buses están distribuidos de la siguiente forma: Bus Slack (Bus 1), Bus PV (Buses 2,5,8,11 y 13) y Bus PQ (Resto de Buses).

Figura 14

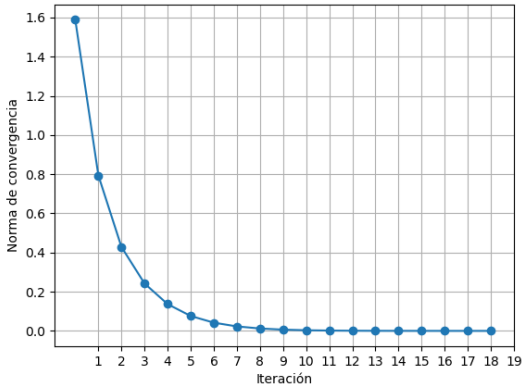
Diagrama unifilar del sistema de 30 Buses de la IEEE (IEEE 30 Bus Test System :Dr. Francisco M. Gonzalez-Longatt, s. f.).



La Figura 15, representa la cantidad de iteraciones para el sistema de 30 Buses, el cual converge después de 1.56 segundos y 18 iteraciones.

Figura 15

Iteraciones para el sistema de 30 Buses de la IEEE.



La Tabla 4 muestra la comparación de Voltajes para el sistema de 30 Buses de la IEEE. Al igual que en los casos anteriores, estos valores se obtuvieron con el software libre OpenDSS y con la herramienta computacional mediante el método de Desacoplado Rápido realizada en Python. En este caso el error mayor fue de 13.50% en el Bus 30 y el error menor, omitiendo el Bus 1, fue 0.0092% de en el Bus 13.

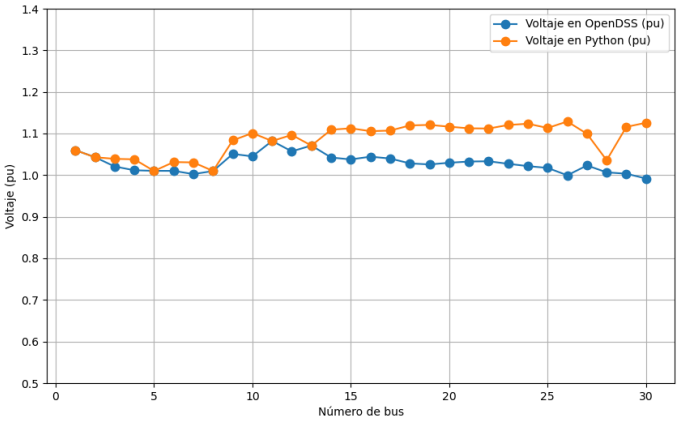
Tabla 4

Comparativa de Voltajes en OpenDSS y Python para el sistema de 30 Buses de la IEEE.

Bus	*Voltaje en OpenDSS (pu)*	*Voltaje en Python (pu)*	*Error (%)*
1	1.06	1.06	0
2	1.0432	1.043	0.019172
3	1.0207	1.0391258	1.805212
4	1.0117	1.0380071	2.600287
5	1.0103	1.01	0.029694
6	1.0102	1.0313628	2.094912
7	1.0025	1.0305831	2.801307
8	1.0098	1.01	0.019806
9	1.0571	1.0836302	3.114492
10	1.0423	1.1009586	5.344809
11	1.0377	1.082	0.009241
12	1.0444	1.0965949	3.736156
13	1.0399	1.071	0.009336
14	1.0282	1.1091303	6.41181
15	1.0257	1.1125349	7.211612
16	1.0297	1.1056185	5.861595
17	1.0451	1.1070527	6.457611
18	1.0327	1.1194345	8.873225
19	1.0333	1.1207092	9.262864
20	1.0272	1.1164504	8.424823
21	1.0216	1.1124718	7.724586
22	1.0173	1.112002	7.616568
23	0.99962	1.1204963	9.082584
24	1.0232	1.1234896	9.973532
25	1.0033	1.1133059	9.437324
26	0.99187	1.1289148	12.934395
27	1.0067	1.0991791	7.425635
28	1.0509	1.0357847	2.889113
29	1.0821	1.11599	11.231935
30	1.0711	1.1258019	13.502969

Con los conjuntos de datos obtenidos, se realizó la Figura 16, la cual es una gráfica que compara los resultados obtenidos con la herramienta de software desarrollada en Python vs OpenDSS. Los 30 Buses que conforman el sistema fueron analizados por ambos programas.

Figura 16
Grafica que ilustra los resultados de la Tabla 4.



Los voltajes en los Buses Slack y PV son controlados directamente con generadores, lo que resulta en valores más estables y cercanos a lo que muestra OpenDSS. En este caso el Bus 1 se mantiene fijo en 1.06 p.u. ya que es el Bus Slack y es la referencia. Mientras que los Buses 2,5,8,11 y 13, son los Buses PV y tienen un porcentaje de error de voltaje menor al 0.02%, esto debido a que como se mencionó anteriormente, el tener un generador conectado directamente, ayuda a una mayor precisión y estabilidad.

Los voltajes en los Buses PQ, tienen una mayor variación debido a las cargas conectadas a ellos y la ubicación que tienen en la red. Los Buses más alejados de los generadores o con cargas grandes, tienden a experimentar mayores caídas de tensión. Por ejemplo, en este análisis, los Buses 27 a 30 tienen los voltajes más bajos debido a la distancia entre los generadores principales y las cargas conectadas.

De igual forma se realizó la Tabla 5, para realizar la comparativa de los ángulos obtenidos. En este caso el mayor error fue de -20.32% para el Bus 28 y el menor error, omitiendo al Bus 1, fue de -0.0225% para el Bus 24.

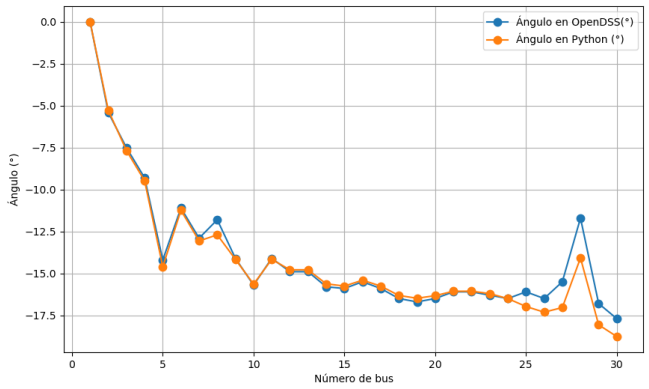
Tabla 5
Comparativa de Ángulos en OpenDSS y Python para el sistema de 30 Buses de la IEEE.

Bus	*Ángulo en OpenDSS (°)*	*Ángulo en Python (°)*	*Error (%)*
1	0	0	0
2	-5.4	-5.266026	-2.480991
3	-7.5	-7.671063	-2.28084
4	-9.3	-9.47666	-1.89957
5	-14.2	-14.604977	-2.851951
6	-11.1	-11.237376	-1.237622
7	-12.9	-13.06435	-1.274031
8	-11.8	-12.699445	-7.622415
9	-14.1	-14.161882	-0.438879
10	-15.7	-15.642843	-0.364057
11	-14.1	-14.161882	-0.438879
12	-14.9	-14.78599	-0.765168
13	-14.9	-14.78599	-0.765168
14	-15.8	-15.622883	-1.120994
15	-15.9	-15.754921	-0.912447
16	-15.5	-15.405	-0.612903
17	-15.9	-15.756139	-0.904786
18	-16.5	-16.316633	-1.111315
19	-16.7	-16.483658	-1.295461
20	-16.5	-16.328957	-1.036624
21	-16.1	-16.065117	-0.216665
22	-16.1	-16.062943	-0.230168
23	-16.3	-16.204514	-0.585804
24	-16.5	-16.503716	-0.022521
25	-16.1	-16.974754	-5.433255
26	-16.5	-17.314304	-4.935176
27	-15.5	-17.037642	-9.920271
28	-11.7	-14.078484	-20.328923
29	-16.8	-18.068233	-7.549006
30	-17.7	-18.768816	-6.0385

En el caso de los ángulos, el Bus Slack se mantiene fijo en 0°. Mientras que en los Buses PV, los ángulos se ajustan para equilibrar la potencia activa, lo cual resulta en valores relativamente consistentes y bajos. Para los Buses PQ los ángulos muestran variaciones debido a la distribución de la carga y las características de la red. En este caso el Bus 28 resultó con un error del -20.32%, este Bus está conectado a través de una línea al Bus 27, mientras que éste a los Buses 29 y 30. Estos Buses son los más alejados de los generadores y del Bus Slack. Una ubicación más alejada en la red significa que cualquier fluctuación en la generación o en la carga tendrá un impacto mayor en los parámetros de ese Bus, esto incluye los ángulos.

Al igual que con la tabla anterior, con los datos obtenidos se realizó la gráfica que compara los resultados de la herramienta de software desarrollada en Python vs OpenDSS, para el sistema de 30 Buses de la EEE, la cual está representada en la Figura 17.

Figura 17
Gráfica que ilustra los resultados obtenidos en la Tabla 5.



La variación en los ángulos también se puede sustentar con el método de Desacoplado Rápido, debido que la potencia activa y reactiva se resuelven de forma independiente. En consecuencia, cualquier variación en la potencia activa debido a algún cambio significativo en la generación o en la carga, puede resultar en variaciones en los ángulos de fase. De igual forma, una variación de la potencia reactiva afecta directamente al voltaje.

IV. CONCLUSIONES

Este trabajo presentó el desarrollo de una herramienta de computacional desarrollada en Python para la solución de flujos de potencia usando el método de desacoplado rápido, además de la comparativa con el software OpenDSS.
La comparación de los resultados muestra que la herramienta de software implementada en Python produce muy buenos resultados en términos de voltajes y ángulos en los Buses de cada uno de los dos sistemas.

La implementación de métodos numéricos en Python ofrece varias ventajas, como lo son la flexibilidad y adaptabilidad del código, la facilidad de modificación y la capacidad de integrar librerías que ayuden a la manipulación de datos y operaciones numéricas, como los son `NumPy`, `SciPy` y `Pandas`. Estas características no solamente facilitan la optimización de recursos computacionales, sino que también promueven la reproducibilidad y la colaboración en futuras investigaciones.
Además, otra ventaja de usar Python es el costo de mantenimiento, ya que al ser éste un software libre, es totalmente gratuito, por lo que cualquier persona puede utilizarlo.

Basado en el análisis efectuado, se concluye que Python es una herramienta computacional eficaz y que si bien, los resultados no fueron totalmente exactos en la comparación, al ser una herramienta computacional, existe la posibilidad de buscar métodos para optimizar los resultados, así como la convergencia, velocidad y memoria utilizada, entre otros, lo cual deja un amplio margen para mejorar.

V. AGRADECIMIENTOS

E.J. Muñoz-Luna agradece al Consejo Nacional de Humanidades, Ciencias y Tecnologías por el respaldo financiero recibido a través del “Programa de Becas Nacionales para Estudios de Posgrado” con número de solicitud 2022-000018-02NACF-08769.
Los autores desean expresar su agradecimiento al TecNM y al Instituto Tecnológico de La Laguna por el apoyo económico proporcionado para la realización de esta investigación

VI. REFERENCIAS

Karimi, M., Shahriari, A., Aghamohammadi, Marzooghi, H., & Terzija, V. (2019). Application of Newton-based load flow methods for determining steady-state condition of well and ill-conditioned power systems: A review. *International Journal Of Electrical Power & Energy Systems,* 113, 298-309. https://doi.org/10.1016/j.ijepes.2019.05.055

Afolabi, O. A., Ali, W. H., Cofie, P., Fuller, J., Obiomon, P., & Kolawole, E. S. (2015). *Analysis of the Load Flow Problem in Power System Planning Studies*. Energy And Power Engineering, 07(10), 509-523. https://doi.org/10.4236/epe.2015.710048

Volkov, K. (2020). Computational Models in Engineering. En IntechOpen eBooks. https://doi.org/10.5772/intechopen.77484

Yanez, T., & Rolando, E. (2019). *Simulación de flujos de potencia en sistemas eléctricos de potencia usando métodos completos, desacoplados y linealizados.* https://dspace.ups.edu.ec/bitstream/123456789/17625/1/UPS%20-%20ST004217.pdf

Jalolov, T. S. (2023). Solving Complex Problems in Python. American Journal of Language, *Literacy and Learning in STEM Education*, 1(9), 481–484. https://grnjournal.us/index.php/STEM/article/view/1584

Sharma, R., & Dhillon, J. (2021). PyPSA: Open Source Python Tool for Load Flow Study. Journal Of Physics. Conference Series, 1854(1), 012036. https://doi.org/10.1088/1742-6596/1854/1/012036

Phongtrakul, T., Kongjeen, Y., & Bhumkittipich, K. (2018). Analysis of Power Load Flow for Power Distribution System based on PyPSA Toolbox. 2018 15th International Conference On Electrical Engineering/Electronics, Computer, Telecommunications And Information Technology. https://doi.org/10.1109/ecticon.2018.8619954

Cui, H., & Li, F. (2018). ANDES: A Python-Based Cyber-Physical Power System Simulation Tool. 2018 North American Power Symposium (NAPS). https://doi.org/10.1109/naps.2018.8600596

Cui, H., Li, F., & Tomsovic, K. (2021). Hybrid Symbolic-Numeric Framework for Power System Modeling and Analysis. IEEE Transactions On Power Systems, 36(2), 1373-1384. https://doi.org/10.1109/tpwrs.2020.3017019

Thurner, L., Scheidler, A., Schäfer, F., Menke, J., Dollichon, J., Meier, F., Meinecke, S., & Braun, M. (2018). Pandapower—An Open-Source Python Tool for Convenient Modeling, Analysis, and Optimization of Electric Power Systems. IEEE Transactions On Power Systems, 33(6), 6510-6521. https://doi.org/10.1109/tpwrs.2018.2829021

Año: 2024. Volumen: 1
Numero:10. ISSN: 2448-623X

Huang, M., Chen, R., & Zhong, H. (2020). Research on an Automatic Conversion Method of Power Flow Input Data from PSD-BPA to Pandapower. 2020 IEEE 4th Information Technology, Networking, Electronic And Automation Control Conference (ITNEC). https://doi.org/10.1109/itnec48623.2020.9084979

Power Systems Test Case Archive - UWEE. (s. f.). https://labs.ece.uw.edu/pstca/

IEEE 14 Bus Test System :Dr. Francisco M. Gonzalez-Longatt. (s. f.). https://fglongatt.org/OLD/Test_Case_IEEE_14.html

IEEE 30 Bus Test System :Dr. Francisco M. Gonzalez-Longatt. (s. f.). https://fglongatt.org/OLD/Test_Case_IEEE_30.html

Saadat, H. (2010). Power System Analysis. Psa Pub.

VII. AUTORES

Elí Johnatan Muñoz Luna

https://orcid.org/0009-0001-5990-7139

Cristian Luna Aguilera

https://orcid.org/0009-0002-3589-8095

Concepción Hernández Flores

https://orcid.org/0000-0002-4757-5309

Marco Antonio Arjona López

https://orcid.org/0000-0003-1826-4066