Trabajo Fin de Máster
Máster Universitario en Ingeniería Industrial

# Flujo de Cargas para Redes Desequilibradas Basado en Residuos de Intensidades

Autor: Raúl Lago Díaz

Tutor: Álvaro Rodríguez del Nozal

Esther Romero Ramos

**Dpto. Ingeniería Eléctrica**
**Escuela Técnica Superior de Ingeniería**
**Universidad de Sevilla**

Sevilla, 2021

Trabajo Fin de Máster
Máster Universitario en Ingeniería Industrial

# Flujo de Cargas para Redes Desequilibradas Basado en Residuos de Intensidades


Autor:

Raúl Lago Díaz

Tutor:

Álvaro Rodríguez del Nozal

Profesor Ayudante Doctor

Esther Romero Ramos

Catedrática de Universidad

Dpto. Ingeniería Eléctrica
Escuela Técnica Superior de Ingeniería
Universidad de Sevilla



Sevilla, 2021

Trabajo Fin de Máster: Flujo de Cargas para Redes Desequilibradas Basado en Residuos de Intensidades

Autor: Raúl Lago Díaz
Tutor: Álvaro Rodríguez del Nozal
Esther Romero Ramos

El tribunal nombrado para juzgar el trabajo arriba indicado, compuesto por los siguientes profesores:

Presidente:

Vocal/es:

Secretario:

acuerdan otorgarle la calificación de:

El Secretario del Tribunal

Fecha:

# Agradecimientos

En primer lugar, querría agradecer a mis tutores Esther y Álvaro toda su disponibilidad y dedicación. Igualmente, quería dar las gracias a todos los profesores del Departamento de Ingeniería Eléctrica que han aportado su granito de arena en la elaboración de este trabajo. Gracias a todas las personas que me han acompañado durante estos dos años en los que he podido poner el broche de oro a mi formación en ingeniería. En especial, gracias a Celia porque es la que más sufre el tiempo que no he podido dedicarle para llevar a cabo este trabajo. Gracias a Juan y Sandra, que han sabido entender que no podía hacer todos los planes que nos hubiese gustado durante estos últimos meses y por entender mi ausencia en las reuniones de los fines de semana. Gracias a todos mis amigos, que me han animado, aconsejado, apoyado, querido y acompañado todo este tiempo. Y finalmente, gracias a mis padres y a mi hermana, que , aunque desde la distancia, me recuerdan día a día que cuento con todo su apoyo y cariño.

*Raúl Lago Díaz*
*Sevilla, 2021*

# Resumen

Este trabajo desarrolla una formulación alternativa del método Newton-Raphson aplicado al problema del flujo de cargas, ampliando las ecuaciones del algoritmo a través de la incorporación de residuos de intensidades. Se detallará en primer lugar el algoritmo para redes trifásicas equilibradas utilizando el circuito monofásico equivalente para después ampliar la formulación para redes trifásicas desequilibradas con cuatro conductores. Después, partiendo de las ecuaciones que describen el algoritmo, se añadirá una formulación específica que permite trabajar con nudos PV. Por otro lado, se presentarán las ecuaciones que permiten incorporar transformadores al modelo.

Una vez desarrollada toda la formulación se lleva esta a la práctica, implementando una herramienta de programación en el lenguaje Python capaz de ejecutar el algoritmo. Esta herramienta se validará utilizando varios ejemplos y comparando los resultados con el programa OpenDSS. Por último, se realizará una simulación utilizando como modelo la red de baja tensión del CIGRE.

# Abstract

This work presents an alternative formulation of the Newton-Raphson method applied to the load flow problem, extending the equations of the algorithm through the incorporation of current residuals. The algorithm for balanced three-phase networks using the equivalent single-phase circuit will first be detailed and then the formulation will be extended to unbalanced three-phase networks with four conductors. Then, starting from the equations describing the algorithm, a specific formulation will be added to allow working with PV nodes. On the other hand, the equations that allow transformers to be incorporated into the model will be presented.

Once the entire formulation has been developed, it will be put into practice by implementing a programming tool in the Python language capable of executing the algorithm. This tool will be validated using several examples and comparing the results with the OpenDSS programme. Finally, a simulation will be carried out using the CIGRE low voltage network as a model.

# Índice

# Índice de Figuras

# Índice de Tablas

# 1 Introducción

En este trabajo se describirá la programación en Python de un algoritmo para calcular el flujo de cargas en redes trifásicas desequilibradas de cuatro hilos. A lo largo de este capítulo de introducción se describirá la motivación y objetivos de este trabajo, asi como algunos conceptos básicos que mejoren la comprensión de este documento.

## 1.1 Motivación de este trabajo

Los sistemas eléctricos de potencia son parte fundamental para satisfacer las necesidades energéticas de las sociedades actuales. Todos los niveles de una economía moderna requieren en mayor o menor medida un suministro de energía para llevar a cabo su actividad. El rápido desarrollo económico mundial en el último medio siglo, ligado al aumento de la población ha disparado las necesidades la demanda de energía. Por otro lado, nuevos descubrimientos científicos como el calentamiento global y su relación directa con las emisiones de gases de efecto invernadero, ha concienciado a los actores energéticos a realizar una transición hacia fuentes primarias de energías diferentes a aquellas provenientes de combustibles fósiles (petróleo, gas, carbón).

La Comisión Europea presentó en noviembre del 2016 el paquete 'Energía Limpia para todos los europeos', cuyas propuestas y medidas tienen como finalidad acelerar la transición energética hacia una energía limpia en línea con el cumplimiento de los objetivos establecidos en el Acuerdo de París 2015 contra el cambio climático [1]. En esas medidas se definieron los objetivos energéticos europeos a corto y medio plazo [2], reflejados en la Figura 1.1:

| | | | | |
|---|---|---|---|---|
| **Objetivos 2020** | **20%** reducción de emisiones de GEI frente a niveles de 1990 | **20%** de energías renovables en la UE | **20%** de mejora de la eficiencia energética | |
| **Objetivos 2030** | **40%** reducción de emisiones de GEI frente a niveles de 1990 | **32%** de energías renovables en la UE | **32,5%** de mejora de la eficiencia energética | **15%** para interconexiones eléctricas |
| **Objetivos 2050** | **85-90%** reducción de emisiones de GEI frente a niveles de 1990 | | | |

**Figura 1.1** Objetivos energéticos europeos siglo XXI.

A nivel nacional, el Ministerio para la Transición Ecológica y el Reto Demográfico publicó en febrero del 2019 su borrador del Plan Nacional Integrado de Energía y Clima 2021-2030 (PNIEC 2021-2030) que ha sido remitido a la Comisión Europea y cuya versión definitiva deberá aprobarse antes de final de año. Los objetivos planteados en el PNIEC 2021-2030 para la necesaria transición energética consiguiendo una economía prácticamente descarbonizada en 2050 son [3]:

- una reducción de emisiones del 21 % respecto de los niveles de 1990
- una cuota de renovables del 42 % sobre la energía final
- una mejora de la eficiencia energética del 39,6 %
- se prevé que la contribución de las renovables en el mix eléctrico alcance el 74 % en el 2030.

Para lograr estos objetivos, se han propuesto varias desafíos que pueden ayudar a dar solución a los problemas que pueden surgir debido a este proceso. Entre los más importantes, tenemos: conseguir una mayor flexibilidad por parte de la demanda del sistema eléctrico, mejorar la gestionabilidad de la generación renovable o integrar la generación distribuida.

Todos ellas tienen algo en común y es que están mayormente relacionadas con la red de distribución dentro de los sistemas eléctricos de potencia. Por ello, es vital la investigación y desarrollo de herramientas de predicción que permitan la operación, monitorización y control de esta parte del sistema para poder medir, verificar y gestionar las variables y parámetros de la red y sacar conclusiones. Este trabajo busca realizar un pequeño aporte a esta tarea, mediante la programación en Python de un algoritmo para resolver el problema de flujo de cargas en redes trifásicas de cuatro hilos (configuración típica en redes de distribución).

## 1.2 El problema del flujo de cargas

En ingeniería eléctrica, el problema de flujo de potencia o flujo de cargas (load flow o power flow) es un problema de análisis numérico que que consiste en obtener las variables eléctricas que definen las condiciones en régimen permanente de un sistema de potencia [4]. El problema de flujo de cargas clásico calcula las tensiones y ángulos en todos los nudos y los flujos de potencia por todos los elementos (ramas, transformadores...) de un sistema de eléctrico de potencia, aunque a lo largo de este trabajo veremos que existen variantes del problema clásico.

Para ello se toma como datos de partida el consumo, la inyección de potencia en todos los nudos de la red y el modelo eléctrico de la misma. Es ampliamente utilizada tanto en planificación como en explotación en tiempo real de sistemas de potencia.

Un ejemplo del problema de flujo de cargas se reprsenta en la Figura 1.2:

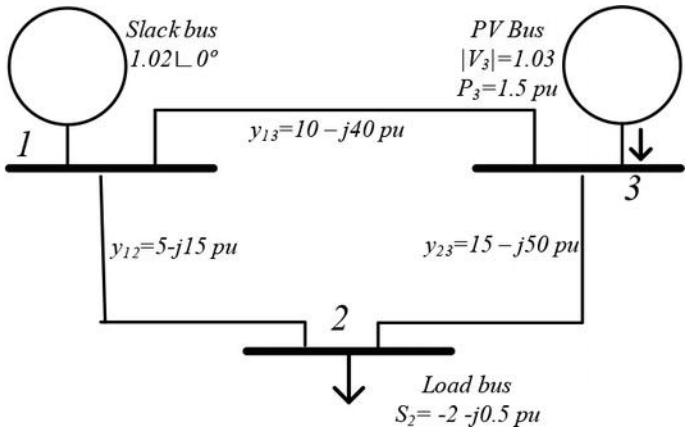


**Figura 1.2** Ejemplo de flujo de cargas para una red de tres nudos .

Es una herramienta muy poderosa, ya que permite calcular y conocer el estado de las variables eléctricas de un sistema de potencia. La investigación en torno a este tema ha sido amplia y profusa, en gran parte debido a que el conocimiento del estado de un sistema de potencia es de vital importancia, tanto para el estudio de los elementos que lo conforman como para la planificación y expansión futura. Resulta trascendental la información que se puede obtener de un sistema eléctrico de potencia al analizar los estudios de flujos de potencias, puesto que los resultados constituyen una fuente para la comparación de los cambios en los flujos de carga y los voltajes bajo condiciones anormales de operación de la red.

Tradicionalmente, el problema del flujo de cargas ha estado reservado únicamente para las redes de alta tensión o redes de transporte de los sistemas eléctricos mientras que las redes de distribución se consideraban "plug-and-play". Esto es así en parte por el hecho de que la red de transporte ha sido hasta hace pocos años la que ha concentrado la mayor parte de la atención y recursos en cuanto a monitorización, operación y control de los sistemas eléctricos de potencia. Por otro lado, las redes de transporte presentan una serie de características que simplifican el problema del flujo de cargas:

- Las redes de transporte de alta tensión son sistemas equilibrados en tensiones, lo cual permite modelar la red como un sistema monofásico equivalente.
- Tienen una alta potencia de cortocircuito, lo que aumenta su "inercia" eléctrica y provoca muy pocas variaciones ante desequilibrios o faltas en la red. Por ejemplo, los consumidores electrointensivos o cargas que producen grandes desequilibrios como los trenes de alta velocidad se conectan directamente a la red de transporte por este motivo.
- Los órdenes de magnitud de las variables eléctricas suelen ser similares, lo que añadido a los saltos en niveles de tensión hace que sea muy interesante trabajar en por unidad.
- Su ratio R/X es muy bajo, lo que permite simplificar la red a un modelo desacoplado P-$\theta$, Q-V

Estas caracteristicas de las redes de alta tensión permiten elaborar métodos como el FDLF (Fast decoupled load flow) que reducen enormemente el coste computacional, permitiendo evaluar redes de gran dimensión (del orden de miles de nudos) en tiempo real.

### 1.2.1 Formulacion del problema del flujo de cargas

El proceso básico del problema de flujo de cargas es el siguiente []

1. Crear un modelo de la red que incluya todos los elementos de la misma (nudos, ramas, transformadores, cargas...etc)
2. Inicializar las variables eléctricas (tensión, corriente) del modelo
3. Definir las variables eléctricas en todos los nudos de la red. Las ecuaciones matemáticas planteadas para la resolución de la red han de ser resueltas mediante un algoritmo que permita obtener el estado del sistema, estado que queda normalmente definido por las tensiones complejas en todos los nudos. Toda otra magnitud que se quiera conocer vendrá definida por dichas tensiones. El algoritmo deberá ser implementado a través de un programa por ordenador capaz de manejar gran cantidad de datos y realizar operaciones que, aunque no son complejas, se repiten normalmente de manera iterativa y requieren el manejo de matrices de grandes dimensiones lo que hace la resolución manual muy tediosa y en ocasiones imposible.
4. Calcular todas las magnitudes de interés como flujos de potencia activa y reactiva, intensidades por las ramas, pérdidas... etc

En la búsqueda de la solución óptima, se ha de tener en cuenta que dependiendo del nivel de exactitud requerido en la solución se adoptará un modelo más o menos preciso del sistema. Además, es determinante la elección de un algoritmo robusto y eficiente que resuelva el sistema de ecuaciones planteado de manera que implique el menor número de operaciones posible con la finalidad de una mayor rapidez en la búsqueda de respuesta, para la cual, también será relevante la óptima programación de dicho algoritmo

## 1.3 Características de redes de distribución

Las redes de distribución europeas suelen tener las siguientes características:

- Están compuestas por uno o varios niveles de media tensión, con configuración de tres fases y tres hilos y por un nivel de baja tensión con configuración de tres fases y cuatro hilos (fases+neutro).
- Las cargas finales suelen ser monofásicas intentando que sean repartidas de manera equilibrada entre cada fase. Las redes de media tensión son equilibradas pero las redes de baja tensión son generalmente desequilibradas, por lo que la intensidad que circula por el neutro suele ser diferente de cero. A diferencia de las redes de transporte, con las redes de distribución no es posible utilizar el modelo monofásico equivalente, ya que las tensiones e intensidades pueden ser diferentes para cada fase.

- Predominan los transformadores trifásicos en cabecera, generalmente del tipo triángulo-estrella y con relaciones de transformación 20kV/400V.
- Son redes malladas pero por simplicidad y ahorro se explotan de manera radial.

De cara a la formulación del problema del flujo de cargas, las redes de distribución presentan menores potencias de cortocircuito y un ratio $\frac{R}{X} \sim 1$, por lo que como se ha comentado anteriormente, algunas de las hipótesis y simplificaciones que se aplican a las redes de transporte para resolver el flujo de cargas no son aplicables para redes de distribución.

Algunos de los método utilizados para calcular el flujo de cargas en redes de distribución son:

- **FBS (Forward-Backward Sweep o Barrido del árbol)** Este método esta formado por dos partes. En primer lugar, se parte de un perfil plano de tensiones y se realiza un barrido aguas arriba para calcular las tensiones de los nudos de la red. En segundo lugar, se realiza un barrido aguas abajo para calcular las intensidades de las ramas de la red. Este método es muy eficiente para redes radiales.
- **Newton-Raphson**: se calcula el flujo de cargas mediante un proceso iterativo que se describirá en el siguiente capítulo.

## 1.4 Objetivos

El objetivo de este trabajo es desarrollar una herramienta de programación en lenguaje Python que sea capaz de resolver el problema del flujo de cargas para sistemas trifásicas desequilibrados a cuatro hilos.
Para ello,

1. Se describirá la formulación matemática del método de Newton-Raphson,
2. Se describirá la formulación matemática del algoritmo sobre el que se va a trabajar: *"Método Rectangular Aumentado"*[5],
3. Se describirá la herramienta de programación desarrolllada en Python,
4. Se validará la herramienta y los modelos teóricos implementados, comparando los resultados obtenidos con el programa *Open DSS*, para una misma red.

# 2 Método de Newton-Raphson aplicado al problema de flujo de cargas

En este capítulo se detallará la formulación del método de Newton-Raphson aplicado al problema del flujo de cargas en sistemas eléctricos de potencia. Se abordará su formulación y base matemática y sus aplicaciones específicas en el campo de la ingeniería eléctrica.

## 2.1 Descripción del método

Encontrar las soluciones de una cierta ecuación es uno de los problemas matemáticos más comunes. Sin embargo, hallar la solución de una ecuación puede ser un proceso no tan trivial como podemos pensar. No tenemos métodos que nos permitan obtener las soluciones de todas las ecuaciones posibles. Por ejemplo, no hay ningún método que nos de las soluciones exactas a esta ecuación:

$$x \cdot 2^x = 1$$

La aproximación más aceptada en estos casos es utilizar algún método o técnica que nos proporcione una solución lo suficientemente buena, es decir, cerca de la solución exacta, como para que pueda servirnos en nuestro problema.

Unos de esos métodos es el método de Newton-Raphson. Es un método iterativo con el que podemos encontrar aproximaciones de soluciones de ecuaciones no lineales [6]. El método parte de un valor inicial que se introduce en una expresión relacionada con la ecuación que queremos resolver, obteniendo así un resultado. Partiendo de ese resultado se introduce nuevamente en la misma expresión, obteniendo un nuevo resultado, y así sucesivamente. La clave del método es conseguir en cada iteración una aproximación a la solución real mejor que la que teníamos anteriormente. Para determinar cuando la solución es lo suficientemente exacta, se define un umbral de tolerancia, que marca el error aceptable entre la solución obtenida y la solución exacta.

Una de las cuestiones más importantes a la hora de aplicar este método, es la elección del valor inicial de la solución o soluciones de nuestro problema, ya que cuanto más alejado del cero sea el valor inicial, la probabilidad de que el método no converja aumenta.

El método tiene la siguiente formulación. Sea $f : [a,b] \to \Re$, derivable en todo $[a,b]$, tomando un valor inicial $x_0$ se define para cada iteración n:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{2.1}$$

Donde $f'$ es la derivada de f, $x_n$ es la solución en la iteración actual y $x_{n+1}$ es la solución en la iteración siguiente.

La forma más habitual de aplicar el método es mediante la expansión en series de Taylor de una o varias variables, despreciando los términos mayores de grado 2.

$$f(x_{n+1}) = f(x_n) + f^{'}(x_n)(x_{n+1} - x_n) \tag{2.2}$$

Donde $f^{'}(x_n)$ es la derivada de la función (o jacobiano para funciones multivariable como será en este trabajo) y $(x_{n+1} - x_n)$ es el residuo o diferencia entre el valor buscado y el valor actual del proceso iterativo.

La solución de la ecuación estará en $f(x_{n+1}) = 0$, por lo que se busca el valor de $x_{n+1}$ que haga que la función sea nula. De esa manera obtenemos la expresión (2.3)

$$-f(x_n) = f^{'}(x_n)(x_{n+1} - x_n) \tag{2.3}$$

Por lo que en cada iteración se calcula la derivada de la función (o jacobiano para funciones multivariable como será en este trabajo) y el residuo. El siguiente paso es sumar este residuo al valor de la variable en la iteración anterior y se sigue iterando hasta que el residuo sean menor que un cierto valor.

## 2.2 Aplicaciones y ventajas de NR en el problema del flujo de cargas

La parametrización y modelado de las componentes de un sistema eléctrico de potencia se ha vuelto cada vez más compleja en los últimos años. Esto es debido a la incorporación de multitud de nuevos actores a los sistemas eléctricos y la expansión de los mismos debida al aumento de la demanda de energía. Una de las soluciones pensadas para dar respuesta a esta problemática ha sido la creación de diferentes algoritmos pensados para trabajar mediante métodos numéricos. Algunos de los métodos más usados se basan en los métodos de Gauss-Seidel y de Newton-Raphson. Abordaremos este último método en este capítulo.

El método de Newton-Raphson se utiliza para resolver flujos de potencia debido a su alta velocidad de convergencia y eficiencia. La principal ventaja frente a otros métodos numéricos es la optimización de los tiempos de cálculo, muy importante en aplicaciones como la estimación de estado de un sistema eléctrico de potencia [7].

La aplicación de este método y sus variantes ha sido una de las contribuciones más exitosas al problema del flujo de cargas. Este aporte fue posible gracias al desarrollo de las técnicas de programación para la manipulación eficiente de grandes matrices y en particular de los métodos de eliminación ordenada, aplicados a matrices dispersas.

Como se ha comentado más arriba, el método Newton-Raphson se basa en la expansión en series de Taylor de una o varias variables. Las características de convergencia hacen que este método sea muy conocido principalmente en sus modalidades desacopladas, aplicadas fundamentalmente a las redes de transporte, donde R«X.

La convergencia cuadrática hace que sea matemáticamente superior al método de Gauss-Seidel, lo que se traduce en más eficiencia de cálculo. Además el tamaño de la red no afecta al número de iteraciones necesarias para que se alcance una solución aceptable, siendo esta otra característica que lo hace superior al método de Gauss-Seidel donde el tamaño del sistema en estudio afecta la cantidad de iteraciones. A pesar de que el algoritmo tiene que invertir la matriz Jacobiana para cada iteración y ello se refleja en el tiempo de cálculo, este método es mucho más veloz que el método de Gauss-Seidel y la convergencia se alcanza en un número reducido de iteraciones.

## 2.3 Formulación Matemática del Problema de Flujo de Cargas

### 2.3.1 Matriz de admitancias de nudos

En este apartado se describirá la formulación matemática del método de Newton Raphson aplicado al problema del flujo de cargas. Esta formulación se ha obtenido de [8] [9].

Suponiendo un modelo monofásico equivalente en $\pi$ como el de la Figura 2.1:

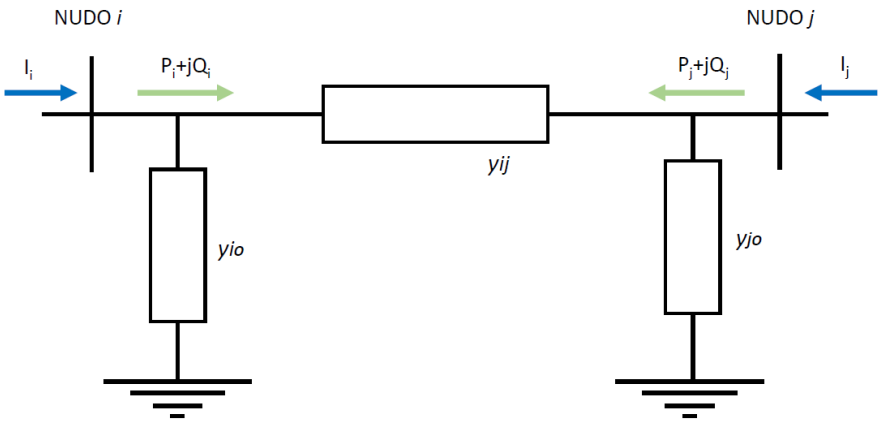


**Figura 2.1** Modelo monofásico equivalente en pi de una linea.

Para una red de m nudos, $y_{ij}$ es la admitancia seria entre dos nudos cualesquiera y $y_{io}$ es la admitancia serie admitancia paralelo entre la línea y el neutro del sistema. Si se representa una red genérica según el sistema que aparece en la Figura 2.2, se observa que las intensidades inyectadas en los nudos $I_i$ están relacionadas con las tensiones complejas en los mismos a través de la matriz de admitancias de nudos $Y_{BUS}$.

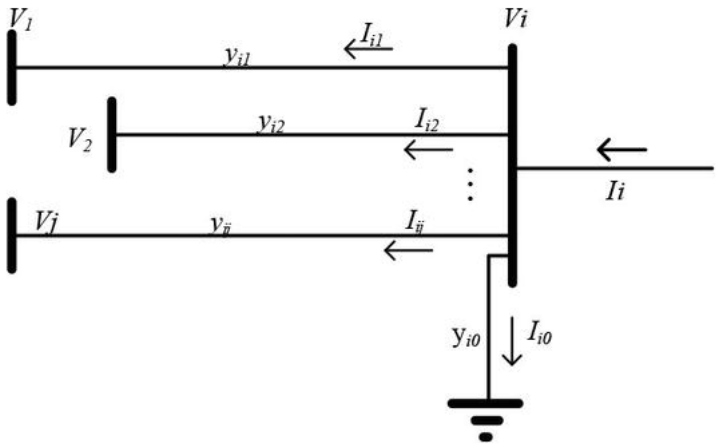


**Figura 2.2** Red m nudos.

De esta manera, podemos calcular las intensidades inyectadas en cada nudo como:

$$I_i = \sum_{i=1}^{m} Y_{BUS} \cdot U_i \quad \text{i =1,2,...m,} \tag{2.4}$$

Donde $U_i$ es la tensión en el nudo i.

$$I_i = \sum_{j \in i} (y_{ij} + y_{i0}) \cdot V_i - \sum_{j \in i} y_{ij} \cdot V_j, \tag{2.5}$$

$V_i, V_j$ son los módulos de las tensiones $U_i, U_j$.

Podemos diferenciar las admitancias entre una admitancia $Y_{ii}$ y otra $Y_{ij}$:

$$Y_{ii} = \sum_{j \in i} (y_{ij} + y_{i0}), \tag{2.6}$$

$$Y_{ij} = -Y_{ji}, \tag{2.7}$$

Estos términos forman la matriz de admitancias de nudos $Y_{BUS}$ cuya dimensión dependerá del número de nudos de la red que se estudie. Los elementos de esta matriz son:

- Elementos de la diagonal principal $(j = 1)$:

$$Y_{ii} = \sum_{i=1}^{m} \frac{1}{R + jX} + \frac{Y}{2} + \frac{1}{R_g}. \tag{2.8}$$

  Como se comentó anteriormente, se ha tomado como referencia un modelo en pi de la linea, por lo que la admitancia paralelo es $Y/2$. $R_g$ hace referencia a la resistencia de la puesta a tierra.

- Elementos fuera de la diagonal principal $(j \neq 1)$:

$$Y_{ij} = -\frac{1}{R + jX}, \tag{2.9}$$

### 2.3.2 Restricciones no lineales del problema

En redes de distribución, que son el tipo de redes con las que trabajaremos en este trabajo fundamentalmente existen tres tipos de modelo de cargas: Potencia constante, Intensidad constante o Impedancia constante. En este trabajo se modelarán las cargas como de Potencia constante. De esta manera, la potencia compleja neta inyectada en cada nudo es un dato conocido y se define como:

$$S_i = S_i^{esp} = P_i^{esp} + jQ_i^{esp}, \quad \text{i =1,2,...m,} \tag{2.10}$$

Dicha potencia corresponde a la diferencia entre la potencia generada y la potencia demandada en cada nudo:

$$S_i = S_{gi} - S_{di} = U_i \cdot I_i^*, \quad \text{i =1,2,...m,} \tag{2.11}$$

Donde $I_i^*$ es la intensidad inyectada compleja conjugada del nudo i.

Utilizando coordenadas cartesianas para definir los elementos de la matriz de admitancias ($y_{ij} = G_i + jB_i$) la ecuación (2.11) se convierte en un sistema de ecuaciones no lineales de $m$ ecuaciones complejas:

$$P_i + jQ_i = U_i \sum_{j=1}^{m} U_j \cdot (G_{ij} - jB_{ij}), \quad \text{i =1,2,...m,} \tag{2.12}$$

Sin embargo, no es posible aplicar el método de Newton-Raphson a las ecuaciones anteriores, debido a que la presencia de variables complejas y variables conjugadas impide llevar a cabo derivadas en forma compleja. Por este motivo es necesario separar las $m$ ecuaciones en un set de $2m$ ecuaciones reales, expresando la tensión en cada nudo en coordenadas polares $U_i = V_i \angle 0$.

$$P_i = V_i \sum_{j=1}^{m} V_j \cdot (G_{ij} \cos\theta_{ij} + jB_{ij} \sin\theta_{ij}), \tag{2.13}$$

$$Q_i = V_i \sum_{j=1}^{m} V_j \cdot (G_{ij} \sin\theta_{ij} - jB_{ij} \cos\theta_{ij}), \tag{2.14}$$

$$\text{Donde } \theta_{ij} = \theta i - \theta j,$$

El hecho de que los ángulos de las tensiones se representen como diferencias de ángulos entre dos nudos conectados por una rama hace que no sea posible determinar todos los $\theta ij$. Por lo tanto, es necesario fijar el ángulo de la tensión de uno de los nudos del sistema como referencia. Por otro lado, no es posible conocer todos los flujos de potencia al no conocer las pérdidas en las líneas. Para resolver estos dos problemas, se fija la tensión compleja de uno de los nudos sustituyendo a la potencia activa y reactiva especificada. Ese nudo se conocera como nudo *slack* o nudo de referencia, se conocerá el valor del módulo de su tensión compleja y se tomará como origen de referencia de los ángulos de las tensiones del resto de los nudos (se suele tomar $\theta_{slack} = 0$ por simplicidad).

De esta manera, el problema del flujo de cargas queda formulado de la siguiente manera:

$$\begin{aligned} P_i &= V_i \sum_{j=1}^{m} V_j \cdot (G_{ij} \cos\theta_{ij} + jBij \sin\theta_{ij}), \\ Q_i &= V_i \sum_{j=1}^{m} V_j \cdot (G_{ij} \sin\theta_{ij} - jBij \cos\theta_{ij}), \end{aligned} \tag{2.15}$$

$$V_{slack} = V_{slack}^{esp} \angle 0,$$

$$\text{i=1,2...m,}$$

Estas ecuaciones son fuertemente no lineales (incluyen senos y cosenos) y por lo tanto la resolución de este problema no es trivial. Las variables de estado del problema del flujo de carga son los módulos $V_i$ y ángulos $\theta_i$ de las tensiones de todos los nudos. Estas serán las incógnitas del problema a resolver. En general, conocidas $P_i$ y $Q_i$ se pueden calcular $V_i$ y $\theta_i$.

### 2.3.3 Tipos de nudos

- **Nudos de consumo o nudos PQ:** donde se conoce el consumo de potencia activa y reactiva, siendo nula o conocida la generación de potencia

$$P_i^{esp} = P_{Gi}^{esp} - P_{Di}^{esp}, \tag{2.16}$$

$$Q_i^{esp} = Q_{Gi}^{esp} - Q_{Di}^{esp}, \tag{2.17}$$

- **Nudos de generación o nudos PV:** donde un generador u otro dispositivo controla y fija el módulo de la tensión y la potencia activa inyectada.

$$P_i^{esp} = P_{Gi}^{esp} - P_{Di}^{esp}, \tag{2.18}$$

$$V_i = V_i^{esp}, \tag{2.19}$$

Es importante señalar que la resolución del sistema de ecuaciones propuesto en esta sección no es sencilla. Debida a la no linealidad de gran parte de las ecuaciones, es necesario el uso de algoritmos iterativos como Newton-Raphson.

## 2.4 Método de NR para la resolución del flujo de cargas

### 2.4.1 Linealización del sistema de ecuaciones

Para resolver el sistema de ecuaciones no lineales presentadas anteriormente, se va a aplicar el método de Newton-Raphson. Para ello, el primer paso es utilizar el desarrollo en serie de Taylor para linealizar las ecuaciones de potencia activa y reactiva para, junto a las ecuaciones de la red, resolver el sistema de ecuaciones:

$$f(x) \cong f(x\lfloor 0) + \frac{\partial f(x)}{\partial x} \cdot \triangle X, \tag{2.20}$$

Donde $X$ hace referencia al vector de variables de estado cuyo valor desconocemos y $f(x) = S(V,\theta)$.

Al ser un problema multivariable la derivada de la función $f(x)$ respecto al vector de variables de estado se expresa en forma matricial.

Y el sistema de ecuaciones se representa en forma matricial de la siguiente forma:

$$\begin{bmatrix} H & N \\ M & L \end{bmatrix} \cdot \begin{bmatrix} \theta_i \\ \triangle V/V_i \end{bmatrix} = \begin{bmatrix} \triangle P_i \\ \triangle Q_i \end{bmatrix}, \quad \text{i =1,2...m,} \tag{2.21}$$

Se pueden distinguir tres términos dentro del sistema de ecuaciones:

- **Jacobiano:**

$$\frac{\partial f(x)}{\partial x} = \frac{\partial S(V,\theta)}{\partial (V,\theta)} = J = \begin{bmatrix} H & N \\ M & L \end{bmatrix}, \tag{2.22}$$

  Donde cada término representa la derivada parcial respecto a una de las variables de estado, utilizando las ecuaciones (2.13) y (2.14).

$$H_{ij} = \frac{\partial P_i}{\partial \theta_j}, \qquad N_{ij} = V_j \cdot \frac{\partial P_i}{\partial V_j}, \qquad M_{ij} = \frac{\partial Q_i}{\partial \theta_j}, \qquad L_{ij} = V_j \cdot \frac{\partial Q_i}{\partial V_j}. \tag{2.23}$$

- **Vector de Residuos:**

$$\begin{aligned} \triangle P_i &= P_i^{esp} - P_i^{cal}, \\ \triangle Q_i &= Q_i^{esp} - Q_i^{cal}, \end{aligned} \tag{2.24}$$

Donde $P_i^{cal}$ y $Q_i^{cal}$ son las potencias activa y reactiva de cada nudo obtenidas a partir de las ecuaciones (2.13) y (2.14). En los nudos en los que se especifica $P_i^{esp}$ y $Q_i^{esp}$ debe cumplirse lo siguiente:

$$\triangle P_i \sim 0 \quad \triangle Q_i \sim 0$$

- **Vector de variables de estado:** Son las incógnitas de nuestro problema: el módulo y el ángulo de los nudos del sistema. La tensión no se calcula directamente, sino que es preferible utilizar el término $\triangle V/V_i$ para trabajar con términos numéricamente más simétricos y mejorar el condicionamiento del vector. Esto aumenta la linealidad del sistema y mejora la convergencia del mismo.

$$\triangle X = \begin{bmatrix} \theta_i \\ \triangle V/V_i \end{bmatrix}, \quad \text{i = 1,2...m,} \tag{2.25}$$

### 2.4.2 Método iterativo

Una vez definido el sistema de ecuaciones matricial linealizado mediante el desarrollo en serie de Taylor (ecuación (2.21)), se define el algoritmo para resolver el problema. Como en cualquier proceso iterativo, el objetivo es alcanzar la convergencia. Para ello, es necesario fijar un *criterio de convergencia*, en nuestro caso se definirá como criterio la acotación del vector de residuos de potencia por debajo de un cierto umbral.

$$\begin{aligned} |\triangle P_i| < \varepsilon \\ |\triangle Q_i| < \varepsilon \end{aligned} \tag{2.26}$$

Donde $\varepsilon$ es la cota de error. Cuando los residuos sean inferiores a esta cota de error se considerará que se ha llegado a una solución suficientemente buena.

En líneas generales, el método iterativo viene definido de forma esquemática en la Figura 4.2:

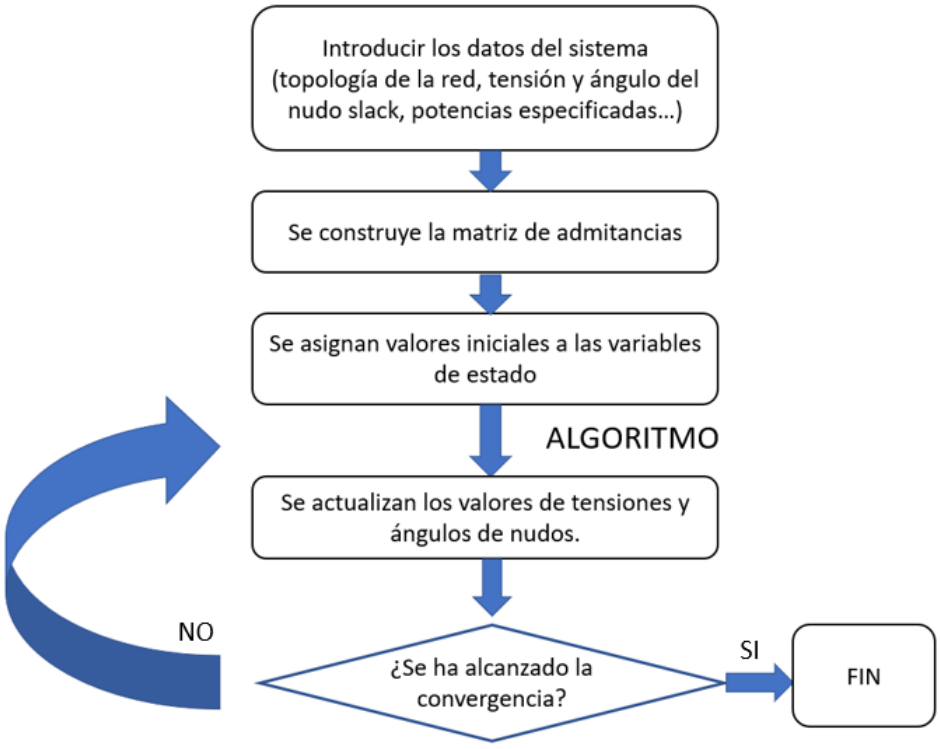


**Figura 2.3** Algoritmo de resolución del método iterativo.

## 2.5 Método de Newton-Raphson rectangular ampliado

Una adaptación del método de Newton-Raphson es el método rectangular aumentado[5]. Este método se diferencia respecto del NR clásico en el uso de coordenadas cartesianas en lugar de coordenadas polares y en la expansión del vector de variables de estado incluyendo las intensidades inyectadas en los nudos. De este modo, el vector de variables de estado pasa a ser:

$$\mathbf{X} = [\bar{V}_i, \bar{I}_i] = [e_i, f_i, I_{ai}, I_{bi}], \tag{2.27}$$

Donde V, I están expresadas en coordenadas cartesianas y se definen como:

$$\bar{U}_i = V_i \angle \theta_i = e_i + j f_i, \tag{2.28}$$
$$\bar{I}_i = \angle \alpha_i = I_{ai} + j I_{bi}, \tag{2.29}$$

Por otro lado:

$$y_{ij} = G_{ij} + B_{ij}, \tag{2.30}$$
$$S_i^{esp} = P_i^{esp} + j Q_i^{esp}, \tag{2.31}$$

Este método intenta aumentar la linealidad del sistema de ecuaciones para mejorar la convergencia del problema. En este caso, además de los residuos de potencia se añaden residuos de intensidad para completar el sistema de ecuaciones.

Del mismo modo que con el método NR clásico, este método también divide el problema en dos conjuntos de ecuaciones, uno lineal y otro no lineal. Las ecuaciones lineales nodales se definen de la misma manera a partir de la matriz de admitancias:

$$\begin{bmatrix} I_{ai} \\ I_{bi} \end{bmatrix} = \sum_{j=1}^{m} \begin{bmatrix} -B_{ij} & G_{ij} \\ G_{ij} & B_{ij} \end{bmatrix} \cdot \begin{bmatrix} f_i \\ e_i \end{bmatrix}, \tag{2.32}$$

Las restricciones no lineales parte de las ecuaciones (2.11), (2.13) y (2.14), pero se descomponen en parte real e imaginaria utilizando coordenadas cartesianas:

$$S_i = U_i \cdot I_i^*, \tag{2.33}$$
$$S_i = (ei + j f_i) \cdot (I_{ai} - j I_{bi}), \tag{2.34}$$
$$S_i = [(ei \cdot I_{ai}) + (f_i \cdot I_{bi})] + j[(fi \cdot I_{ai}) - (e_i \cdot I_{bi})]. \tag{2.35}$$

Expresando esta ecuación en forma matricial:

$$\begin{bmatrix} P_i^{esp} \\ Q_i^{esp} \end{bmatrix} = \begin{bmatrix} I_{bi} & I_{ai} \\ I_{ai} & -I_{bi} \end{bmatrix} \cdot \begin{bmatrix} f_i \\ e_i \end{bmatrix} = \begin{bmatrix} e_i & f_i \\ f_i & -e_i \end{bmatrix} \cdot \begin{bmatrix} I_{ai} \\ I_{bi} \end{bmatrix}, \tag{2.36}$$

Las ecuaciones lineal (2.32) y cuadrática (2.36) forman el sistema de ecuaciones a resolver. Una ventaja del método rectangular aumentando es que si se hubiesen usado coordenadas polares ambas ecuaciones hubiesen sido no lineales.

### 2.5.1 Linealización del sistema de ecuaciones

Igualmente que en la sección anterior, se va a aplicar el desarrollo en serie de Taylor para linealizar las ecuaciones no lineales del problema, utilizando la expresión de la ecuación (2.20)

$$f(x) \cong f(x\lfloor 0) + \frac{\partial f(x)}{\partial x} \cdot \triangle X, \tag{2.37}$$

Como el objetivo es hacer que los residuos se hagan cero, se iguala la función $f(x)$ a cero:

$$f(x\lfloor 0) = -\frac{\partial f(x)}{\partial x} \cdot \triangle X, \tag{2.38}$$

Igual que en la sección precedente, el vector $\mathbf{X}$ representa el vector de variables de estado.

### 2.5.2 Obtención de los residuos de potencia e intensidad

En este caso, cada conjunto de ecuaciones nos permite obtener un tipo de residuo. Las ecuaciones lineales nos permiten calcular los residuos de intensidad mientras que las ecuaciones cuadráticas los residuos de potencia.

En primer lugar, se define $f(x)$ a partir de (3.1) como:

$$f(x) = Y_{BUS} \cdot \bar{U}_i - \bar{I}_i, \tag{2.39}$$

Sustituyendo la ecuación (2.39) en la ecuación (2.38) y operando se obtiene:

$$Y_{BUS} \cdot \triangle U_i - \mathbf{I} \cdot \triangle I_i = -(Y_{BUS} \cdot U_i\lfloor_0 + I_i\lfloor_0, \tag{2.40}$$

$$Y_{BUS} \cdot \triangle U_i - \triangle I_i = \triangle \Lambda_i, \tag{2.41}$$

I es la matriz identidad y $\triangle \Lambda_i$ es el vector de residuos de intensidad, cuya expresión en coordenadas cartesianas es:

$$\triangle \Lambda_i = \begin{bmatrix} \triangle \alpha_i \\ \triangle \beta_i \end{bmatrix}, \tag{2.42}$$

Descomponiendo la ecuación (2.40) en forma compleja y expresando en forma matricial se obtiene la ecuación (2.43)

$$\sum_{j=1}^{m} \begin{bmatrix} -B_{ij} & G_{ij} \\ G_{ij} & B_{ij} \end{bmatrix} \cdot \begin{bmatrix} \triangle f_i \\ \triangle e_i \end{bmatrix} - \begin{bmatrix} \triangle I_{ai} \\ \triangle I_{bi} \end{bmatrix} = \begin{bmatrix} \triangle \alpha_i \\ \triangle \beta_i \end{bmatrix}, \tag{2.43}$$

Los residuos de intensidad se hacen nulos tras la primera iteración, debido al carácter lineal de los mismos.

Para obtener los residuos de potencia se procede de manera similar. Se parte igualmente del desarrollo en serie de Taylor, esta vez aplicado a las restricciones no lineales del problema:

$$f(x) = S_i^{esp} - \bar{U}_i \cdot \bar{I}_i, \tag{2.44}$$

De la misma manera que para la ecuación (2.39), pero sustituyendo la ecuación (2.45), se obtiene:

$$-I_i \cdot \triangle U_i - U_i \triangle I_i = -(S_i esp - U_i\lfloor_0 \cdot I_i\lfloor_0), \tag{2.45}$$
$$I_i \cdot \triangle U_i + U_i \triangle I_i = S_i esp - U_i\lfloor_0 \cdot I_i\lfloor_0 = \triangle S_i, \tag{2.46}$$

De esta manera se obtiene el vector de residuos de potencia $triangleS_i$, que expresado en coordenadas cartesianas es:

$$\triangle S_i = \begin{bmatrix} \triangle P_i \\ \triangle Q_i \end{bmatrix}, \tag{2.47}$$

Y por lo tanto, las restricciones cuadráticas del problema se linealizan, quedando el sistema de ecuaciones matricial siguiente:

$$\begin{bmatrix} I_{bi} & I_{ai} \\ I_{ai} & -I_{bi} \end{bmatrix} \begin{bmatrix} \triangle f_i \\ \triangle e_i \end{bmatrix} + \begin{bmatrix} e_i & f_i \\ f_i & -e_i \end{bmatrix} \cdot \begin{bmatrix} \triangle I_{ai} \\ \triangle I_{bi} \end{bmatrix} = \begin{bmatrix} \triangle P_i \\ \triangle Q_i \end{bmatrix}, \tag{2.48}$$

### 2.5.3 Sistema de ecuaciones a resolver

Una vez obtenidas las expresiones de los residuos de potencia e intensidad y linealizadas las ecuaciones iniciales, es posible unificar las ecuaciones (2.43) y (2.48) en un único conjunto de ecuaciones expresadas en forma matricial:

$$\begin{bmatrix} -B_{ij} & G_{ij} & -1 & 0 \\ G_{ij} & B_{ij} & 0 & -1 \\ I_{bi} & I_{ai} & e_i & f_i \\ I_{ai} & -I_{bi} & f_i & -e_i \end{bmatrix} \cdot \begin{bmatrix} \triangle f_i \\ \triangle e_i \\ \triangle I_{ai} \\ \triangle I_{bi} \end{bmatrix} = \begin{bmatrix} \triangle P_i \\ \triangle Q_i \\ \triangle \alpha_i \\ \triangle \beta_i \end{bmatrix} \tag{2.49}$$

Como su tensión y ángulo son conocidos, las filas y columnas de de las matrices relacionadas con el nudo *Slack* se eliminan. La matriz de la izquierda en la ecuación (2.49) es el Jacobiano y puede descomponerse en varias submatrices:

$$Y_{BUS} = \begin{bmatrix} -B_{ij} & G_{ij} \\ G_{ij} & B_{ij} \end{bmatrix}, \tag{2.50}$$

$$-I = \begin{bmatrix} -1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{2.51}$$

$$D_I = \begin{bmatrix} I_{bi} & I_{ai} \\ I_{ai} & -I_{bi} \end{bmatrix}, \tag{2.52}$$

$$D_V = \begin{bmatrix} e_i & f_i \\ f_i & -e_i \end{bmatrix}. \tag{2.53}$$

Por lo que el sistema de ecuaciones de (2.49) se puede representar como:

$$\begin{bmatrix} Y_{BUS} & -I \\ D_I & D_V \end{bmatrix} \cdot \begin{bmatrix} \triangle U \\ \triangle I \end{bmatrix} = \begin{bmatrix} \triangle \Lambda \\ \triangle S \end{bmatrix}, \tag{2.54}$$

### 2.5.4 Método iterativo de resolución. Caso 1: Únicamente nudos PQ

Una vez definido el sistema de ecuaciones, se desarrolla la resolución del mismo. En primer lugar se detalla la resolución para el caso general, solo con nudos PQ, para después definir en otra sección los pasos necesarios para incluir nudos PV.

Los datos necesarios para comenzar el problema son la tensión y ángulo del nudo de referencia, las potencias activa y reactiva de los nudos PQ y la matriz de admitancias de las líneas.

1. **Partir de un perfil inicial**: Existen varias técnicas para definir un perfil inicial. El objetivo de todas ellas es partir de un conjunto de soluciones lo más próxima posible a la solución final. De hecho, es especialmente importante porque la convergencia del problema depende de la solución inicial adoptada. En nuestro caso, partiremos de un perfil plano de tensiones, asignando la tensión del nudo *Slack* ( tensión conocida) a todos los nudos del sistema. Por otro lado, también se asigna la potencia activa y reactiva de los nudos.

$$S_i^{esp} = P_i^{esp} + jQ_i^{esp}, \tag{2.55}$$
$$U_i\lfloor_0 = e_i\lfloor_0 + jf_i\lfloor_0 = \bar{U}_{slack}, \tag{2.56}$$

Las intensidades inyectadas en los nudos se calculan a partir de los valores de tensión y potencia:

$$I_i\lfloor = (\frac{S_i^{esp}}{U_i\lfloor_0})^* = \frac{P_i^{esp} - jQ_i^{esp}}{U_i^*\lfloor_0}, \tag{2.57}$$

2. **Calcular las submatrices que forman el Jacobiano**:

   - $Y_{BUS}$ se calcula a partir de los términos de la matriz de admitancias($Y_{ii}$.$Y_{ij}$), ya definidos en las ecuaciones (2.8) y (2.9)
   - $D_V$ se construye con los términos real e imaginario de las tensiones de los nudos, a partir de los valores del perfil inicial.
   - $D_I$ se construye con los términos real e imaginario de las intensidades de los nudos, a partir de los valores del perfil inicial.

$$I_{ai}\lfloor_0 = \frac{P_i^{esp}}{e_i\lfloor_0}, \tag{2.58}$$
$$I_{bi}\lfloor_0 = \frac{Q_i^{esp}}{f_i\lfloor_0} \tag{2.59}$$

3. **Obtener los residuos de potencia e intensidad:** Los residuos de potencia e intensidad se obtienen a partir del sistema de ecuaciones matricial (2.49). Primero se utiliza la ecuación (2.48) para despejar $\triangle U$:

$$(YBUS + D_V^{-1} \cdot D_I \cdot \triangle U = \triangle \Lambda + D_V^{-1} \cdot \triangle S, \tag{2.60}$$

Una vez obtenido $\triangle U$ se utiliza la ecuación (2.43) para calcular $\triangle I$:

$$\triangle I = D_V^{-1} \cdot \triangle S - (D_V^{-1} \cdot D_I) \cdot \triangle U, \tag{2.61}$$

Estos vectores de residuos expresados en forma matricial son:

$$\triangle U_i = \begin{bmatrix} \triangle f_i \\ \triangle e_i \end{bmatrix}, \tag{2.62}$$

$$\triangle I_i = \begin{bmatrix} \triangle I_{ai} \\ \triangle I_{bi} \end{bmatrix}, \tag{2.63}$$

4. **Actualizar los valores de las variables de estado:** Calculados los residuos, solo queda actualizar los valores de tensiones e intensidades del sistema:

$$\begin{bmatrix} f_i \lfloor_1 = \triangle f_i + f_i \lfloor_0 \\ e_i \lfloor_1 = \triangle e_i + e_i \lfloor_0 \end{bmatrix} \begin{bmatrix} I_{ai} \lfloor_1 = \triangle I_{ai} + I_{ai} \lfloor_0 \\ I_{bi} \lfloor_1 = \triangle I_{bi} + I_{bi} \lfloor_0 \end{bmatrix} \tag{2.64}$$

5. **Comprobar que se cumple el criterio de convergencia establecido**.Si se cumple el criterio los resultados son válidos y el problema se considerará resuelto, si no se cumple este criterio se volverá al paso 1º. y se repiten los mismos pasos excepto calcular $Y_{BUS}$, ya que esta no varía.

### 2.5.5 Método iterativo de resolución. Caso 2: Inclusión de nudos PV

En la sección anterior se ha detallado el método iterativo para la resolución de un problema de flujo de cargas en un sistema formado exclusivamente por nudos PQ, donde los valores especificados en cada nudo son la potencia activa y la potencia reactiva. Para el caso con nudos PV, la resolución es globalmente equivalente al método anterior pero las restricciones no lineales se modifican para los nudos PV.

En esta sección se detallará el método para incluir también nudos PV, donde los valores especificados son el módulo de la tensión del nudo y la potencia activa.

$$(V_i^{esp})^2 = e_i^2 + f_i^2, \tag{2.65}$$

Para incluir esta nueva restricción se modifica la ecuación (2.11), donde se sustituye:

$$Q_i^{esp} = I_{ai} \cdot f_i - I_{bi} \cdot e_i, \tag{2.66}$$

por la restricción de tensión especificada. De esta manera, las filas asociadas a los nudos PV del sistema dentro de la ecuación (2.36) pasan a ser:

$$\begin{bmatrix} P_i^{esp} \\ (V_i^{esp})^2 \end{bmatrix} = \begin{bmatrix} I_{bi} & I_{ai} \\ f_i & e_i \end{bmatrix} \cdot \begin{bmatrix} f_i \\ e_i \end{bmatrix}, \tag{2.67}$$

Por lo tanto, las filas asociadas a los nudos PV dentro de la ecuación matricial de residuos de potencia se modifican:

$$\begin{bmatrix} \triangle P_i^{esp} \\ \triangle (V_i^{esp})^2 \end{bmatrix} = \begin{bmatrix} P_i^{esp} \\ (V_i^{esp})^2 \end{bmatrix} - \begin{bmatrix} I_{bi} & I_{ai} \\ f_i & e_i \end{bmatrix} \cdot \begin{bmatrix} f_i \\ e_i \end{bmatrix}, \tag{2.68}$$

Y la ecuación linealizada es ahora:

$$\begin{bmatrix} I_{bi} & I_{ai} \\ 2f_i & 2e_i \end{bmatrix} \begin{bmatrix} \triangle f_i \\ \triangle e_i \end{bmatrix} + \begin{bmatrix} e_i & f_i \\ 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \triangle I_{ai} \\ \triangle I_{bi} \end{bmatrix} = \begin{bmatrix} \triangle P_i \\ \triangle V_i^2 \end{bmatrix}, \tag{2.69}$$

La modificación de esta ecuación hace que las submatrices de tensiones e intensidades del Jacobiano sean diferentes en aquellas filas asociadas a nudos PV:

$$D_I = \begin{bmatrix} I_{bi} & I_{ai} \\ 2f_i & 2e_i \end{bmatrix}, \tag{2.70}$$

$$D_V = \begin{bmatrix} e_i & f_i \\ 0 & 0 \end{bmatrix}, \tag{2.71}$$

La modificación de estas submatrices puede parecer un aspecto sin importancia, pero modifica enormemente la resolución por el método iterativo propuesto en la sección anterior. En ese método, para calcular los $\triangle U_i$ para cada nodo, era necesario invertir la submatriz $D_V$, siguiendo la ecuación (2.62). Sin embargo, el determinante de la matriz de la ecuación (2.71) es igual a cero y por lo tanto no es posible calcular su inversa.

Ante este inconveniente, la solución es calcular de forma directa el sistema de ecuaciones (2.73) mediante alguno de los métodos de sustitución-eliminación de sistemas de ecuaciones matricial de la forma $A \cdot x = b$.

$$\begin{bmatrix} Y_{BUS} & -I \\ D_I & D_V \end{bmatrix} \cdot \begin{bmatrix} \triangle U \\ \triangle I \end{bmatrix} = \begin{bmatrix} \triangle \Lambda \\ \triangle S \end{bmatrix}, \tag{2.72}$$

Donde las ecuaciones de las filas asociadas a nudos PV serán de la forma:

$$\begin{bmatrix} -B_{ij} & G_{ij} & -1 & 0 \\ G_{ij} & B_{ij} & 0 & -1 \\ I_{bi} & I_{ai} & e_i & f_i \\ 2e_i & 2f_i & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} \triangle f_i \\ \triangle e_i \\ \triangle I_{ai} \\ \triangle I_{bi} \end{bmatrix} = \begin{bmatrix} \triangle P_i \\ \triangle V_i^2 \\ \triangle \alpha_i \\ \triangle \beta_i \end{bmatrix} \tag{2.73}$$

Este método es menos eficiente que el de la sección anterior y requiere un número mayor de iteraciones y gasto computacional. Sin embargo, la resolución propuesta en el primer método no es factible para sistemas con nudos PV.

# 3 Método Rectangular Aumentado para redes con cuatro conductores

En este capítulo se desarrollará la formulación del método rectangular aumentado aplicado a redes trifásicas desequilibradas de cuatro hilos.

Como se comentó durante el capítulo de introducción de este documento, este trabajo tiene como objetivo programar un algoritmo que resuelva el problema de flujo de cargas en redes de distribución. Esta parte del sistema eléctrico presenta normalmente cargas desequilibradas, especialmente en baja tensión y por lo tanto no es posible aplicar un modelo monofásico equivalente, si no que es necesario estudiar todas las fases (a,b,c) y el neutro de forma independiente.

Para ello, en este capítulo se partirá de la formulación detallada en el capítulo anterior y se desarrollará para poder aplicar el método rectangular aumentado a redes desequilibradas. Para ello se partirá de la formulación presentada en [5]. Además, se incluirá una sección para explicar el modelado de transformadores de cabecera, parte fundamental en redes de distribución.

## 3.1 Ampliación del método rectangular aumentado para redes trifásicas de cuatro conductores

### 3.1.1 Ecuaciones ampliadas de nudos

Las ecuaciones lineales parten de la misma formulación que en el capítulo anterior.

$$I_i = y_{ii} \cdot U_i + \sum_{j \neq i}^{m} y_{ij} \cdot U_j, \tag{3.1}$$

Hay que adecuar el modelo lineal dado por la relación entre tensiones e intensidades en el sistema. Se tomará como referencia la ecuación (3.1) y se ampliará para incluir las cuatro fases. Si se desarrolla matricialmente la expresión resultante es la ecuación (3.2):

$$\begin{bmatrix} I_i^a \\ I_i^b \\ I_i^c \\ I_i^n \end{bmatrix} = \begin{bmatrix} Y_{ii}^{aa} & Y_{ii}^{ab} & Y_{ii}^{ac} & Y_{ii}^{an} \\ Y_{ii}^{ba} & Y_{ii}^{bb} & Y_{ii}^{bc} & Y_{ii}^{bn} \\ Y_{ii}^{ca} & Y_{ii}^{cb} & Y_{ii}^{cc} & Y_{ii}^{cn} \\ Y_{ii}^{na} & Y_{ii}^{nb} & Y_{ii}^{nc} & Y_{ii}^{nn} \end{bmatrix} \cdot \begin{bmatrix} U_i^a \\ U_i^b \\ U_i^c \\ U_i^n \end{bmatrix} + \sum_{j \neq i}^{m} \begin{bmatrix} Y_{ij}^{aa} & Y_{ij}^{ab} & Y_{ij}^{ac} & Y_{ij}^{an} \\ Y_{ij}^{ba} & Y_{ij}^{bb} & Y_{ij}^{bc} & Y_{ij}^{bn} \\ Y_{ij}^{ca} & Y_{ij}^{cb} & Y_{ij}^{cc} & Y_{ij}^{cn} \\ Y_{ij}^{na} & Y_{ij}^{nb} & Y_{ij}^{nc} & Y_{ij}^{nn} \end{bmatrix} \tag{3.2}$$

Es importante destacar que todos estos desarrollos y expresiones siguen utilizando coordenadas cartesianas, por lo que cada elemento del vector de tensiones y el vector de intensidades tendrá dos componentes: uno para la parte real Y otro para la parte imaginaria, utilizando la notación de las expresiones (3.4) y (3.3)

$$I_i^d = \begin{bmatrix} I_{ai}^d \\ I_{bi}^d \end{bmatrix}, \quad \text{d = a,b,c,n,} \tag{3.3}$$

$$U_i^d = \begin{bmatrix} f_i^d \\ e_i^d \end{bmatrix}, \quad \text{d = a,b,c,n,} \tag{3.4}$$

Por lo tanto, los vectores de tensión e intensidad serán de dimensiones [8*m x 1], donde m es el número de nudos de la red.

$$I_i = \begin{bmatrix} I_{ai}^a \\ I_{bi}^a \\ I_{ai}^b \\ I_{bi}^b \\ I_{ai}^c \\ I_{bi}^c \\ I_{ai}^n \\ I_{bi}^n \end{bmatrix}, \qquad U_i = \begin{bmatrix} f_i^a \\ e_i^a \\ f_i^b \\ e_i^b \\ f_i^c \\ e_i^c \\ f_i^n \\ e_i^n \end{bmatrix} \tag{3.5}$$

Por otro lado, los componentes de la matriz de admitancias pasan a ser matrices de dimensión [4x4], Ya que son la relación entre dos vectores [2x1]. Estas matrices se representan en las expresiones (3.6) y (3.7):

$$Y_{ii}^{dt} = \begin{bmatrix} -B_{ii}^{dt} & G_{ii}^{dt} \\ G_{ii}^{dt} & B_{ii}^{dt} \end{bmatrix}, \quad \text{d,t = a,b,c,n,} \tag{3.6}$$

$$Y_{ij}^{dt} = \begin{bmatrix} -B_{ij}^{dt} & G_{ij}^{dt} \\ G_{ij}^{dt} & B_{ij}^{dt} \end{bmatrix}, \quad \text{d,t = a,b,c,n,} \tag{3.7}$$

Entre estas submatrices cabe destacar la componente $Y_{ii}^{nn}$. En la configuración de puesta a tierra más utilizada en redes de distribución la conexión a tierra se realiza mediante el neutro, por lo que habría que añadir un término correspondiente a la conductancia de puesta a tierra $R_g$:

$$Y_{ii}^{nn} = \begin{bmatrix} -B_{ii}^{nn} & G_{ii}^{nn} + \frac{1}{R_g} \\ G_{ii}^{nn} + \frac{1}{R_g} & B_{ii}^{nn} \end{bmatrix}, \tag{3.8}$$

A partir de estas expresiones podemos extraer la matriz de admitancia de nudos $Y_{BUS}$. Si generalizamos para un conjunto de nudos *m* perteneciente a una red cualquiera, el sistema de ecuaciones matriciales que representa las ecuaciones de nudos de forma compacta es:

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ \vdots \\ I_m \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} & Y_{13} & \cdots & Y_{1m} \\ Y_{21} & Y_{22} & Y_{23} & \cdots & Y_{2m} \\ Y_{31} & Y_{32} & Y_{33} & \cdots & Y_{3m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ Y_{m1} & Y_{m2} & Y_{m3} & \cdots & Y_{mm} \end{bmatrix} \cdot \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ \vdots \\ U_m \end{bmatrix} \tag{3.9}$$

La matriz de admitancias relación tensiones con intensidades, siendo simétrica y de dimensión [8*m x 8*m]. Los términos de la matriz de admitancias son de la siguiente forma:

- **Elementos de la diagonal principal:**

$$Y_{ii} = \begin{bmatrix} Y_{ii}^{aa} & Y_{ii}^{ab} & Y_{ii}^{ac} & Y_{ii}^{an} \\ Y_{ii}^{ba} & Y_{ii}^{bb} & Y_{ii}^{bc} & Y_{ii}^{bn} \\ Y_{ii}^{ca} & Y_{ii}^{cb} & Y_{ii}^{cc} & Y_{ii}^{cn} \\ Y_{ii}^{na} & Y_{ii}^{nb} & Y_{ii}^{nc} & Y_{ii}^{nn} \end{bmatrix} \tag{3.10}$$

- **Elementos fuera de la diagonal principal:**

$$Y_{ij} = \begin{bmatrix} Y_{ij}^{aa} & Y_{ij}^{ab} & Y_{ij}^{ac} & Y_{ij}^{an} \\ Y_{ij}^{ba} & Y_{ij}^{bb} & Y_{ij}^{bc} & Y_{ij}^{bn} \\ Y_{ij}^{ca} & Y_{ij}^{cb} & Y_{ij}^{cc} & Y_{ij}^{cn} \\ Y_{ij}^{na} & Y_{ij}^{nb} & Y_{ij}^{nc} & Y_{ij}^{nn} \end{bmatrix} conj \neq i \tag{3.11}$$

- **Resistencias de puesta a tierra:** Este término solo afectan a los componentes de la diagonal principal asociadas a nudos donde exista resistencia de tierra $R_{gi}$, más concretamente al término $Y_{ii}^{nn}$.

$$Y_{ii} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{R_{gi}} \end{bmatrix} \tag{3.12}$$

Este primer desarrollo Ya muestra que la ampliación del problema a cuatro fases aumenta considerablemente el tamaño de las ecuaciones.

### 3.1.2 Tipos de puesta a tierra

Como se ha comentado en las ecuaciones (3.8) y (3.12), la puesta a tierra de los nudos se modela mediante la resistencia de puesta a tierra del nudo $R_{gi}$. Este término se coloca en la matriz $Y_{nn}$ de cada nudo y se modela de manera diferente según el tipo de puesta a tierra. De esta manera, podemos distinguir tres casos:

- Puesta a tierra finita: Cuando el valor de $R_{gi}$ toma valores finitos mayores que cero. Es el caso más simple y el más habitual. La puesta a tierra se define simplemente añadiendo el término definido en la ecuación (3.8).

- Puesta a tierra rígida: Cuando $R_{gi} = 0$. En este caso no es posible añadir el término porque provoca una indeterminación en la submatriz $Y_{nn}$. Lo que se hace en estos casos es obviar el término de puesta a tierra y la matriz $Y_{nn}$ es la de la expresión (3.13):

$$Y_{ii}^{nn} = \begin{bmatrix} -B_{ii}^{nn} & G_{ii}^{nn} \\ G_{ii}^{nn} & B_{ii}^{nn} \end{bmatrix}, \tag{3.13}$$

- Nudo sin puesta a tierra: Cuando $R_{gi} = \infty$. En este caso el término $\frac{1}{R_g}$ se anula y nos queda una expresión equivalente a la del caso anterior.

### 3.1.3 Restricciones no lineales ampliadas

Igual que en el capítulo anterior, para el método rectangular a cuatro hilos existen dos conjuntos de ecuaciones. Además de las ecuaciones de nudos hay que tener en cuenta las ecuaciones cuadráticas del modelo, ligadas a las cargas o generación existentes.

La Figura 3.1 representa un ejemplo de nudo dentro de la red:

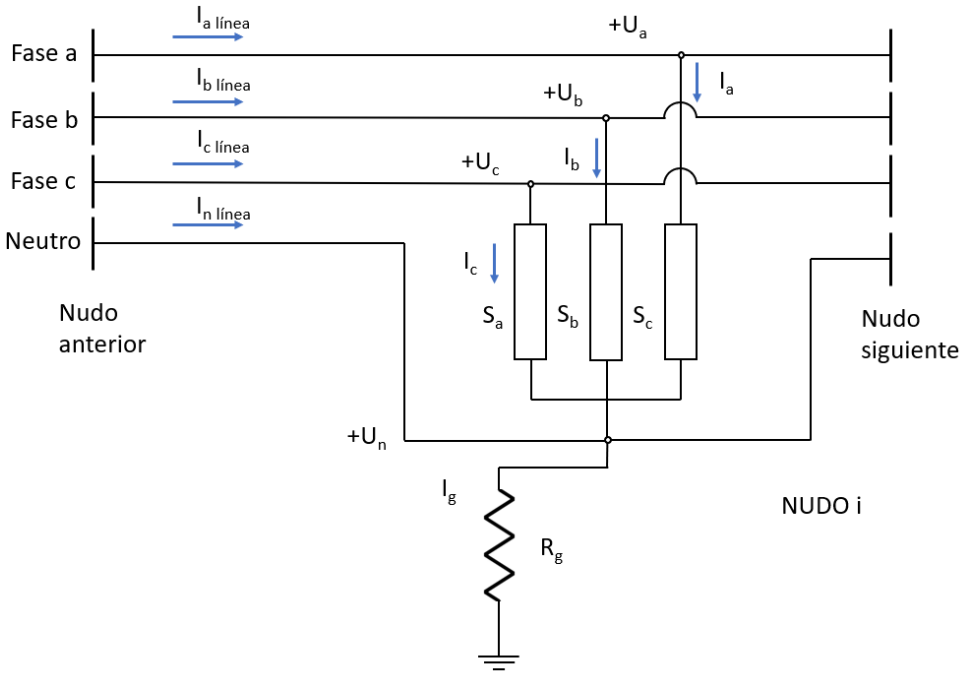


**Figura 3.1** Red trifásica desequilibrada a cuatro hilos. Conexión estrella con neutro puesto a tierra.

Partiendo del desarrollo anterior y de la ecuación (2.11) y conocida la potencia aparente especificada en cada nudo podemos despejar la intensidad inyectada en cada nudo $i$ en cada fase $ph$:

$$I_i = \left( \frac{S_i}{U_i^{ph} - U_i^n} \right)^*, \quad \text{ph = a,b,c,} \tag{3.14}$$

$$I_i^a = \left( \frac{S_i^{esp,a}}{U_i^a - U_i^n} \right)^* = \frac{P_i^{esp,a} - jQ_i^{esp,a}}{(e_i^a - e_i^n) - j(f_i^a - f_i^n)} \tag{3.15}$$

$$I_i^b = \left( \frac{S_i^{esp,b}}{U_i^b - U_i^n} \right)^* = \frac{P_i^{esp,b} - jQ_i^{esp,b}}{(e_i^b - e_i^n) - j(f_i^b - f_i^n)} \tag{3.16}$$

$$I_i^c = \left( \frac{S_i^{esp,c}}{U_i^c - U_i^n} \right)^* = \frac{P_i^{esp,c} - jQ_i^{esp,c}}{(e_i^c - e_i^n) - j(f_i^c - f_i^n)} \tag{3.17}$$

El valor de la intensidad del neutro se puede calcular a partir de la restricción de que la suma de intensidades que circulan a través de las cuatro fases en de un nudo debe ser nula:

$$I_i^n = -(I_i^a + I_i^b + I_i^c), \tag{3.18}$$

Si se agrupan estas expresiones de manera matricial se obtiene una ecuación equivalente a (2.11):

$$\begin{bmatrix} P_i^{esp,a} \\ Q_i^{esp,a} \\ P_i^{esp,b} \\ Q_i^{esp,b} \\ P_i^{esp,c} \\ Q_i^{esp,c} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} (e_i^a - e_i^n) & (f_i^a - f_i^n) & 0 & 0 & 0 & 0 & 0 & 0 \\ (f_i^a - f_i^n) & -(e_i^a - e_i^n) & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & (e_i^a - e_i^n) & (f_i^b - f_i^n) & 0 & 0 & 0 & 0 \\ 0 & 0 & (f_i^a - f_i^n) & -(e_i^b - e_i^n) & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & (e_i^c - e_i^n) & (f_i^c - f_i^n) & 0 & 0 \\ 0 & 0 & 0 & 0 & (f_i^c - f_i^n) & -(e_i^c - e_i^n) & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} I_{ai}^a \\ I_{bi}^a \\ I_{ai}^b \\ I_{bi}^b \\ I_{ai}^c \\ I_{bi}^c \\ I_{ai}^n \\ I_{bi}^n \end{bmatrix} \tag{3.19}$$

Esta ecuación tiene tres términos fundamentales:

1. $S_i$ es el vector de potencias especificadas, formado por la potencia activa Y reactiva en cada fase para cada nudo i Y de dimensión [8 x 1]. Este vector no varía a lo largo del proceso iterativo, al estar formado por las potencias especificadas del problema, que son constantes.
2. $I_i$ es el vector de intensidades complejas por fase, formado por la parte real e imaginaria de las intensidades de cada fase de un nudo i. Su dimensión es [8 x 1].
3. $D_V$ es la matriz de tensiones complejas fase-neutro. Relaciona las potencias con las intensidades en cada fase de un nudo i Y su dimensión es [8x8].

De esta manera, para cada nudo i, se obtiene la siguiente expresión compacta.

$$S_i = D_V \cdot I_i, \tag{3.20}$$

Conocida esta expresión para un nudo i, esta puede generalizarse para una red cualquiera de m nudos, obteniéndose la siguiente expresión:

$$\begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ \vdots \\ S_m \end{bmatrix} = \begin{bmatrix} D_{V1} & 0 & 0 & \cdots & 0 \\ 0 & D_{V2} & 0 & \cdots & 0 \\ 0 & 0 & D_{V3} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & D_{Vm} \end{bmatrix} \cdot \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ \vdots \\ I_m \end{bmatrix} \tag{3.21}$$

En esta expresión podemos observar que la matriz $D_V$ solo tiene elementos en su diagonal principal, siendo el resto de elementos nulos.

### 3.1.4 Obtención de los residuos para redes de cuatro hilos

Como se ha comentado, este capítulo sigue la estructura del capítulo anterior, por lo que el siguiente paso corresponde a la linealización del modelo y la obtención de los residuos de potencia e intensidad. Para ello se utiliza nuevamente el desarrollo en serie de Taylor para linealizar el modelo, obteniendo las expresiones (3.22) y (3.27):

- **Residuos de intensidad:**

$$\triangle\Lambda_i = Y_{ii}\cdot\triangle U_i + \sum_{j\neq i}^{m} Y_{ij}\cdot U_j - \triangle I_i, \tag{3.22}$$

Que expresado en forma matricial ampliado a redes de 4 hilos con tres fases Y neutro:

$$\begin{bmatrix}\triangle\Lambda_1\\ \triangle\Lambda_2\\ \triangle\Lambda_3\\ \vdots\\ \triangle\Lambda_m\end{bmatrix} = \begin{bmatrix}\triangle I_1\\ \triangle I_2\\ \triangle I_3\\ \vdots\\ \triangle I_m\end{bmatrix} - \begin{bmatrix}Y_{11} & Y_{12} & Y_{13} & \cdots & Y_{1m}\\ Y_{21} & Y_{22} & Y_{23} & \cdots & Y_{2m}\\ Y_{31} & Y_{32} & Y_{33} & \cdots & Y_{3m}\\ \vdots & \vdots & \ddots & \vdots & \\ Y_{m1} & Y_{m2} & Y_{m3} & \cdots & Y_{mm}\end{bmatrix}\cdot\begin{bmatrix}\triangle U_1\\ \triangle U_2\\ \triangle U_3\\ \vdots\\ \triangle U_m\end{bmatrix} \tag{3.23}$$

Donde:

- $\triangle\Lambda_i$ es el vector de residuos de intensidades, de dimensión [8*m x 1], donde m representa el número de nudos. Para cada nudo i su expresión matricial es la siguiente:

$$\triangle\Lambda_i = \begin{bmatrix}\alpha_i^a\\ \beta_i^a\\ \alpha_i^b\\ \beta_i^b\\ \alpha_i^c\\ \beta_i^c\\ \alpha_i^n\\ \beta_i^n\end{bmatrix} \tag{3.24}$$

- La matriz de admitancias(matriz $Y_{BUS}$) mantiene la misma formulación que en la ecuación (3.9).

- $\triangle U_i$, $\triangle I_i$ son los vectores de residuos de tensión e intensidades ,respectivamente, para el proceso iterativo.

$$\triangle I_i = \begin{bmatrix}\triangle I_{ai}^a\\ \triangle I_{bi}^a\\ \triangle I_{ai}^b\\ \triangle I_{bi}^b\\ \triangle I_{ai}^c\\ \triangle I_{bi}^c\\ \triangle I_{ai}^n\\ \triangle I_{bi}^n\end{bmatrix}, \quad \triangle U_i = \begin{bmatrix}\triangle f_i^a\\ \triangle e_i^a\\ \triangle f_i^b\\ \triangle e_i^b\\ \triangle f_i^c\\ \triangle e_i^c\\ \triangle f_i^n\\ \triangle e_i^n\end{bmatrix} \tag{3.25}$$

La ecuación (3.23) representa el modelo linealizado de la ecuaciones de nudos. Para obtener los residuos de intensidades a partir de la tensiones e intensidades complejas ampliadas a cuatro hilos se

utiliza la siguiente expresión:

$$\begin{bmatrix} \triangle\Lambda_1 \\ \triangle\Lambda_2 \\ \triangle\Lambda_3 \\ \vdots \\ \triangle\Lambda_m \end{bmatrix} = \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ \vdots \\ I_m \end{bmatrix} - \begin{bmatrix} Y_{11} & Y_{12} & Y_{13} & \cdots & Y_{1m} \\ Y_{21} & Y_{22} & Y_{23} & \cdots & Y_{2m} \\ Y_{31} & Y_{32} & Y_{33} & \cdots & Y_{3m} \\ \vdots & \vdots & \ddots & \vdots & \\ Y_{m1} & Y_{m2} & Y_{m3} & \cdots & Y_{mm} \end{bmatrix} \cdot \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ \vdots \\ U_m \end{bmatrix} \tag{3.26}$$

Donde:

- $I_i$ es el vector de intensidades inyectadas en el nudo i (Ecuación 3.5).
- $U_i$ es el vector de tensiones de cada nudo i (Ecuación (3.5)).
- La matriz de admitancias(matriz $Y_{BUS}$) mantiene la misma formulación que en la ecuación 3.9

- **Residuos de potencia:** En el capítulo anterior se obtenía la siguiente expresión (ecuación (2.45)) a partir de la linealización de las ecuaciones cuadráticas.

$$U_i \triangle I_i + I_i \cdot \triangle U_i = \triangle S_i = S_i - D_V \cdot I_i, \tag{3.27}$$

Para obtener la expresión equivalente para una red de cuatro hilos, solo es necesario ampliar los vectores $\triangle I_i, \triangle U_i, \triangle S_i$ para incluir las cuatro fases en cada nudo.

Por otro lado,

$$I_i = \frac{\partial f(x)}{\partial U} = D_{Ii}, \qquad U_i = \frac{\partial f(x)}{\partial I} = D_{Vi}$$

Si expresamos en forma matricial la ecuación 3.27 obtenemos la siguiente ecuación matricial:

$$\begin{bmatrix} \triangle S_1 \\ \triangle S_2 \\ \triangle S_3 \\ \vdots \\ \triangle S_m \end{bmatrix} = \begin{bmatrix} D_{V1} & 0 & 0 & \cdots & 0 \\ 0 & D_{V2} & 0 & \cdots & 0 \\ 0 & 0 & D_{V3} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & D_{Vm} \end{bmatrix} \cdot \begin{bmatrix} \triangle I_1 \\ \triangle I_2 \\ \triangle I_3 \\ \vdots \\ \triangle I_m \end{bmatrix} + \begin{bmatrix} D_{I1} & 0 & 0 & \cdots & 0 \\ 0 & D_{I2} & 0 & \cdots & 0 \\ 0 & 0 & D_{I3} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & D_{Im} \end{bmatrix} \begin{bmatrix} \triangle U_1 \\ \triangle U_2 \\ \triangle U_3 \\ \vdots \\ \triangle U_m \end{bmatrix} \tag{3.28}$$

Donde:

$$\triangle S_i = \begin{bmatrix} \triangle P_i^a \\ \triangle Q_i^a \\ \triangle P_i^b \\ \triangle Q_i^b \\ \triangle P_i^c \\ \triangle Q_i^c \\ \triangle P_i^n \\ \triangle Q_i^n \end{bmatrix} \tag{3.29}$$

En la ecuación (3.28) aparece una matriz que no se ha definido todavía, $D_I$, matriz diagonal de intensidades formada a partir de las ecuaciones de nudos y cuYos términos son de la forma:

$$D_I = \begin{bmatrix} D_{I1} & 0 & 0 & \cdots & 0 \\ 0 & D_{I2} & 0 & \cdots & 0 \\ 0 & 0 & D_{I3} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & D_{Im} \end{bmatrix} \tag{3.30}$$

La matriz de intensidades es una matriz diagonal al igual que $D_V$ con un término por cada nudo de la red.

$$D_{Ii} = \begin{bmatrix} I^a_{bi} & I^a_{ai} & 0 & 0 & 0 & 0 & -I^a_{bi} & -I^a_{ai} \\ I^a_{ai} & -I^a_{bi} & 0 & 0 & 0 & 0 & -I^a_{ai} & I^a_{bi} \\ 0 & 0 & I^b_{bi} & I^b_{ai} & 0 & 0 & -I^b_{bi} & -I^b_{ai} \\ 0 & 0 & I^b_{ai} & -I^b_{bi} & 0 & 0 & -I^b_{ai} & I^b_{bi} \\ 0 & 0 & 0 & 0 & I^c_{bi} & I^c_{ai} & -I^c_{bi} & -I^c_{ai} \\ 0 & 0 & 0 & 0 & I^c_{ai} & -I^c_{bi} & -I^c_{ai} & I^c_{bi} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.31}$$

Los componentes de $D_I$ están formadas por las intensidades complejas de cada fase expresadas en submatrices [4 x 4]. Las últimas dos filas y dos columnas corresponden a la intensidad compleja del neutro Y se definen a partir de la ecuación (3.41).

Una vez definido el modelo linealizado, volviendo a la ecuación (3.27), los residuos de potencia se definen como:

$$\begin{bmatrix} \triangle S_1 \\ \triangle S_2 \\ \triangle S_3 \\ \vdots \\ \triangle S_m \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \\ \vdots \\ S_m \end{bmatrix} - \begin{bmatrix} D_{V1} & 0 & 0 & \cdots & 0 \\ 0 & D_{V2} & 0 & \cdots & 0 \\ 0 & 0 & D_{V3} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & D_{Vm} \end{bmatrix} \cdot \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ \vdots \\ I_m \end{bmatrix} \tag{3.32}$$

### 3.1.5 Sistema de ecuaciones a resolver y método iterativo para redes con cuatro conductores y únicamente nudos PQ

Las ecuaciones obtenidas en la sección anterior pueden expresarse de forma compacta. Si se expresan las ecuaciones (3.28) y (3.23) de forma conjunta se obtiene:

$$\begin{bmatrix} Y_{BUS} & -I \\ D_I & D_V \end{bmatrix} \cdot \begin{bmatrix} \triangle U \\ \triangle I \end{bmatrix} = \begin{bmatrix} \triangle \Lambda \\ \triangle S \end{bmatrix}, \tag{3.33}$$

Que desarrollando sus términos es:

$$\left[\begin{array}{ccccc|ccccc} Y_{11} & Y_{12} & Y_{13} & \cdots & Y_{1m} & -1 & 0 & 0 & \cdots & 0 \\ Y_{21} & Y_{22} & Y_{23} & \cdots & Y_{2m} & 0 & -1 & 0 & \cdots & 0 \\ Y_{31} & Y_{32} & Y_{33} & \cdots & Y_{3m} & 0 & 0 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ Y_{m1} & Y_{m2} & Y_{m3} & \cdots & Y_{mm} & 0 & 0 & 0 & \cdots & -1 \\ \hline D_{I1} & 0 & 0 & \cdots & 0 & D_{V1} & 0 & 0 & \cdots & 0 \\ 0 & D_{I2} & 0 & \cdots & 0 & 0 & D_{V2} & 0 & \cdots & 0 \\ 0 & 0 & D_{I3} & \cdots & 0 & 0 & 0 & D_{V3} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & D_{Im} & 0 & 0 & 0 & \cdots & D_{Vm} \end{array}\right] \cdot \left[\begin{array}{c} \triangle U_1 \\ \triangle U_2 \\ \triangle U_3 \\ \vdots \\ \triangle U_m \\ \hline \triangle I_1 \\ \triangle I_2 \\ \triangle I_3 \\ \vdots \\ \triangle I_m \end{array}\right] = \left[\begin{array}{c} \triangle \Lambda_1 \\ \triangle \Lambda_2 \\ \triangle \Lambda_3 \\ \vdots \\ \triangle \Lambda_m \\ \hline \triangle S_1 \\ \triangle S_2 \\ \triangle S_3 \\ \vdots \\ \triangle S_m \end{array}\right] \tag{3.34}$$

Al igual que en la sección (2.5.4) una vez definido el sistema de ecuaciones se desarrolla la resolución del mismo. En esta primera sección se detalla la resolución para el caso general: solo nudos PQ. en las secciones posteriores en este capítulo se definen también los pasos necesarios para incluir nudos PV y transformadores en cabecera.

Los datos necesarios para comenzar el problema son la tensión y ángulo del nudo de referencia, las potencias activa y reactiva de los nudos PQ y la matriz de admitancias de las líneas.

1. **Partir de un perfil inicial**: Al igual que en la sección (2.5.4), partiremos de un perfil plano de tensiones, asignando la tensión de cada fase del nudo *Slack* (tensión conocida) a todos los nudos del sistema. Por otro lado, también se asigna la potencia activa y reactiva de los nudos en cada fase.

$$S_i^{esp,d} = P_i^{esp,d} + jQ_i^{esp,d}, \quad \text{d = a,b,c,n,} \tag{3.35}$$

$$U_i^d \lfloor_0 = e_i^d \lfloor_0 + jf_i^d \lfloor_0 = \bar{U}_{slack}^d, \quad \text{d = a,b,c,n,} \tag{3.36}$$

Las intensidades inyectadas en los nudos se calculan a partir de los valores de tensión Y potencia:

$$I_i^p h \lfloor_0 = \left(\frac{S_i^{esp,ph} \lfloor_0}{(U_i^{ph} - U_i^n) \lfloor_0}\right)^* \quad \text{ph = a,b,c,} \tag{3.37}$$

$$I_i^a \lfloor_0 = \left(\frac{S_i^{esp,a}}{(U_i^a - U_i^n) \lfloor_0}\right)^* = \frac{P_i^{esp,a} - jQ_i^{esp,a},}{(e_i^a - e_i^n) \lfloor_0 - j(f_i^a - f_i^n) \lfloor_0}, \tag{3.38}$$

$$I_i^b \lfloor_0 = \left(\frac{S_i^{esp,b}}{(U_i^b - U_i^n) \lfloor_0}\right)^* = \frac{P_i^{esp,b} - jQ_i^{esp,b}}{(e_i^b - e_i^n) \lfloor_0 - j(f_i^b - f_i^n) \lfloor_0} \tag{3.39}$$

$$I_i^c \lfloor_0 = \left(\frac{S_i^{esp,c}}{(U_i^c - U_i^n) \lfloor_0}\right)^* = \frac{P_i^{esp,c} - jQ_i^{esp,c}}{(e_i^c - e_i^n) \lfloor_0 - j(f_i^c - f_i^n) \lfloor_0} \tag{3.40}$$

El valor de la intensidad del neutro se puede calcular a partir de la restricción de que la suma de intensidades que circulan a través de las cuatro fases en de un nudo debe ser nula:

$$I_i^n\lfloor_0 = -(I_i^a\lfloor_0 + I_i^b\lfloor_0 + I_i^c\lfloor_0), \tag{3.41}$$

2. **Calcular las submatrices que forman el Jacobiano**:

   - $Y_{BUS}$ se calcula a partir de los términos de la matriz de admitancias($Y_{ii}$.$Y_{ij}$), Ya definidos en las ecuaciones (3.6) y (3.7)
   - $D_V$ se construye con los términos real e imaginario de las tensiones de los nudos, a partir de los valores del perfil inicial.
   - $D_I$ se construye con los términos real e imaginario de las intensidades de los nudos, a partir de los valores del perfil inicial.

$$I_i^d\lfloor_0 = I_{i,ai}^d\lfloor_0 + jI_{i,bi}^d\lfloor_0, \quad \text{d = a,b,c,n}, \tag{3.42}$$

3. **Obtener los residuos de potencia e intensidad:** Los residuos de potencia e intensidad se obtienen a partir del sistema de ecuaciones matricial (3.34). En primer lugar se utiliza la ecuación (3.32) para despejar $\triangle U$:

$$(YBUS + D_V^{-1}\cdot D_I\cdot\triangle U = \triangle\Lambda + D_V^{-1}\cdot\triangle S, \tag{3.43}$$

Una vez obtenido $\triangle U$ se utiliza la ecuación 3.23 para calcular $\triangle I$:

$$\triangle U = D_V^{-1}\cdot\triangle S - (D_V^{-1}\cdot D_I)\cdot\triangle U, \tag{3.44}$$

4. **Actualizar los valores de las variables de estado:** Calculados los residuos, solo queda actualizar los valores de tensiones e intensidades del sistema:

$$\begin{bmatrix} f_i^d\lfloor_1 = \triangle f_i^d + f_i^d\lfloor_0 \\ e_i^d\lfloor_1 = \triangle e_i^d + e_i^d\lfloor_0 \end{bmatrix}, \quad \text{d = a,b,c,n}, \tag{3.45}$$

$$\begin{bmatrix} I_{ai}^d\lfloor_1 = \triangle I_{ai}^d + I_{ai}^d\lfloor_0 \\ I_{bi}^d\lfloor_1 = \triangle I_{bi}^d + I_{bi}^d\lfloor_0 \end{bmatrix}, \quad \text{d = a,b,c,n}, \tag{3.46}$$

5. **Comprobar que se cumple el criterio de convergencia establecido**.Si se cumple el criterio los resultados son válidos y el problema se considerará resuelto, si no se cumple este criterio se volverá al paso 1º. y se repetirán los mismos pasos excepto calcular $Y_{BUS}$, ya que esta no varía.

### 3.1.6 Inclusión de nudos PV

De manera similar a lo desarrollado en la sección (2.5.5), la singularidad de la matriz $D_V$ impide resolver el problema mediante el método iterativo descrito en la sección anterior.

Si para un nudo PQ tenemos la expresión (3.32), para nudos PV las matrices de intensidades Y tensiones $D_I, D_V$ cambian y son de la forma:

$$\mathrm{D}_{Ii} = \begin{bmatrix} I_{bi}^a & I_{ai}^a & 0 & 0 & 0 & 0 & -I_{bi}^a & -I_{ai}^a \\ 2(f_i^a - f_i^n) & 2(e_i^a - e_i^n) & 0 & 0 & 0 & 0 & -2(f_i^a - f_i^n) & -2(e_i^a - e_i^n) \\ 0 & 0 & I_{bi}^b & I_{ai}^b & 0 & 0 & -I_{bi}^b & -I_{ai}^b \\ 0 & 0 & 2(f_i^b - f_i^n) & 2(e_i^b - e_i^n) & 0 & 0 & -2(f_i^b - f_i^n) & -2(e_i^b - e_i^n) \\ 0 & 0 & 0 & 0 & I_{bi}^c & I_{ai}^c & -I_{bi}^c & -I_{ai}^c \\ 0 & 0 & 0 & 0 & 2(f_i^c - f_i^n) & 2(e_i^c - e_i^n) & -2(f_i^c - f_i^n) & -2(e_i^c - e_i^n) \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{3.47}$$

$$D_{Vi} = \begin{bmatrix} e_i^a - e_i^n & f_i^a - f_i^n & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & e_i^b - e_i^n & f_i^b - f_i^n & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e_i^c - e_i^n & f_i^c - f_i^n & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \tag{3.48}$$

El hecho de añadir estas matrices modificadas en el Jacobiano provoca que este último se acerque peligrosamente a la singularidad, por lo que la resolución propuesta en el capítulo anterior deja de ser válida, al no poder invertir el Jacobiano, al menos de forma eficiente y robusta y con poco gasto computacional. Debido a esto no se aconseja resolver el sistema de ecuaciones mediante este método ya que la convergencia no esta asegurada.

Para resolver este problema, se utiliza el siguiente método propuesto en [5]. Se añade una nueva variable binaria $\varepsilon$, que puede tomar como valores 0 y 1. Esta variable se incluye en las matrices $D_{Ii}$ y $D_{Vi}$ de todos los nudos y nos permite elegir si el nudo es PQ ($\varepsilon_i = 1$) o PV ($\varepsilon_i = 0$). Por ejemplo, los primeros cuatro elementos de la matriz $D_{Ii}$ ahora serían:

$$D_{Ii} = \begin{bmatrix} I_{bi}^a & I_{ai}^a \\ I_{ai}^a \cdot \varepsilon_i + (1-\varepsilon_i) \cdot 2(f_i^a - f_i^n) & -I_{bi}^a \cdot \varepsilon_i + (1-\varepsilon_i) \cdot 2(e_i^a - e_i^n) \end{bmatrix} \tag{3.49}$$

De esta manera, si $\varepsilon_i = 1$ se obtiene:

$$\begin{bmatrix} I_{bi}^a & I_{ai}^a \\ I_{ai}^a & -I_{bi}^a \end{bmatrix} \tag{3.50}$$

En cambio, si $\varepsilon_i = 0$ se obtiene:

$$D_{Ii} = \begin{bmatrix} I_{bi}^a & I_{ai}^a \\ 2(f_i^a - f_i^n) & 2(e_i^a - e_i^n) \end{bmatrix} \tag{3.51}$$

Análogamente para la matriz $D_{Vi}$:

$$D_{Vi} = \begin{bmatrix} e_i^a - e_i^n & f_i^a - f_i^n \\ \varepsilon_i(f_i^a - f_i^n) & -\varepsilon_i(e_i^a - e_i^n) \end{bmatrix} \tag{3.52}$$

Como se puede observar, el valor de $\varepsilon_i$ determina si esa fila corresponde a un nudo PQ o PV. Realmente, si solo se hiciesen estas modificaciones, no cambiaría nada respecto del caso original.

La clave de este método es asignar un valor $\varepsilon_i \cong 0$ a la variable para las submatrices de las filas correspondientes a nudos PV. Se debe buscar un valor de $\varepsilon_i$ lo más cercano a cero posible pero que afecte lo menor posible al condicionamiento numérico del Jacobiano. Teóricamente cuanto más grande sea el valor de $\varepsilon_i$ más se alejará el resultado obtenido del resultado real y el número de iteraciones será mayor. Por otro lado, cuanto más pequeño (cercano a 0) sea $\varepsilon_i$ peor acondicionado estará el problema desde un punto de vista numérico. De esta manera se garantiza que el Jacobiano no sea singular y tenga inversa.

Una vez solucionado el inconveniente con el Jacobiano, el problema se resolvería de la misma manera que en la sección (2.5.5), utilizando las matrices ampliadas a 4 hilos.

## 3.2 Incorporación de transformadores al modelo

En esta sección se desarrollará un modelo para las simulación de transformadores y su inclusión en el problema del flujo de cargas. El transformador será modelado mediante una matriz de admitancias asociada al transformador que permitirá tratar al transformador como si fuese una clase especial de línea que une dos nudos (primario y secundario).

Este desarrollo y las ecuaciones que definen el modelo se han tomado de [10].

### 3.2.1 Transformadores en redes de distribución

Los transformadores son una parte fundamental de las redes eléctricas de distribución. Normalmente, en redes europeas, las redes de distribución se componen por una red trifásica de media tensión a tres hilos a la que se conectan redes trifásicas de baja tensión de cuatro hilos a partir de transformadores trifásicos. Los transformadores en redes de distribución suelen ser bastante grandes (entre 100-1000 kVA) cuya relación de transformación más habitual es de 20 kV/400 V , aunque depende de la tensión nominal de la red de media tensión. La configuración más habitual suele ser triángulo-estrella, Ya que permite que los desequilibrios de la red de baja tensión provocados por consumos diferentes en cada fase no se transfieran a la red de media tensión.

### 3.2.2 Modelo de transformador

Los transformadores trifásicos son muy difíciles de modelar ya que conseguir un modelo exacto y completo requeriría un largo proceso de simulación y costosas pruebas de cortocircuito. Por ello, para simplificar el modelo, se toma como hipótesis que el transformador trifásico es equivalente a tres transformadores monofásicos conectados entre sí, ya que el error cometido con esta suposición es muy pequeño.

Una vez hecha esta hipótesis, se puede definir la relación intensidad inyectada / tensiones de primario y secundario a través de la ecuación (3.53):

$$\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} Y_t + Y_m & -rY_t \\ -rY_t & r^2 Y_t \end{bmatrix} \cdot \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \tag{3.53}$$

Donde:

- $Y_t$ es la admitancia de cortocircuito del transformador referida al primario,
- $Y_m$ es la admitancia de magnetización del transformador referida al primario,
- r es la relación del transformador monofásico

La Figura 3.2 representa el circuito equivalente de un transformador monofásico.

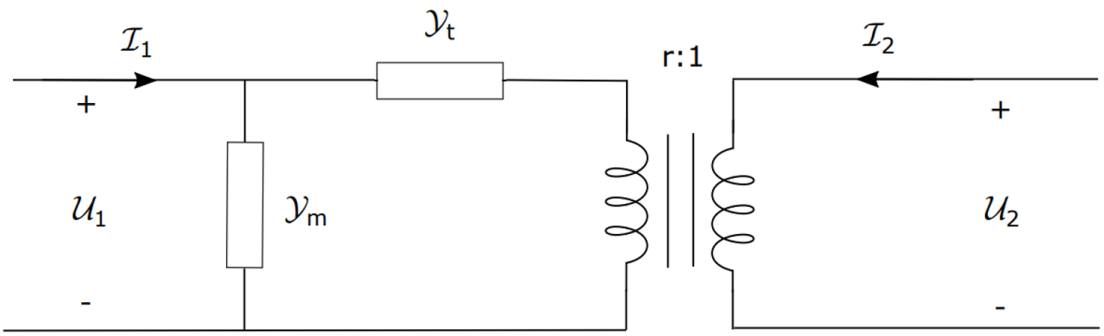


**Figura 3.2** Circuito equivalente de un transformador monofásico.

Para modelar el transformador trifásico se juntan las ecuaciones correspondientes a las tres fases en una única ecuación matricial (ecuación (3.54)). Cada fase esta representada por una letra (a,b,c) y cada toma del primario o secundario del transformador se ha numerado del 1 al 6. Cada fase tiene una relación de transformación propia, nombrada con un subíndice correspondiente a la letra de cada fase.

$$\begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \\ I_6 \end{bmatrix} = \left[\begin{array}{cc|cc|cc} Y_{ta}+Y_{ma} & -r_aY_{ta} & 0 & 0 & 0 & 0 \\ -r_aY_{ta} & r_a^2Y_{ta} & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & Y_{tb}+Y_{mb} & -r_bY_{tb} & 0 & 0 \\ 0 & 0 & -r_bY_{tb} & r_b^2Y_{tb} & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & Y_{tc}+Y_{mc} & -r_cY_{tc} \\ 0 & 0 & 0 & 0 & -r_cY_{tc} & r_c^2Y_{tc} \end{array}\right] \cdot \begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \\ U_6 \end{bmatrix} \tag{3.54}$$

Por otro lado, dependiendo de la configuración del transformador y el tipo de conexión del primario y del secundario la relación entre las tensiones e intensidades en cada lado del transformador es diferente.
La relación anterior es válida para las tensiones que existen entre las bobinas del transformador,tensión de fase ,así como para las intensidades que circulan a través de ellas, intensidad de fase. En la práctica no se trabaja con estos valores si no con los valores de línea, y estos valores vienen determinados por el tipo de conexión tanto de primario como de secundario.

Para pasar de las tensiones e intensidades de los transformadores monofásicos equivalentes a las del transformador trifásico se usa una matriz de transferencia N que relaciona las tensiones e intensidades del transformador monofásico (nombradas con números) con las del transformador trifásico (nombradas con letras).

Al ser la configuración mayormente utilizada en redes de distribución europeas, la matriz $N$ y $N^T$ de las ecuaciones (3.55) y (3.56) corresponde a un conexionado $\triangle$ - Yn. Esta conexión se ilustra en la Figura 3.3

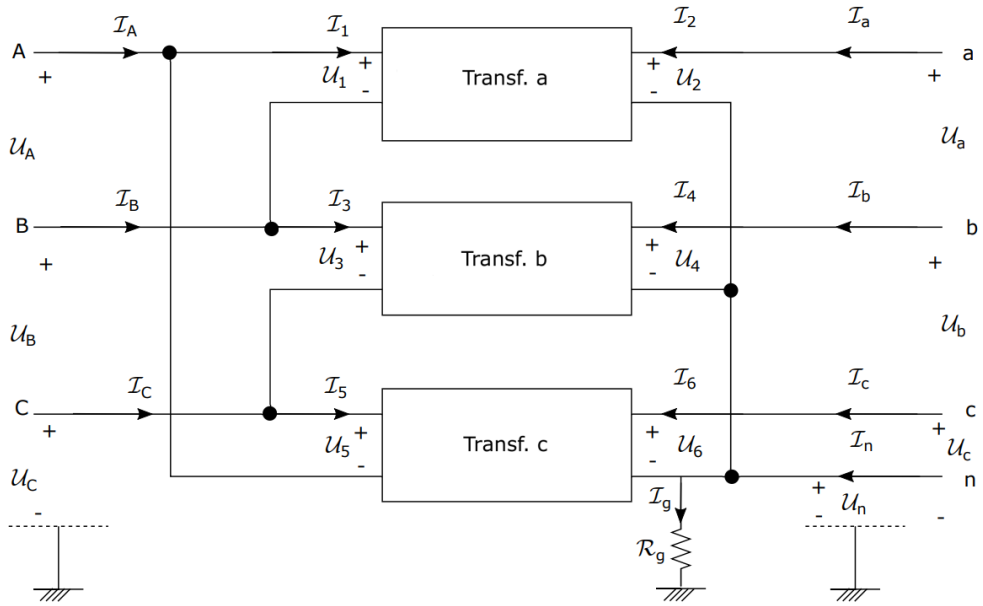


**Figura 3.3** Transformador trifásico configuración triángulo - estrella con neutro puesto a tierra.

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \\ U_6 \end{bmatrix} = \left[\begin{array}{ccc|cccc} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & 1 & -1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{array}\right] \cdot \begin{bmatrix} U_A \\ U_B \\ U_C \\ U_a \\ U_b \\ U_c \\ U_n \end{bmatrix} = N \cdot \begin{bmatrix} U_A \\ U_B \\ U_C \\ U_a \\ U_b \\ U_c \\ U_n \end{bmatrix} \tag{3.55}$$

$$\begin{bmatrix} I_A \\ I_B \\ I_C \\ I_a \\ I_b \\ I_c \\ I_n \end{bmatrix} = N^T \cdot \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ I_4 \\ I_5 \\ I_6 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \frac{U_n}{R_g} \end{bmatrix} \tag{3.56}$$

Sustituyendo la ecuación (3.55) en (3.54) y esta en (3.56) y tomando $Y_{ta} = Y_{tb} = Y_{tc} \quad Y_{ma} = Y_{mb} = Y_{mc}$ se obtiene la ecuación que modela el transformador trifásico:

$$\begin{bmatrix} I_A \\ I_B \\ I_C \\ I_a \\ I_b \\ I_c \\ I_n \end{bmatrix} = \left[\begin{array}{ccc|ccc|c} 2(Y_t+Y_m) & -Y_t-Y_m & -Y_t-Y_m & -r_aY_t & 0 & r_cY_t & (r_a-r_c)Y_t \\ -Y_t-Y_m & 2(Y_t+Y_m) & -Y_t-Y_m & r_aY_t & -r_bY_t & 0 & (r_b-r_a)Y_t \\ -Y_t-Y_m & -Y_t-Y_m & 2(Y_t+Y_m) & 0 & r_bY_t & -r_cY_t & (r_c-r_b)Y_t \\ \hline -r_aY_t & r_aY_t & 0 & -r_a^2Y_t & 0 & 0 & -r_a^2Y_t \\ 0 & -r_bY_t & r_bY_t & 0 & -r_b^2Y_t & 0 & -r_b^2Y_t \\ r_cY_t & 0 & -r_cY_t & 0 & 0 & -r_c^2Y_t & -r_c^2Y_t \\ \hline (r_a-r_c)Y_t & (r_b-r_a)Y_t & (r_c-r_b)Y_t & -r_a^2Y_t & -r_b^2Y_t & -r_c^2Y_t & (r_a^2+r_b^2+r_c^2)Y_t+\frac{1}{R_g} \end{array}\right] \cdot \begin{bmatrix} U_A \\ U_B \\ U_C \\ U_a \\ U_b \\ U_c \\ U_n \end{bmatrix} \tag{3.57}$$

En el caso de que $R_g = 0$, el último término de la diagonal no estaría definido. En ese caso, se procede de forma similar a lo establecido en la sección 3.8 y se define $U_n = 0$, al ser la tensión de neutro igual a la tensión de tierra. Se elimina por lo tanto la última fila de la matriz. Por otro lado, la corriente del neutro puede obtenerse aplicando la primera ley de Kirchhoff al secundario del transformador.

### 3.2.3 Adaptación a nuestro problema

La ecuación (3.59) es equivalente a la matriz $Y_{BUS}$ [11]:

$$I = Y_{BUS} \cdot U \tag{3.58}$$

Por lo tanto, los transformadores pueden modelarse como una "línea" que une dos nudos a través de esta matriz de admitancias.

En este caso, se tomarán algunas hipótesis adicionales para simplificar el modelo:

- La admitancia de magnetización es despreciable frente a la admitancia de cortocircuito del transformador $Y_t >> Y_m$, por lo que $Y_t + Y_m \sim Y_t$.
- Las relaciones de transformación de cada fase suelen ser idénticas en los transformadores de uso habitual, por lo que se tomará $r_a = r_b = r_c$
- Para que la matriz que modele el transformador pueda incluirse en el modelo es necesario añadir una fila y una columna de ceros que simulen un "neutro" en el lado del primario, ya que el algoritmo trabaja con matrices de admitancias de dimensión 8x8.

Por lo tanto, la matriz de admitancias del transformador será de la forma:

$$\begin{bmatrix} I_A \\ I_B \\ I_C \\ I_N \\ I_a \\ I_b \\ I_c \\ I_n \end{bmatrix} = \left[\begin{array}{cccc|cccc} 2Y_t & -Y_t & -Y_t & 0 & -r_aY_t & 0 & r_cY_t & 0 \\ -Y_t & 2Y_t & -Y_t & 0 & r_aY_t & -r_bY_t & 0 & 0 \\ -Y_t & -Y_t & 2Y_t & 0 & 0 & r_bY_t & -r_cY_t & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline -r_aY_t & r_aY_t & 0 & 0 & -r_a^2Y_t & 0 & 0 & -r_a^2Y_t \\ 0 & -r_bY_t & r_bY_t & 0 & 0 & -r_b^2Y_t & 0 & -r_b^2Y_t \\ r_cY_t & 0 & -r_cY_t & 0 & 0 & 0 & -r_c^2Y_t & -r_c^2Y_t \\ 0 & 0 & 0 & 0 & -r_a^2Y_t & -r_b^2Y_t & -r_c^2Y_t & (r_a^2+r_b^2+r_c^2)Y_t + \frac{1}{R_g} \end{array}\right] \cdot \begin{bmatrix} U_A \\ U_B \\ U_C \\ U_N \\ U_a \\ U_b \\ U_c \\ U_n \end{bmatrix} \tag{3.59}$$

Esta matriz puede expresarse como:

$$\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} Y_{pp} & Y_{ps} \\ Y_{sp} & Y_{ss} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \tag{3.60}$$

Expresión análoga a :

$$\begin{bmatrix} I_1 \\ I_2 \end{bmatrix} = \begin{bmatrix} Y_{11} & Y_{12} \\ Y_{21} & Y_{22} \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \tag{3.61}$$

# 4 Herramienta de programación en Python

Tomando como punto de partida la formulación desarrollada en los capítulos 2 y 3, en este capítulo se describirá la herramienta de programación implementada para la resolución del problema de flujo de cargas en redes trifásicas de cuatro hilos

## 4.1 Lenguaje y entorno de programación

Para ello se hará uso del lenguaje de programación Python, concretamente de la distribución Anaconda [12] [13]. Anaconda ofrece varias herramientas y espacios que, basandose en Python, permiten procesar grandes volúmenes de información, análisis predictivo y cálculos científicos. Se ha utilizado Spyder como entorno de programación para facilitar la valoración de los resultados,la limpieza del código y la búsqueda de errores.

### 4.1.1 Tipos de datos en Python

Python tiene muchos tipos de datos, en esta sección hablaremos sobre aquellos utilizados en este trabajo, para que las secciones siguientes sean más sencillas de entender. Por un lado están los datos de tipo numérico. Existen tres tipos de datos numéricos básicos: enteros, números de punto flotante (simularía el conjunto de los números reales) y los números complejos.

Por otro lado tenemos las cadenas de caracteres, también conocidas como *Strings*.En tercer lugar están las listas. Una lista es una de las estructura de datos más versátil. Lo especial de las listas en Python es que nos permiten almacenar cualquier tipo de valor como enteros, cadenas y hasta otras funciones; por ejemplo:

```
lista = [1, 2.5, 'Flujo de cargas', [5,6] ,4]
```

Una lista es un arreglo de elementos donde podemos ingresar cualquier tipo de dato. Para acceder a los datos dentro de una lista se utiliza el índice de ese dato dentro de la lista.

$$lista[0] \rightarrow 1$$

$$lista[3] \rightarrow [5,6]$$

En la herramienta que se ha implementado las listas son una estructura de datos esencial, ya que permiten almacenar todos los datos necesarios para la resolución del problema. Como único inconveniente, no es posible realizar operaciones matemáticas con listas, pero se soluciona utilizando la librería *Numpy* como se explicará en la sección siguiente.

### 4.1.2 Librerías utilizadas

Python tiene una gran variedad de librerías de código que permiten tener a disposición una amplia gama de herramientas y operadores sin tener que programar todo desde cero. Durante el desarrollo de la herramienta de programación se han utilizado varias librerías, a destacar:

- **Numpy** es una librería especializada en el cálculo numérico y el análisis de datos. Permite trabajar con matrices (arrays) en lugar de con listas de Python. También es de utilidad para trabajar con números complejos y tiene implementado distintos operadores algebraicos y constantes matématicas como el número $\pi$.
- **Matplotlib/Seaborn** son dos librerías muy útiles para dibujar y representar datos a través de gráficas. La primera es una librería que imita el comando "plot" de *MATLAB*, con todas sus herramientas para la representación de información. Seaborn, por otro lado, se ha utilizado como complemento a esta última y sirve para dar una mejor presentación y forma a las gráficas obtenidas.
- **Pandas** es una de las librerías más utilizadas en Python para el tratamiento de datos. Permite sacar datos tabulados y clasificados a partir de los resultados obtenidos en las simulaciones.

### 4.1.3 Objetos y clases

Python es un lenguaje de programación orientado a objetos, que es un paradigma de programación en la que los datos y las operaciones que pueden realizarse con esos datos se agrupan en unidades lógicas llamadas *objetos*.

Los objetos suelen representar conceptos del dominio del programa, como un nudo de la red, una línea etc. Los datos que describen las características del objeto se llaman atributos y son la parte estática del objeto, mientras que las operaciones que puede realizar el objeto se llaman métodos y son la parte dinámica del objeto.

Por ejemplo, una línea de la red puede representarte como un objeto:

- **Atributos**: Intensidad que circula por ella, impedancia, número de línea, nudos que conecta...
- **Métodos**: Calcular pérdidas, calcular potencia...

Para acceder a un atributo o utilizar un método de un objeto se utilizan los comandos *objeto.atributo* y *objeto.método*. Por ejemplo, si definimos el objeto linea y le asignamos un atributo intensidad (I), si queremos obtener la intensidad de esa rama escribiríamos

*linea.I*

Los objetos con los mismos atributos y métodos se agrupan clases. Las clases definen los atributos y los métodos, y por tanto, la semántica o comportamiento que tienen los objetos que pertenecen a esa clase. Se puede pensar en una clase como en un molde a partir del cuál se pueden crear objetos.

Para declarar una clase se utiliza la palabra clave class seguida del nombre de la clase y dos puntos, de acuerdo a la sintaxis de la Figura 4.1:

```
class <nombre-clase>:
    <atributos>
    <métodos>
```

**Figura 4.1** Definición de una clase.

Los atributos se definen igual que las variables mientras que los métodos se definen igual que las funciones. Tanto unos como otros tienen que estar indentados por 4 espacios en el cuerpo de la clase.

Los métodos de una clase son las funciones que definen el comportamiento de los objetos de esa clase. Se definen como las funciones con la palabra reservada *def*. La única diferencia es que su primer parámetro es especial y se denomina *self*. Este parámetro hace siempre referencia al objeto desde donde se llama el método, de manera que para acceder a los atributos o métodos de una clase en su propia definición se puede utilizar la sintaxis *self.atributo* o *self.método*.

En el código implementado para este trabajo se han definido varias clases que engloban los componentes de una red eléctrica de distribución:

- **Clase BUS**: Esta clase define los nudos de la red a simular. Tendrá los siguientes atributos:
  - **Tensión del nudo**: lista de cuatro componentes con las tensiones complejas de las tres fases y la tensión del neutro.
  - **Potencia consumida/generada especificada en el nudo**: lista con tres componentes con las potencias de cada fase.
  - **Potencia reactiva especificada** : lista con tres componentes con las potencias de cada fase.
  - **Tipo de nudo**: Se asigna el tipo de nudo mediante un string: 'PQ', 'PV', 'Slack'.
  - **Resistencia de puesta a tierra**: Resistencia $R_g$ entre el neutro y tierra en ese nudo.
  - **Módulo de la tensión asignada**: En el caso de los nudos PV, se define también un atributo para la tensión especificada. Se diferencia del atributo de tensión de nudo en que no varía durante el proceso iterativo y que no es valor complejo. El ángulo de la tensión no se fija, al no fijarse la potencia reactiva.
  - **Intensidad inyectada en el nudo**: se inicializa con el método desarrollado en el capítulo 3.
  - **Conexión con líneas**: este atributo permite saber que líneas entran o salen de un nudo.
- **Clase LINE**:Esta clase define las líneas de la red a simular. Tiene los siguientes atributos:
  - **Impedancia de la línea**: es una matriz de dimensión 4x4, al trabajar con redes de 4 hilos. Esta formada por las impedancias propias y mutuas entre fases en $\Omega/km$.
  - **Longitud de la línea**: permite calcular la impedancia total de la línea.
  - **Nudos que conecta**: este atributo esta relacionado con el atributo Conexión con líneas de las Clase BUS.
- **Clase TRANSFORMER**:Esta clase define el transformador de cabecera. Es una clase muy similar a la Clase LINE, pero tiene diferencias que hacen que sea necesario definir una clase adicional. Sus atributos son:
  - **Admitancia de cortocircuito**: referida al lado del primario en configuración de $\triangle$.
  - **Relación de transformación**: una para cada fase
  - **Nudos que conecta**:este atributo esta relacionado con el atributo Conexión con líneas de las Clase BUS.
  - **Resistencia de puesta a tierra del transformador**
- **Clase GRID**: Esta clase define la red en si misma. Los métodos de esta clase permiten la construcción del algoritmo y llevar a cabo el proceso iterativo.Estos métodos se describen en la sección 4.2.

### 4.1.4 Métodos *check*

Por otro lado, para comprobar que la solución que se alcanza es coherente se han utilizado los métodos check dentro de cada tipo de clase. Estas funciones check comprueban que:

- **Nudos**: La función check comprueba que se cumple la primera Ley de Kirchhoff en cada nudo.
- **Lineas y transformadores**: Comprueba que la caída de tensión entre los nudos que conectan la rama sea igual a la impedancia de la línea por la corriente que circula por ella.

## 4.2 Estructura del algoritmo implementado

La herramienta de programación esta formada por dos programas o 'scripts'. En el primer script se encuentran la definición de las clases y las funciones implementadas. En el segundo es en el que introducimos los datos de entrada del programa y los atributos de las clases y en el que se llama a las funciones que ejecutan el algoritmo.

### 4.2.1 Datos de entrada

Hay varios tipos de datos de entrada. Los más importantes son aquellos relacionados con las clases definidas en la sección anterior. Para incorporar un nuevo objeto a una clase, se define un método *Add_clase* dentro de la clase GRID, que incluye los atributos de ese objeto. Para introducir un nuevo objeto, se utiliza el comando *grid.add_clase*. Así, para cada clase tenemos:

```
net.add_BUS(ref, P, Q, U, tipo, R_g, V_esp)
net.add_LINE(Z, long,BUS_0, BUS_1)
net.add_TRANSFORMER( BUS_0, BUS_1), Y_t, r, R_gt)
```

Por otro lado, también se definen como datos de entrada las constantes del problema, como la matriz de impedancias de línea, la tensión del nudo Slack, el grado de desequilibrio... etc

### 4.2.2 Inicialización del algoritmo

Una vex definidos los datos de entrada se crea un perfil inicial de tensiones e intensidades mediante un método dentro de la clase GRID. Para los nudos PQ se inicializan los valores de las tensiones de fase igualandolas a las tensiones de fase del nudo Slack. Para los nudos PV, si se trabaja con un transformador en cabecera, es recomendable asignar como tensión inicial una tensión que sea de módulo exacto al especificado y trifásica simétrica en ángulos (0º, -120º,120º). Esto se hace para mejorar el condicionamiento numérico. Si se inicializan los nudos PV con la tensión de media del nudo Slack, se parte de un punto inicial muy alejado del valor especificado y que puede crear problemas e convergencia.

Una vez definido el perfil inicial de tensiones, se define el perfil inicial de intensidades. Para ello, se hace uso de la fórmula (3.37) del capítulo 3. En los nudos PV, donde la potencia reactiva no esta especificada y es una variable del problema, se toma como hipótesis para inicializar las intensidades que el factor de potencia es igual a la unidad ( $\cos\phi = 0$).

### 4.2.3 Construcción de la matriz de admitancias

Para construir la matriz de admitancias $Y_{BUS}$ se parte de los siguientes datos de entrada:

- Matriz de impedancias complejas de cada línea.
- Matriz de admitancias complejas del transformador
- Relaciones nudo-línea
- Resistencias de puesta a tierra

Se crea un método dentro de la clase GRID que construya la matriz. En primer lugar, se calculan los componentes $Y_{ij}$ de las líneas a partir de las matrices de impedancias utilizando el comando *np.linalg.solve*, que calcula su inversa. Estas matrices se van colocando en la matriz de admitancias mediante un doble bucle for que recorre primero los buses y después las líneas que llegan y salen de esos buses, utilizando los atributos que unen nudos y líneas.

Una vez calculados los componentes $Y_{ij}$, las matrices $Y_{ii}$ se calculan sumando los primeros para cada fila, siguiendo la ecuación (2.8).

Al mismo tiempo, se incorpora las submatrices de la matriz de admitancias (ecuacion 3.60) en los nudos que representan el primario y el secundario del transformador.

Una vez colocados todos los componentes en la matriz de admitancias, se separa cada elemento en cuatro, formando las submatrices de las ecuaciones (3.6) y (3.7). Por último, se añade el término correspondiente a la puesta a tierra en los componentes de neutro de cada nudo (ecuación (3.8).

### 4.2.4 Construcción de las matrices del Jacobiano

Como se vio en el capítulo 2, el Jacobiano esta formado por cuatro submatrices: $Y_{BUS}, I, D_V, D_I$. La matriz de admitancias ya se ha construido y la matriz identidad se puede obtener mediante el comando de la librería Numpy *np.eye()*. Se forman ahora las matrices $D_V, D_I$.

Este proceso requiere como entrada los perfiles iniciales de tensión y corriente calculados previamente y definir el valor para los nudos PV de la variable binaria $\varepsilon$. Mediante un método dentro de la clase GRID se recorren los elementos dentro de la clase bus y se colocan los atributos de tensión e intensidad de estos elementos en las posiciones definidas por las ecuaciones (3.19) y (3.31). Por otro lado, para redes con nudos PQ y PV, la matriz se construye añadiendo la variable $\varepsilon$ de la forma definida en la sección 3.1.6.

Una vez colocados todos los componentes en las matrices se eliminan las filas correspondientes al nudo Slack.

### 4.2.5 Obtención de los residuos

Para calcular los residuos son necesarios como datos de entrada las tensiones e intensidades de los buses, igual que en el paso anterior. Se crea otro método dentro de la clase GRID que realice este proceso. El primer paso será formar varias listas:

1. Una lista con las tensiones de todas las fases de cada nudo.
2. Una lista con las intensidades de todas las fases de cada nudo.
3. Una lista donde se almacenaran los términos correspondientes a los residuos de potencia

Una vez definidas estas listas, las dos primeras se llenan recorriendo todos los elementos tipo bus, almacenando sus atributos de tensión y corriente respectivamente.

Para calcular los residuos de intensidad se utiliza la ecuación (3.26), teniendo como datos de entrada las listas de tensiones e intensidades y la matriz $Y_{BUS}$ del capítulo 3. Los residuos de potencia se calculan de manera diferente para nudos PQ y para nudos PV. Para los primeros se utiliza la ecuación (3.32) mientras que para los segundos la ecuación (2.68), particularizada para cuatro hilos.

Con los vectores de residuos ya formados, se forma una lista concatenando ambos y se realiza una verificación para obtener el valor más grande de ambos vectores. Este valor será el que se compare con la tolerancia para determinar si se detiene el proceso iterativo o la solución no es todavía lo suficientemente buena.

### 4.2.6 Cálculo de $\triangle U, \triangle I$

Como se comentó en los capítulos 2 y 3, la inclusión de nudos PV en el modelo modifica la forma en la que se calculan $\triangle U, \triangle I$. Para resolver el sistema de ecuaciones (3.34) con nudos PQ y PV en primer lugar se construye el Jacobiano, colocando las matrices obtenidas en las secciones anteriores en sus lugares correspondientes dentro del Jacobiano. Se utiliza entonces el comando *np.linalg.solve*:

```
delta = np.linalg.solve(J, res)
```

Siendo *delta* el vector formado por $\triangle U, \triangle I$, J el Jacobiano y res el vector formado por los residuos de intensidad y potencia. Después se parte el vector *delta* por la mitad, siendo la primera parte $\triangle U$ y la segunda $\triangle I$. Todos estos pasos forman parte de un nuevo método dentro de la clase GRID.

Los componentes de estos vectores se suman a los atributos de tensión y corriente de los elementos tipo bus y se vuelve a empezar si la tolerancia es inferior al mayor de los residuos de potencia. Para ello se utiliza un bucle while:

```
while tol< maxres
```

Donde *maxres* es el mayor valor dentro del vector de residuos.

### 4.2.7 Obtención de resultados

Una vez que el método ha convergido, hace falta extraer los resultados para analizarlos. Para ello, se utilizan las librerías *Matplotlib* y *Pandas*. Estas librerías permiten graficar los valores obtenidos y representarlos en tablas respectivamente.

# 5 Modelo de la red y simulación

En este capítulo se describirá el modelo de red implementado para la validación del algoritmo y se compararán los resultados obtenidos con el programa OpenDSS.

Para ello, en primer lugar se simularán varios ejemplos de validación donde se irán implementado herramientas del código paso a paso. Una vez simulados y validados estos ejemplos se procederá a montar una red mayor, basándose en la red de media tensión con configuración europea del CIGRE.

## 5.1 Modelo inicial de validación

Los ejemplos que se van a simular son los siguientes:

1. Red de 4 nudos PQ sin trafo de cabecera
2. Red de 4 nudos PQ + PV sin trafo de cabecera
3. Red de 4 nudos PQ con trafo de cabecera
4. Red de 4 nudos PQ + PV con trafo de cabecera

Todos estos ejemplos tendrán cono base la misma topología de red, formada por una fuente ideal de tensión (Slack) que alimenta una red de 4 nudos y 3 líneas. la Figura 5.1 muestra esta topología base:

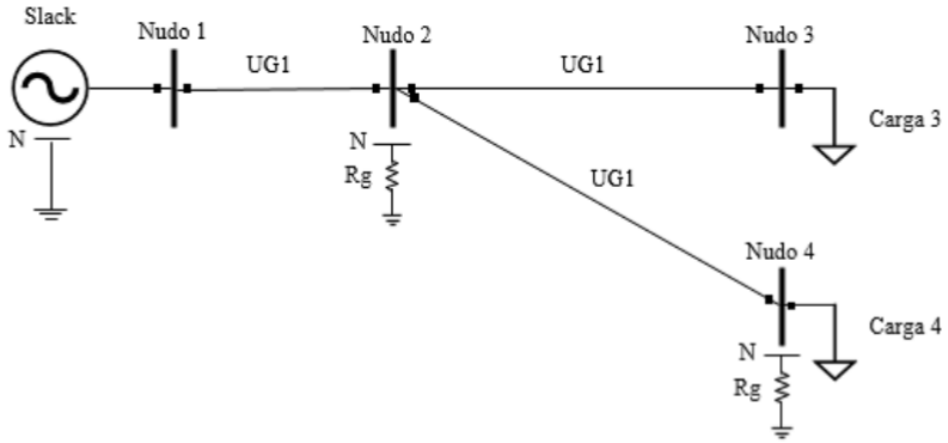


**Figura 5.1** Topología del modelo de validación.

El nudo Slack tendrá una tensión trifásica equilibrada tal como se muestra en la Tabla 5.1:

**Tabla 5.1** Nudo Slack de baja tensión.

| Fase | Tensión (V) |
|---|---|
| a | $400\text{V}/\sqrt{3}\ \angle 0$ |
| b | $400\text{V}/\sqrt{3}\ \angle -120$ |
| c | $400\text{V}/\sqrt{3}\ \angle 120$ |

Las líneas se modelan a través de su impedancia por fase la cual depende del tipo de conductor. Se toman como valores de referencia las matrices de la tabla 7.25 del documento del CIGRE [14], representada en la Figura 5.2. En este caso todas las líneas estarán definidas con el conductor *UG1*.

| Conductor ID/ Installation | | The primitive impedance matrix [Ω/km] | | | |
|---|---|---|---|---|---|
| | | A | B | C | N |
| UG1 / 3-ph | A | 0.211 + j0.747 | 0.049 + j0.673 | 0.049 + j0.651 | 0.049 + j0.673 |
| | B | 0.049 + j0.673 | 0.211 + j0.747 | 0.049 + j0.673 | 0.049 + j0.651 |
| | C | 0.049 + j0.651 | 0.049 + j0.673 | 0.211 + j0.747 | 0.049 + j0.673 |
| | N | 0.049 + j0.673 | 0.049 + j0.651 | 0.049 + j0.673 | 0.211 + j0.747 |
| UG2 / 3-ph | A | 0.314 + j0.762 | 0.049 + j0.687 | 0.049 + j0.665 | 0.049 + j0.687 |
| | B | 0.049 + j0.687 | 0.314 + j0.762 | 0.049 + j0.687 | 0.049 + j0.665 |
| | C | 0.049 + j0.665 | 0.049 + j0.687 | 0.314 + j0.762 | 0.049 + j0.687 |
| | N | 0.049 + j0.687 | 0.049 + j0.665 | 0.049 + j0.687 | 0.314 + j0.762 |
| UG3 / 3-ph | A | 0.871 + j0.797 | 0.049 + j0.719 | 0.049 + j0.697 | 0.049 + j0.719 |
| | B | 0.049 + j0.719 | 0.871 + j0.797 | 0.049 + j0.719 | 0.049 + j0.697 |
| | C | 0.049 + j0.697 | 0.049 + j0.719 | 0.871 + j0.797 | 0.049 + j0.719 |
| | N | 0.049 + j0.719 | 0.049 + j0.697 | 0.049 + j0.719 | 0.871 + j0.797 |

**Figura 5.2** Matrices de impedancia por fase para conductores subterráneos.

La longitud y conexiones de las ramas vienen expresados en la Tabla 5.2:

**Tabla 5.2** Longitud y conexiones de las ramas de la red de validación.

| RAMA | NUDO i | NUDO j | TIPO CONDUCTOR | LONGITUD(m) |
|---|---|---|---|---|
| 1 | 1 | 2 | UG1 | 200 |
| 2 | 2 | 3 | UG1 | 100 |
| 3 | 2 | 4 | UG1 | 200 |

La red tiene varias cargas modeladas como cargas de potencia constante en varios nudos. Estas cargas son monofásicas conectadas en estrella. Se definen con la potencia aparente total trifásica, el factor de potencia y el desequilibrio entre fases. Para la red de nuestro modelo, las cargas están detalladas en la Tabla 5.3:

**Tabla 5.3** Distribución de las cargas en la red.

| NUDO | S(kVA) | $\cos\varphi$ | Fase A (%) | Fase B (%) | Fase C (%) |
|---|---|---|---|---|---|
| 3 | 100 | 1 | 33 | 33 | 33 |
| 4 | 50 | 1 | 33 | 33 | 33 |

Por último, en la Tabla 5.4 se definen las resistencias de puesta a tierra en los nudos:

**Tabla 5.4** Resistencias de puesta a tierra de los nudos del modelo.

| NUDO | R(Ω) |
|---|---|
| 2 | 5 |
| 4 | 5 |

## 5.2 Modelo de validación, ejemplo 1: Red de 4 nudos PQ sin trafo de cabecera

En este primer ejemplo se definen todos los nudos como nudos PQ, a excepción del nudo de cabecera. Las características de la red son las definidas anteriormente. La Tabla 5.5 muestra los resultados utilizando la herramienta desarrollada en Python y la Tabla 5.6 los resultados obtenidos en OpenDSS:

**Tabla 5.5** Datos de tensiones de nudos obtenidos en Python (Ejemplo 1).

| NUDO | $U_a$(V) | $\varphi_a$(º) | $U_b$(V) | $\varphi_b$(º) | $U_c$(V) | $\varphi_c$(º) | $U_n$(V) | $\varphi_n$(º) |
|---|---|---|---|---|---|---|---|---|
| 0 | 230.94 | 0 | 230.94 | -120 | 230.94 | 120 | 0 | 0 |
| 1 | 222.62 | -0.95 | 223.65 | -120.82 | 224.44 | 119.07 | 1.06 | -32.82 |
| 2 | 219.87 | -1.29 | 221.21 | -121.10 | 222.29 | 118.74 | 1.42 | -31.83 |
| 3 | 219.85 | -1.29 | 221.25 | -121.10 | 222.28 | 118.75 | 1.41 | -33.53 |

**Tabla 5.6** Datos de tensiones de nudos obtenidos en OpenDSS (Ejemplo 1).

| NUDO | $U_a$(V) | $\varphi_a$(º) | $U_b$(V) | $\varphi_b$(º) | $U_c$(V) | $\varphi_c$(º) | $U_n$(V) | $\varphi_n$(º) |
|---|---|---|---|---|---|---|---|---|
| 0 | 230.94 | 0 | 230.94 | -120 | 230.94 | 120 | 0 | 0 |
| 1 | 222.59 | -1.06 | 223.63 | -120.80 | 224.45 | 119.10 | 1.06 | -33.1 |
| 2 | 219.84 | -1.30 | 221.20 | -121.10 | 222.30 | 118.70 | 1.42 | -32.1 |
| 3 | 219.81 | -1.30 | 221.22 | -121.109 | 222.30 | 118.80 | 1.41 | -33.8 |

Como se puede observar comparando ambas tablas, se ha alcanzado la misma solución. Hay pequeñas variaciones entre ambas herramientas , que se deben a varios factores:

- El solver que se utiliza no es el mismo.
- La definición de los modelos no es idéntica al 100 %

Donde más diferencias se observa es en el ángulo de la tensión de neutro, aunque no llega a ser una diferencia enormemente significativa. Esto puede deberse a la hipótesis que se hizo en la sección 3.8 para nudos rígidamente puestos a tierra.

Por otro lado, podemos observar como la red, aunque pequeña, tiene un perfil de tensiones coherente y típico del de una red de baja tensión. Las tensiones descienden desde el nudo de cabecera, que esta fijado a la tensión nominal especificada, siendo la tensión de los nudos más pequeña según nos alejamos de del nudo 'Slack'.

Las diferencias entre las tensiones de cada fase se deben a que la impedancia es diferente para cada fase.

En cuánto a los ángulos, apenas varían respecto a los ángulos de las fases del nudo de referencia. Esto es lógico, teniendo en cuenta que las cargas tiene factor de potencia igual a la unidad, por lo que la variación en los ángulos es debida únicamente a la impedancia de las líneas. Al ser estás de carácter inductivo, vemos como los ángulos se retrasan ligeramente respecto del nudo 'Slack'.

Estos resultados validan por lo tanto la herramienta implementada para el caso más sencillo dentro de los ejemplos de validación.

## 5.3 Modelo de validación, ejemplo 2: Red de 4 nudos PV sin trafo de cabecera

Una vez validado el primer caso de simulación, se sustituye la carga del nudo 4 por un generador con las características de la Tabla 5.7:

**Tabla 5.7** Datos del generador.

| NUDO | P(kW) | $\|U_a\|$(V) | $\|U_b\|$(V) | $\|U_c\|$(V) | $\|U_n\|$(V) |
|---|---|---|---|---|---|
| 3 | 50 | $400/\sqrt{(3)}$ | $400/\sqrt{(3)}$ | $400/\sqrt{(3)}$ | 0 |

Los resultados obtenidos con la herramienta implementada en este caso se muestran en la Tabla 5.8:

**Tabla 5.8** Datos de tensiones de nudos obtenidos en Python (Ejemplo 2).

| NUDO | $U_a$(V) | $\varphi_a$(º) | $U_b$(V) | $\varphi_b$(º) | $U_c$(V) | $\varphi_c$(º) | $U_n$(V) | $\varphi_n$(º) |
|---|---|---|---|---|---|---|---|---|
| 0 | 230.94 | 0 | 230.94 | -120 | 230.94 | 120 | 0 | 0 |
| 1 | 228.25 | -0.34 | 228.55 | -120.3 | 228.82 | 119.67 | 0.34 | -21.74 |
| 2 | 225.57 | -0.66 | 226.16 | -120.57 | 226.72 | 119.36 | 0.67 | -25.66 |
| 3 | 230.94 | -0.08 | 230.94 | -120.05 | 230.94 | 119.96 | 0.10 | 61.13 |

Y por otro lado, en OpenDSS se obtienen las tensiones de la Tabla 5.9

**Tabla 5.9** Datos de tensiones de nudos obtenidos en OpenDSS (Ejemplo 2).

| NUDO | $U_a$(V) | $\varphi_a$(º) | $U_b$(V) | $\varphi_b$(º) | $U_c$(V) | $\varphi_c$(º) | $U_n$(V) | $\varphi_n$(º) |
|---|---|---|---|---|---|---|---|---|
| 0 | 230.94 | 0 | 230.94 | -120 | 230.94 | 120 | 0 | 0 |
| 1 | 228.17 | -0.3 | 228.5 | -120.3 | 228.8 | 119.7 | 0.356 | -21.2 |
| 2 | 225.48 | -0.6 | 226.1 | -120.5 | 226.7 | 119.4 | 0.695 | -25.7 |
| 3 | 230.94 | 0 | 230.94 | -120 | 230.94 | 120 | 0 | 60 |

Al igual que en el ejemplo 1, se alcanza la misma solución en ambos casos. Se aprecia una mayor variación que en el caso anterior que es debida a la variable $\varepsilon$ en el nudo PV. Como se comentó en el capítulo 3, esta variable mejoraba el condicionamiento numérico del Jacobiano a cambio de un mayor error en la solución final.

Por otro lado, se observa como la herramienta fija perfectamente la tensión especificada en el nudo PV. Como se asignó la tensión nominal de la red, vemos que la tensión del Nudo 3 es idéntica a la del nudo de referencia.

Los ángulos de los nudos varían menos respecto del nudo de cabecera que en el caso anterior. El nudo PV tiene fijada el módulo de la tensión y la potencia activa, pero no así la potencia reactiva. En este caso el nudo Pv inyecta potencia reactiva en la red y hace que la diferencia de ángulos sea menor.

Por otro lado, se observa como el perfil de tensiones esta más "levantado" que en el ejemplo 1, otro efecto provocado por la reducción de la intensidad necesaria en la red y también afectado por la inyección de potencia reactiva del nudo PV.

Estos resultados validan por lo tanto el método para nudos PV de la herramienta implementada.

## 5.4 Modelo de validación, ejemplo 3: Red de 4 nudos PQ con trafo de cabecera

Una vez validado el modelo para la red sin transformador de cabecera, tanto con nudos PQ como con nudos PV, vamos a introducir un transformador entre el nudo Slack y el primer nudo de la red, tal como se muestra en la Figura 5.3

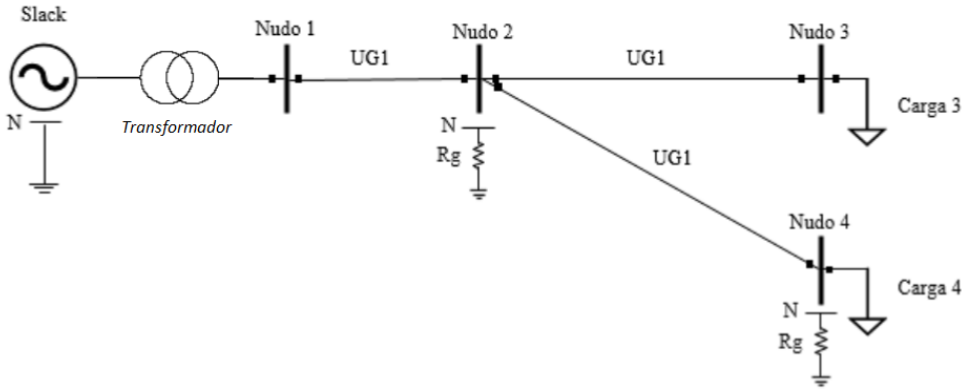


**Figura 5.3** Topología del modelo de validación para los casos con transformador de cabecera.

Se parte del mismo modelo utilizado para los casos anteriores, pero con algunas modificaciones para incluir al trafo. En primer lugar, el nudo de cabecera tendrá una tensión de red de media tensión cuyo valor se muestra en la Tabla 5.10

**Tabla 5.10** Nudo Slack de media tensión.

| Fase | Tensión (V) |
|---|---|
| a | $20\text{kV}/\sqrt{3}\angle 0$ |
| b | $20\text{kV}/\sqrt{3}\angle -120$ |
| c | $20\text{kV}/\sqrt{3}\angle 120$ |

Por otro lado, se definen las características de este transformador que se han tomado de la tabla 7.30 del documento del CIGRE y se representan en la Tabla 5.11:

**Tabla 5.11** Datos del transformador.

| | |
|---|---|
| NUDO INICIO | Nudo 0 |
| NUDO FIN | Nudo 1 |
| S(kVA) | 500 |
| Admitancia de cortocircuito | 0.0024-0.0098j |
| Relación de transformación | $\frac{20kV}{400/\sqrt{3}}$ |
| Resistencia de puesta a tierra | 3 Ω |

Una vez definido el modelo, se ejecuta de nuevo la herramienta en Python y se compara con la misma red en OpenDSS. Los resultados de la primera se muestran en la Tabla 5.12 y las de la segunda en la Tabla 5.13:

**Tabla 5.12** Datos de tensiones de nudos obtenidos en Python (Ejemplo 3).

| NUDO | $U_a$(V) | $\varphi_a$(º) | $U_b$(V) | $\varphi_b$(º) | $U_c$(V) | $\varphi_c$(º) | $U_n$(V) | $\varphi_n$(º) |
|---|---|---|---|---|---|---|---|---|
| 0 | 11 547 | 0 | 11 547 | -120 | 11 547 | 120 | 0 | 0 |
| 1 | 230.09 | 29.27 | 230.07 | -90.68 | 230.07 | 149.29 | 0.005 | -2.49 |
| 2 | 221.53 | 28.29 | 222.76 | -91.62 | 223.51 | 148.39 | 0.911 | -2.08 |
| 3 | 218.68 | 27.97 | 220.38 | -91.85 | 221.36 | 148.02 | 1.234 | -1.09 |
| 4 | 218.96 | 27.82 | 220.36 | -91.70 | 221.37 | 147.96 | 1.221 | -2.79 |

**Tabla 5.13** Datos de tensiones de nudos obtenidos en OpenDSS (Ejemplo 3).

| NUDO | $U_a$(V) | $\varphi_a$(º) | $U_b$(V) | $\varphi_b$(º) | $U_c$(V) | $\varphi_c$(º) | $U_n$(V) | $\varphi_n$(º) |
|---|---|---|---|---|---|---|---|---|
| 0 | 11 547 | 0 | 11 547 | -120 | 11 547 | 120 | 0 | 0 |
| 1 | 229.98 | 29.3 | 229.99 | -90.7 | 229.99 | 149.3 | 0 | -95 |
| 2 | 221.51 | 28.3 | 222.78 | -91.6 | 223.49 | 148.4 | 0.899 | -13.9 |
| 3 | 218.66 | 28.0 | 220.42 | -91.85 | 221.35 | 148.1 | 1.199 | -13.9 |
| 4 | 218.63 | 28.0 | 220.45 | -91.9 | 221.34 | 148.1 | 1.199 | -13.9 |

Como se puede observar comparando ambas tablas, se obtiene una misma solución aproximada tanto en valores absolutos como en ángulos. Se valida por tanto también el modelo de transformador implementado.

En este caso la tensión nominal cambia a una de media tensión, que se asigna al nudo de cabecera. El nexo de unión con el resto de la red es un transformador $\triangle - Y$ que retrasa el secundario 30º respecto del primario. Por ese motivo, vemos que los nudos de la red de baja tensión (nudos 1-5) tienen un ángulo de aproximadamente 30º. De igual manera que en el ejemplo1, vemos como los ángulos caen ligeramente según nos vamos alejando del nudo de cabecera. Esto es debido de nuevo al efecto de las líneas de la red.

Respecto a los módulos de las tensiones, podemos ver como hay una mayor caída de tensión media en los nudos de la red.

## 5.5 Modelo de validación, ejemplo 4: Red de 4 nudos PV con trafo de cabecera

Por último, vamos a validar el modelo más complejo de los cuatro, que será una combinación de los ejemplos 2 y 3. Para ello, montaremos de nuevo la red definida al principio de este capítulo, añadiendo el transformador del ejemplo 3 (Tabla 5.11) y el generador del ejemplo 2 (Tabla 5.7).

Los resultados en Python y en OpenDSS se muestran en las Tabla 5.14 y en la Tabla 5.15, respectivamente:

**Tabla 5.14** Datos de tensiones de nudos obtenidos en Python (Ejemplo 4).

| NUDO | $U_a$(V) | $\varphi_a$(º) | $U_b$(V) | $\varphi_b$(º) | $U_c$(V) | $\varphi_c$(º) | $U_n$(V) | $\varphi_n$(º) |
|---|---|---|---|---|---|---|---|---|
| 0 | 11 547 | 0 | 11 547 | -120 | 11 547 | 120 | 0 | 0 |
| 1 | 230.62 | 30.2 | 230.65 | -89.9 | 230.65 | 150.21 | 0 | -136.95 |
| 2 | 227.82 | 29.94 | 228.13 | -89.99 | 228.41 | 149.84 | 0.296 | -26.8 |
| 3 | 225.13 | 29.63 | 225.73 | -90.26 | 226.30 | 149.66 | 0.6348 | -7.5 |
| 4 | 230.4 | 30.27 | 230.4 | -89.69 | 230.4 | 150.31 | 0.124 | 137 |

**Tabla 5.15** Datos de tensiones de nudos obtenidos en OpenDSS (Ejemplo 4).

| NUDO | $U_a$(V) | $\varphi_a$(º) | $U_b$(V) | $\varphi_b$(º) | $U_c$(V) | $\varphi_c$(º) | $U_n$(V) | $\varphi_n$(º) |
|---|---|---|---|---|---|---|---|---|
| 0 | 11 547 | 0 | 11 547 | -120 | 11 547 | 120 | 0 | 0 |
| 1 | 230.62 | 29.8 | 230.63 | -90.2 | 230.63 | 149.8 | 0 | -139.5 |
| 2 | 227.78 | 29.4 | 228.3 | -90.5 | 228.42 | 149.5 | 0.337 | -22.8 |
| 3 | 225.07 | 29.1 | 225.96 | -90.1 | 226.28 | 149.2 | 0.659 | -17.1 |
| 4 | 230.4 | 29.7 | 230.63 | -90.2 | 230.48 | 149.8 | 0.126 | -95.1 |

Este es el último caso de validación. Al igual que en el ejemplo 2, se aprecia un mayor error entre la solución con OpenDSS y la solución con la herramienta en Python, pero ambas alcanzan la misma solución.

Se observa como las tensiones son de nuevo mayores cuando se introduce un nudo PV en la red y como los ángulos varían menos, en este caso respecto del ángulo de 30º del secundario del transformador.

Con este ejemplo se ha validado la integración completa de todas las características de la herramienta de programación: nudos PQ, nudos PV y el transformador.

# 6 Simulación de una red residencial de 18 nudos

Una vez validado nuestra herramienta matemática en Python vamos a proceder a simular un modelo de red residencial europea de baja tensión basada en la red propuesta por el CIGRE [14].

La red que se va a modelar y sobre la que se llevarán a cabo las simulaciones es una red de media tensión con configuración típica europea para redes residenciales, elaborada por el CIGRE. Esta formada por un nudo de cabecera de media tensión que alimenta una red de baja tensión formada por 18 nudos. Es una red a cuatro hilos con tres fases y neutro. En concreto, la red que se va a utilizar como modelo corresponde al feeder izquierdo de la Figura 6.1:

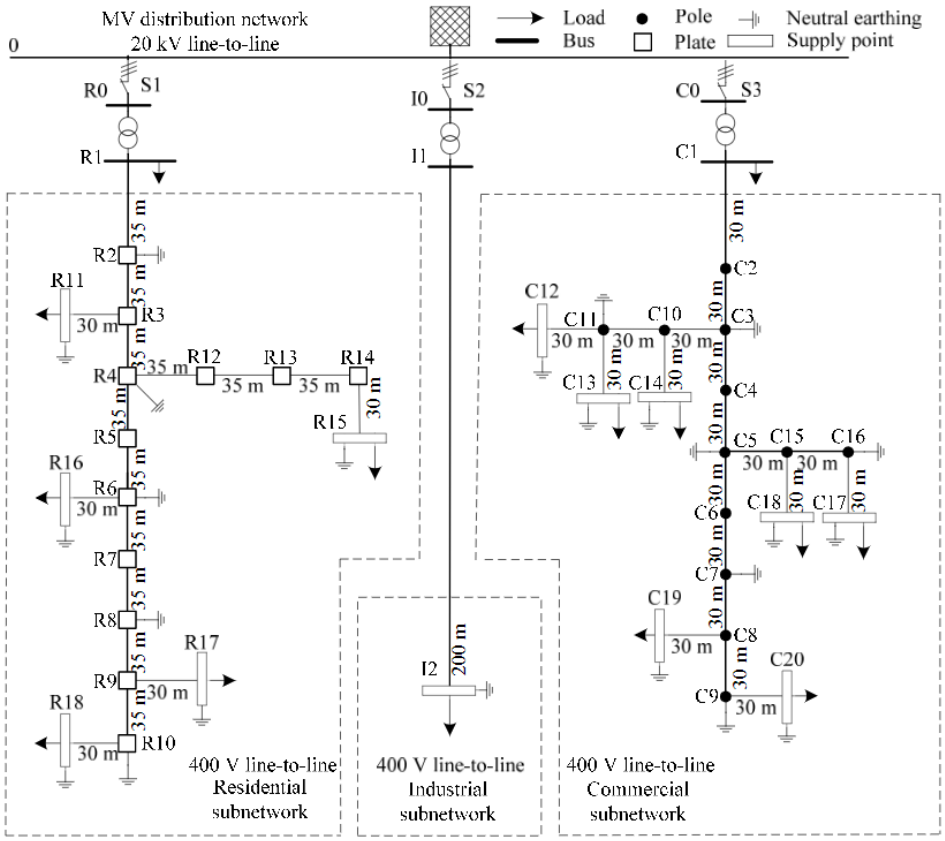


**Figura 6.1** Topología de una red típica europea.

## 6.1 Características de la red

De la misma manera que para los ejemplos propuestos como modelos de validación de la herramienta de programación en Python en el capítulo anterior, vamos a empezar definiendo los parámetros y características de la red a simular.

La Tabla 6.1 define la tensión del nudo de cabecera o nudo Slack:

**Tabla 6.1** Nudo Slack de la red de 18 nudos.

| Fase | Tensión (V) |
|---|---|
| a | $20\text{kV}/\sqrt{3}\angle 0$ |
| b | $20\text{kV}/\sqrt{3}\angle -120$ |
| c | $20\text{kV}/\sqrt{3}\angle 120$ |

El nudo Slack tiene una tensión trifásica equilibrada con una tensión nominal de 20 kV, equivalente a la de los ejemplos 3 y 4 del Capítulo 4.

Una vez definida la tensión del nudo de cabecera, vamos a definir la topología de la red, siguiendo el esquema de la Figura 6.1. Para modelar las líneas de la red, se utiliza la impedancia por fase la cual depende del tipo de conductor. Por lo tanto, se van a utilizar como referencia las matrices de la tabla 7.25 del documento del CIGRE, donde se especifican las impedancias por fase de varios tipos de conductores. Estos valores se representan en la Figura 6.2 y sus unidades son $\Omega/km$:

| Conductor ID/ Installation | | The primitive impedance matrix [Ω/km] | | | |
|---|---|---|---|---|---|
| | | A | B | C | N |
| UG1 / 3-ph | A | 0.211 + j0.747 | 0.049 + j0.673 | 0.049 + j0.651 | 0.049 + j0.673 |
| | B | 0.049 + j0.673 | 0.211 + j0.747 | 0.049 + j0.673 | 0.049 + j0.651 |
| | C | 0.049 + j0.651 | 0.049 + j0.673 | 0.211 + j0.747 | 0.049 + j0.673 |
| | N | 0.049 + j0.673 | 0.049 + j0.651 | 0.049 + j0.673 | 0.211 + j0.747 |
| UG2 / 3-ph | A | 0.314 + j0.762 | 0.049 + j0.687 | 0.049 + j0.665 | 0.049 + j0.687 |
| | B | 0.049 + j0.687 | 0.314 + j0.762 | 0.049 + j0.687 | 0.049 + j0.665 |
| | C | 0.049 + j0.665 | 0.049 + j0.687 | 0.314 + j0.762 | 0.049 + j0.687 |
| | N | 0.049 + j0.687 | 0.049 + j0.665 | 0.049 + j0.687 | 0.314 + j0.762 |
| UG3 / 3-ph | A | 0.871 + j0.797 | 0.049 + j0.719 | 0.049 + j0.697 | 0.049 + j0.719 |
| | B | 0.049 + j0.719 | 0.871 + j0.797 | 0.049 + j0.719 | 0.049 + j0.697 |
| | C | 0.049 + j0.697 | 0.049 + j0.719 | 0.871 + j0.797 | 0.049 + j0.719 |
| | N | 0.049 + j0.719 | 0.049 + j0.697 | 0.049 + j0.719 | 0.871 + j0.797 |

**Figura 6.2** Matrices de impedancia por fase para conductores subterráneos.

Además de las impedancias por fase en $\Omega/km$, hacen falta otros parámetros para modelar completamente las líneas de la red residencial. Estos parámetros son la longitud de cada línea y los nudos que conecta. Estos datos sumados al tipo de conductor para cada línea se definen en la Tabla 6.2:

**Tabla 6.2** Datos de las líneas de la red.

| Rama | Nudo inicio | Nudo fin | Distancia(m) | Tipo de conductor |
|---|---|---|---|---|
| 1 | R1 | R2 | 35 | UG1 |
| 2 | R2 | R3 | 35 | UG1 |
| 3 | R3 | R4 | 35 | UG1 |
| 4 | R4 | R5 | 35 | UG1 |
| 5 | R5 | R6 | 35 | UG1 |
| 6 | R6 | R7 | 35 | UG1 |
| 7 | R7 | R8 | 35 | UG1 |
| 8 | R8 | R9 | 35 | UG1 |
| 9 | R9 | R10 | 35 | UG1 |
| 10 | R3 | R11 | 30 | UG3 |
| 11 | R4 | R12 | 35 | UG3 |
| 12 | R12 | R13 | 35 | UG3 |
| 13 | R13 | R14 | 35 | UG3 |
| 14 | R14 | R15 | 30 | UG3 |
| 15 | R6 | R16 | 30 | UG3 |
| 16 | R9 | R17 | 30 | UG3 |
| 17 | R10 | R18 | 30 | UG3 |

Una vez definidas las líneas de la red, vamos a introducir los datos de las cargas. En este caso, existen varias cargas modeladas como cargas de potencia constante en varios nudos. Estas cargas son monofásicas conectadas en estrella. Se definen con la potencia aparente total trifásica, el factor de potencia y el desequilibrio entre fases.

Para la red de nuestro modelo, las cargas están detalladas en la Tabla 6.3 para el caso de perfil de cargas equilibrado y en la Tabla 6.4 para el caso de perfil de cargas desequilibrado. Los parámetros de la cargas se han obtenido de [15].Se simularán ambos perfiles de carga:

- **Perfil de cargas equilibrado:**

**Tabla 6.3** Datos de cargas para el caso de perfil equilibrado.

| **Bus** | **R2** | **R11** | **R15** | **R16** | **R17** | **R18** |
|---|---|---|---|---|---|---|
| S[$kVA$] | -12 | -55 | -25 | -100 | -44 | -10 |
| cos $\varphi$ | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 |
| Ph.a (%) | 33.3 | 33.3 | 33.3 | 33.3 | 33.3 | 33.3 |
| Ph.b (%) | 33.3 | 33.3 | 33.3 | 33.3 | 33.3 | 33.3 |
| Ph.c (%) | 33.3 | 33.3 | 33.3 | 33.3 | 33.3 | 33.3 |

- **Perfil de cargas desequilibrado:**

**Tabla 6.4** Datos de cargas para el caso de perfil desequilibrado.

| **Bus** | **R2** | **R11** | **R15** | **R16** | **R17** | **R18** |
|---|---|---|---|---|---|---|
| S[$kVA$] | -12 | -55 | -25 | -100 | -44 | -10 |
| cos $\varphi$ | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 |
| Ph.a (%) | 20 | 30 | 40 | 40 | 35 | 40 |
| Ph.b (%) | 20 | 20 | 30 | 10 | 25 | 20 |
| Ph.c (%) | 60 | 50 | 30 | 50 | 40 | 40 |

La red esta formada por una parte en baja tensión a 400 V y una parte en media tensión a 20 kV. La conexión entre las dos partes se realiza mediante un transformador trifásico de configuración $\triangle$ - Yg.

Los parámetros de este transformador se definen en la Tabla 6.5:

**Tabla 6.5** Parámetros del transformador de cabecera.

| Primario | Secundario | Conexión | $V_1$[kV] | $V_2$[kV] | $Y_{tr}$[S] | S[$kVA$] | $R_g[\Omega]$ |
|---|---|---|---|---|---|---|---|
| R0 | R1 | 3ph Dy11 | 20 | 0.4 | 0.0024-0.0098j | 500 | 3 |

Se simulará también la incorporación de generación distribuida en la red, que se simularán como nudos PV , con una consigna de tensión nominal de 400 V y una potencia generada de 100 kW.

Por último se definen las resistencias de puesta a tierra de los nudos, que serán de 40 Ω.

## 6.2 Perfil de cargas equilibrado

En primer lugar se llevará a cabo una simulación para un perfil de cargas equilibrado, definido en la Tabla 6.3. Se realizarán dos casos de estudio: un primer caso para una red con solo nudos de consumo y una segunda red con generación distribuida y se compararán los resultados obtenidos.

### 6.2.1 Caso de estudio 1: Red únicamente con nudos de consumo

Las características de la red se toman de la Sección 6.1. Los datos se introducen en el programa utilizando las herramientas descritas en el Capítulo 4.

Los resultados de la simulación se muestran en la Figura 6.3 y en la Tabla 6.6:

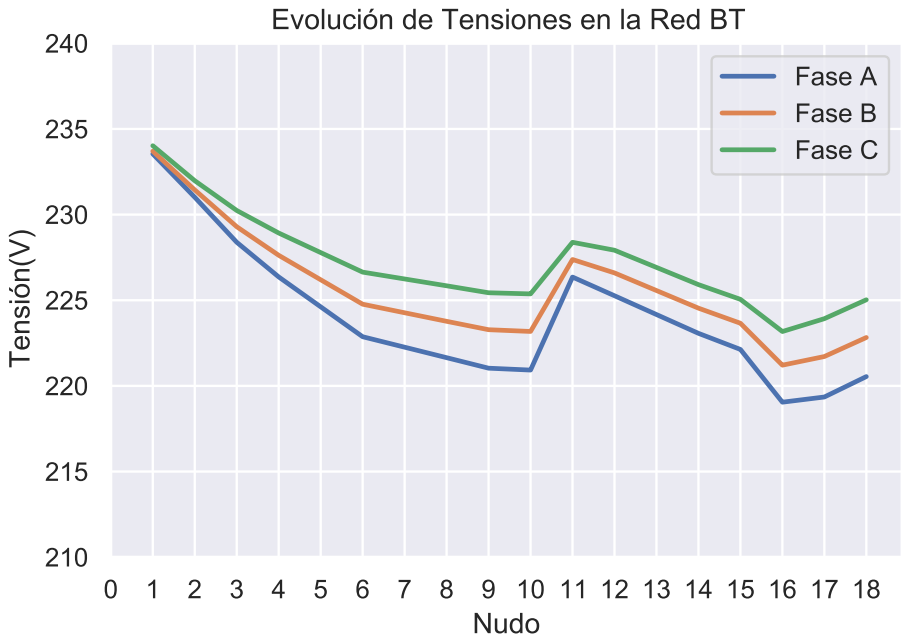


**Figura 6.3** Evolución de las tensiones de cada fase para un perfil equilibrado únicamente con nudos de consumo.

En la Figura 6.3 podemos observar la evolución del perfil de tensiones para el primer caso de simulación.Se observan diferencias entre las tensiones de cada fase. Esto es debido principalmente a que el tipo de conductor utilizado (*UG1* y *UG3*) no tienen una impedancia perfectamente simétrica entre fases, lo que origina diferentes caídas de tensión en la red, incluso para un perfil de cargas equilibrado. Por otro lado, se ha definido el perfil de tensiones de la red en la Tabla 6.6:

**Tabla 6.6** Valor absoluto y ángulo de las tensiones de fase para un perfil equilibrado únicamente con nudos de consumo.

| NUDO | $U_a$ | $\varphi_a$ | $U_b$ | $\varphi_b$ | $U_c$ | $\varphi_c$ | $U_n$ | $\varphi_n$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 11547 | 0 | 11547 | -120 | 11547 | 120 | 0 | 0 |
| 1 | 233.5433 | 31.1801 | 233.7027 | -88.9548 | 234.0218 | 151.1327 | 0.3006 | 168.6383 |
| 2 | 231.0124 | 31.1062 | 231.4409 | -89.0137 | 231.979 | 151.0176 | 0.0001 | 97.0996 |
| 3 | 228.3854 | 31.0633 | 229.2932 | -89.1299 | 230.2414 | 150.9382 | 0.0134 | -73.1092 |
| 4 | 226.3613 | 31.0348 | 227.6247 | -89.2232 | 228.9249 | 150.8741 | 0 | -144.872 |
| 5 | 224.6159 | 31.0075 | 226.1967 | -89.3035 | 227.7833 | 150.8199 | 0 | -147.235 |
| 6 | 222.8705 | 30.9797 | 224.7691 | -89.3848 | 226.6418 | 150.7652 | 0 | -148.208 |
| 7 | 222.2592 | 30.9701 | 224.2696 | -89.4132 | 226.2428 | 150.7462 | 0 | -149.712 |
| 8 | 221.6479 | 30.9605 | 223.7701 | -89.4418 | 225.8439 | 150.7271 | 0 | -158.676 |
| 9 | 221.0341 | 30.9482 | 223.2804 | -89.4697 | 225.4374 | 150.7098 | 0.012 | -84.4229 |
| 10 | 220.9239 | 30.9491 | 223.1786 | -89.4758 | 225.3713 | 150.7045 | 0 | -155.177 |
| 11 | 226.3541 | 31.1726 | 227.3793 | -89.0305 | 228.384 | 151.045 | 0 | -64.7044 |
| 12 | 225.2637 | 31.0925 | 226.596 | -89.1693 | 227.9208 | 150.9316 | 0 | -129.389 |
| 13 | 224.1663 | 31.1508 | 225.5674 | -89.1149 | 226.917 | 150.9896 | 0 | -104.063 |
| 14 | 223.0691 | 31.2097 | 224.539 | -89.0599 | 225.9134 | 151.0481 | 0 | -73.771 |
| 15 | 222.1288 | 31.2606 | 223.6577 | -89.0124 | 225.0534 | 151.0987 | 0 | -53.7455 |
| 16 | 219.0486 | 31.1858 | 221.2101 | -89.1942 | 223.174 | 150.9705 | 0 | -52.4052 |
| 17 | 219.3519 | 31.0413 | 221.7145 | -89.3871 | 223.9245 | 150.8002 | 0 | -62.0389 |
| 18 | 220.5443 | 30.9698 | 222.8255 | -89.4567 | 225.0276 | 150.725 | 0 | -52.414 |

El transformador introduce un desfase de 30º entre primario y secundario, característica del grupo de conexión Dy11. Las tensiones van decreciendo según nos vamos alejando del nudo de cabecera, perfil habitual de redes de baja tensión. En la gráfica de tensiones se oberva un "salto" a partir del nudo 11, debido a que la rama de nudos más alejada del nudo de cabecera termina en el nudo 10. Las tensiones no caen por debajo de la caída de tensión máxima admisible para redes de baja tensión, que es de +/- 7 % [16].

Respecto a los ángulos, vemos como efectivamente hay una diferencia de aproximadamente +30º entre el nudo de cabecera y la red de baja tensión. A diferencia de los modelos de validación, en este caso las cargas tienen un factor de potencia de 0.95, lo que provoca que los ángulos sean mayores de 30º. Por otro lado, según nos alejamos del nudo 'Slack', vemos como el ángulo disminuye, por efecto de la impedancia de las líneas.

En cuánto a los módulos de las tensiones, se observan diferencias entre las fases como se ha comentado previamente. Los módulos de las tensiones disminuyen proporcionalmente a la distancia del nudo con el nudo de cabecera.

### 6.2.2 Red con generación distribuida

Una vez realizada la primera simulación, se introducen dos generadores en los nudos R8 y R13, cuyos parámetros se definieron anteriormente. El perfil de tensiones se representa en la Figura 6.4:

En este caso vemos que el perfil de tensiones es mucho más plano que en el caso anterior. Los generadores ayudan a "levantar" las tensiones. Esto es debido a que reducen la corriente proveniente del nudo de cabecera y fijan las tensiones en su nudo. Por otro lado, vemos que esto provoca que las diferencias entre las tensiones de cada fase sean más pequeñas que en el caso anterior.En la Tabla 6.7 se puede observar como el algoritmo fija la tensión especificada en los nudos PV.

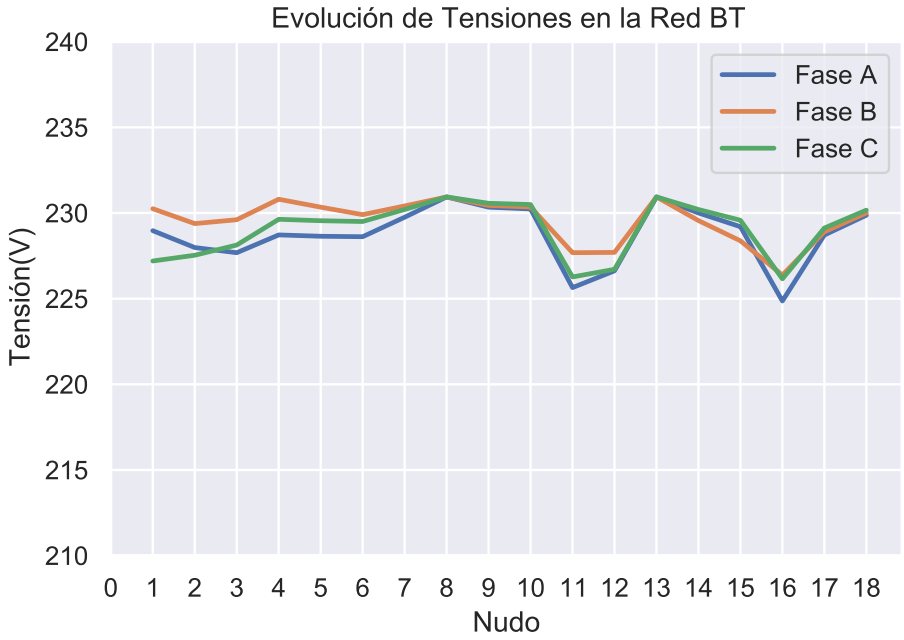


**Figura 6.4** Evolución de las tensiones de cada fase para un perfil equilibrado con generación distribuida.

**Tabla 6.7** Valor absoluto y ángulo de las tensiones de fase para un perfil equilibrado con generación distribuida.

| NUDO | $U_a$ | $\varphi_a$ | $U_b$ | $\varphi_b$ | $U_c$ | $\varphi_c$ | $U_n$ | $\varphi_n$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 11547 | 0 | 11547 | -120 | 11547 | 120 | 0 | 0 |
| 1 | 228.7864 | 33.2511 | 229.668 | -86.8303 | 227.5155 | 153.6447 | 0.5094 | -97.8792 |
| 2 | 228.3468 | 30.742 | 229.277 | -89.0091 | 228.0551 | 151.3992 | 0.0002 | -170.252 |
| 3 | 228.2925 | 28.1284 | 229.7591 | -91.1757 | 228.7729 | 149.2653 | 0.0082 | -73.1039 |
| 4 | 229.3111 | 25.5377 | 231.0553 | -93.2997 | 230.2158 | 147.1705 | 0.0004 | -6.4087 |
| 5 | 229.1654 | 25.4215 | 230.5645 | -93.3489 | 230.0341 | 147.0048 | 0.0002 | -6.7693 |
| 6 | 229.0206 | 25.305 | 230.0738 | -93.3984 | 229.8542 | 146.8389 | 0 | -175.097 |
| 7 | 229.98 | 25.2032 | 230.507 | -93.3946 | 230.3966 | 146.7115 | 0.0001 | 115.7368 |
| 8 | 230.9401 | 25.1022 | 230.9401 | -93.3909 | 230.9401 | 146.5847 | 0.0001 | 110.9687 |
| 9 | 230.3507 | 25.0946 | 230.4494 | -93.4194 | 230.555 | 146.5654 | 0.0042 | -117.678 |
| 10 | 230.2448 | 25.094 | 230.3548 | -93.4244 | 230.4853 | 146.5609 | 0 | 171.6948 |
| 11 | 226.2622 | 28.2408 | 227.8363 | -91.078 | 226.9131 | 149.371 | 0 | -84.5374 |
| 12 | 226.676 | 15.569 | 227.6854 | -103.011 | 226.8779 | 136.8756 | 0.4782 | -165.978 |
| 13 | 230.9401 | 5.6713 | 230.94 | -112.728 | 230.9402 | 126.6135 | 0.9572 | -165.965 |
| 14 | 230.0623 | 5.7902 | 229.6276 | -112.666 | 230.0813 | 126.5978 | 0.4417 | -165.989 |
| 15 | 229.3108 | 5.8927 | 228.5031 | -112.612 | 229.3449 | 126.5842 | 0.0001 | 157.0975 |
| 16 | 225.2978 | 25.5119 | 226.5562 | -93.2246 | 226.4821 | 147.0323 | 0 | -88.2809 |
| 17 | 228.7368 | 25.1848 | 228.9143 | -93.3444 | 229.0924 | 146.6489 | 0 | -93.0706 |
| 18 | 229.8798 | 25.1143 | 230.0081 | -93.4074 | 230.154 | 146.5798 | 0 | -90.3378 |

## 6.3 Perfil de cargas desequilibrado

Una vez evaluados los casos para perfiles de cargas equilibradas, se modificará el perfil de consumo siguiendo la Tabla 6.4. Al igual que en el caso anterior, primero se simulará una red únicamente con nudos de consumo y se comparará con el caso con generación distribuida.

### 6.3.1 Caso de estudio 3: Red únicamente con nudos de consumo

El modelo de red es el mismo que para el caso de estudio 1, excepto por el perfil de consumo. En este caso se toma un perfil de cargas desequilibrado, definido en la Tabla 6.4.El perfil de tensiones resultante se representa en la Figura 6.5:

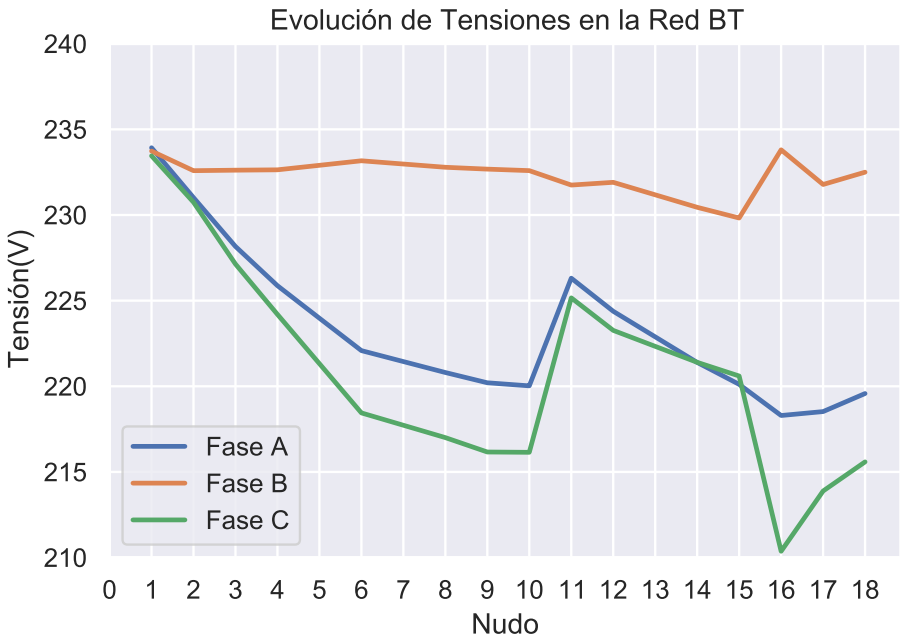


**Figura 6.5** Evolución de las tensiones de cada fase para un perfil desequilibrado únicamente con nudos de consumo.

El alto grado de desequilibrio produce grandes diferencias entre fases. Si calculamos el porcentaje por fase a partir de la Tabla 6.4 obtenemos la Tabla

**Tabla 6.8** Porcentajes globales de carga de cada fase.

| | |
|---|---|
| Ph.a(%) Total | 35.35 |
| Ph.b(%) Total | 21.55 |
| Ph.c(%) Total | 43.10 |

Vemos que la fase B es la que menor porcentaje de carga global tiene de las tres fases, menos de la mitad que la fase C, y es por tanto la fase con menor caída de tensión. Las fases A y C tienen una mayor caída de tensión, correspondiéndose el porcentaje de carga total con la caída de tensión.

En este ejemplo las tensiones quedan por debajo de la tensión mínima admisible.

**Tabla 6.9** Valor absoluto y ángulo de las tensiones de fase para un perfil desequilibrado únicamente con nudos de consumo.

| NUDO | $U_a$ | $\varphi_a$ | $U_b$ | $\varphi_b$ | $U_c$ | $\varphi_c$ | $U_n$ | $\varphi_n$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 11547 | 0 | 11547 | -120 | 11547 | 120 | 0 | 0 |
| 1 | 233.9269 | 30.8653 | 233.7319 | -89.2876 | 233.4534 | 151.7084 | 1.4405 | -68.8134 |
| 2 | 231.0161 | 30.7946 | 232.5872 | -89.2921 | 230.7446 | 151.576 | 0.0005 | -146.628 |
| 3 | 228.1629 | 30.4015 | 232.6162 | -89.1517 | 227.135 | 151.6211 | 0.1245 | -71.1306 |
| 4 | 225.8732 | 30.0982 | 232.6369 | -89.0358 | 224.1971 | 151.6378 | 0.0001 | -171.583 |
| 5 | 223.9744 | 29.7737 | 232.9002 | -88.8874 | 221.3216 | 151.6647 | 0.0004 | -126.867 |
| 6 | 222.0829 | 29.4437 | 233.1651 | -88.7392 | 218.4462 | 151.6923 | 0.0008 | -122.001 |
| 7 | 221.4433 | 29.3783 | 232.9754 | -88.7257 | 217.7251 | 151.6876 | 0.0004 | -122.371 |
| 8 | 220.8041 | 29.3125 | 232.7858 | -88.7121 | 217.0039 | 151.6828 | 0.0001 | -127.718 |
| 9 | 220.2049 | 29.2161 | 232.6777 | -88.6772 | 216.1632 | 151.6857 | 0.1413 | -53.3717 |
| 10 | 220.025 | 29.2303 | 232.5899 | -88.6961 | 216.1454 | 151.6812 | 0.0001 | -132.107 |
| 11 | 226.3117 | 30.4239 | 231.7469 | -89.0579 | 225.1622 | 151.7615 | 0.0001 | 26.1767 |
| 12 | 224.3796 | 30.1415 | 231.9075 | -89.0105 | 223.2636 | 151.7408 | 0.0001 | -158.211 |
| 13 | 222.8862 | 30.1853 | 231.1781 | -88.9849 | 222.3309 | 151.8446 | 0 | -101.008 |
| 14 | 221.3929 | 30.2297 | 230.4488 | -88.9593 | 221.399 | 151.9493 | 0.0001 | -38.7063 |
| 15 | 220.113 | 30.2683 | 229.8237 | -88.9371 | 220.6007 | 152.0397 | 0.0001 | -25.3144 |
| 16 | 218.2941 | 29.0853 | 233.8049 | -88.2306 | 210.3682 | 152.1714 | 0.0011 | 51.6292 |
| 17 | 218.5236 | 29.2269 | 231.7812 | -88.5564 | 213.8856 | 151.8263 | 0.0002 | 45.6998 |
| 18 | 219.5773 | 29.2198 | 232.4995 | -88.667 | 215.587 | 151.7242 | 0.0001 | 34.8683 |

Algo que se puede hacer para evitar que haya tensiones por debajo del mínimo es modificar la relación de transformación. Si se aumenta un 2 % se obtiene el perfil de tensiones mostrado en la Figura 6.6:

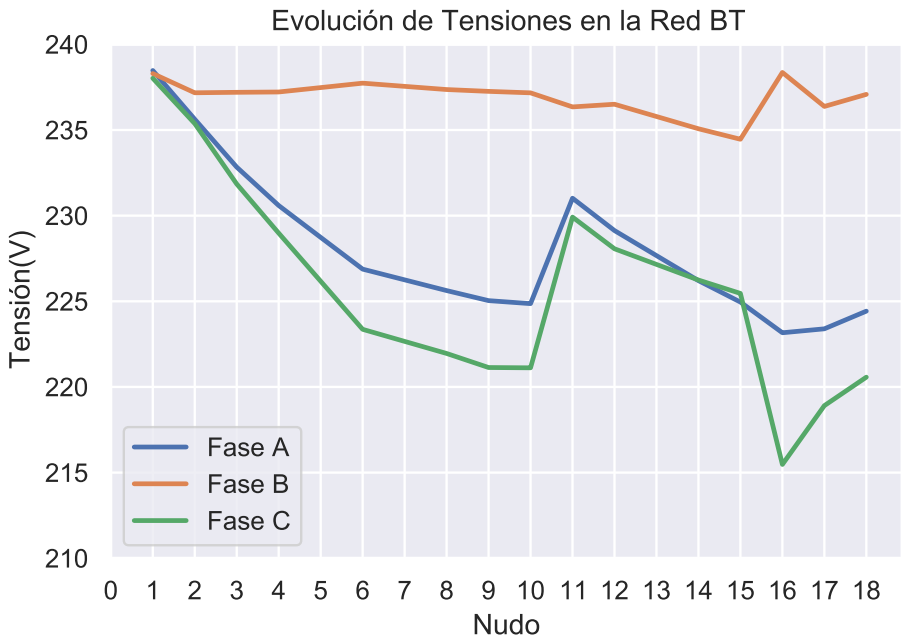


**Figura 6.6** Perfil de tensiones con un aumento de la relación de transformación de un 2 %.

Por lo que la mínima tensión de la red estaría dentro de los valores admisibles.

### 6.3.2 Caso de estudio 4: Red con generación distribuida

Por último, se simula el caso de estudio con cargas desequilibradas y generación distribuida. Se toma las cargas definidas en la Tabla 6.4 y se definen dos generadores en los nudos R8 y R13 de la misma manera que en el caso de estudio 2. El perfil de tensiones resultante se representa en la Figura 6.7:

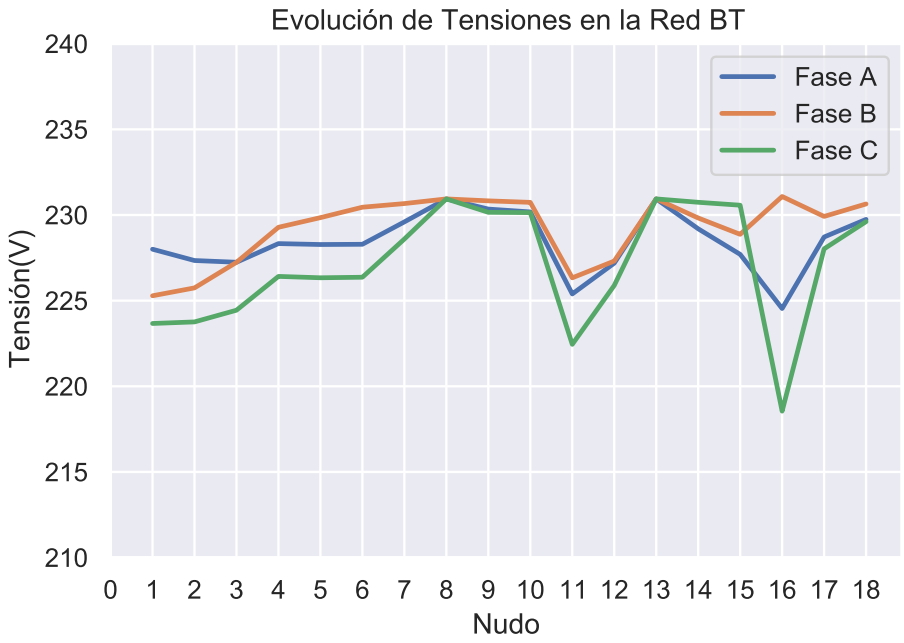


**Figura 6.7** Evolución de las tensiones de cada fase para un perfil desequilibrado con generación distribuida..

**Tabla 6.10** Valor absoluto y ángulo de las tensiones de fase para un perfil desequilibrado con generación distribuida..

| NUDO | $U_a$ | $\varphi_a$ | $U_b$ | $\varphi_b$ | $U_c$ | $\varphi_c$ | $U_n$ | $\varphi_n$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 11547 | 0 | 11547 | -120 | 11547 | 120 | 0 | 0 |
| 1 | 228.0021 | 33.8537 | 225.284 | -86.3136 | 223.6661 | 155.0372 | 0.4115 | -84.6815 |
| 2 | 227.3419 | 30.8629 | 225.7445 | -89.159 | 223.7581 | 151.9326 | 0.0002 | -169.202 |
| 3 | 227.241 | 27.7688 | 227.2374 | -91.9711 | 224.4411 | 148.9341 | 0.1224 | -72.1523 |
| 4 | 228.3298 | 24.7846 | 229.2826 | -94.7656 | 226.415 | 145.9443 | 0.0003 | 16.6696 |
| 5 | 228.2761 | 23.8002 | 229.8456 | -95.5727 | 226.3348 | 144.6703 | 0.0003 | -116.001 |
| 6 | 228.2899 | 22.8155 | 230.4541 | -96.3758 | 226.3664 | 143.3961 | 0.0008 | -130.496 |
| 7 | 229.5968 | 22.0896 | 230.6667 | -97.3065 | 228.596 | 142.1008 | 0.0006 | -144.154 |
| 8 | 230.9401 | 21.3721 | 230.9401 | -98.2352 | 230.9401 | 140.8312 | 0.0005 | -166.953 |
| 9 | 230.3456 | 21.2853 | 230.8252 | -98.2054 | 230.1561 | 140.8362 | 0.1334 | -65.4552 |
| 10 | 230.1794 | 21.2987 | 230.7373 | -98.2229 | 230.1392 | 140.8321 | 0.0001 | -143.778 |
| 11 | 225.39 | 27.7931 | 226.3312 | -91.8722 | 222.4472 | 149.0749 | 0.0001 | 24.9246 |
| 12 | 227.187 | 16.3636 | 227.3103 | -103.747 | 225.8859 | 136.892 | 1.0589 | -99.479 |
| 13 | 230.94 | 8.0383 | 230.9401 | -112.663 | 230.9402 | 128.0184 | 2.1188 | -99.4655 |
| 14 | 229.1948 | 8.2364 | 229.8256 | -112.775 | 230.7416 | 128.1019 | 0.9777 | -99.4878 |
| 15 | 227.7017 | 8.4085 | 228.871 | -112.871 | 230.5713 | 128.1734 | 0.0003 | -125.491 |
| 16 | 224.5386 | 22.4652 | 231.0807 | -95.8745 | 218.5472 | 143.8466 | 0.0011 | 43.117 |
| 17 | 228.7139 | 21.2958 | 229.9156 | -98.0895 | 228.0238 | 140.9625 | 0.0001 | 33.2692 |
| 18 | 229.7441 | 21.2886 | 230.6443 | -98.1947 | 229.6116 | 140.8709 | 0.0001 | 23.7208 |

Se puede observar que el perfil de tensiones es más plano y mejora considerablemente respecto del caso sin generación distribuida. El perfil de cargas desequilibrado sigue produciendo diferencias entre las tensiones de fase, pero estas diferencias son menores.

Igual que en el caso de estudio 3, se observa como las caídas de tensión de cada fase están relacionadas con el porcentaje total de carga, según la Tabla 6.4. Por otro lado, se observa como nuevamente el algoritmo fija las tensiones de los nudos PV, ajustándolas a la tensión especificada.

# 7 Conclusiones y trabajo futuro

Este trabajo ha consistido en la implementación de una herramienta de programación en el lenguaje Python para la resolución del problema del flujo de cargas aplicado a redes trifásicas desequilibradas de cuatro hilos.

Este documento se puede considerar como la continuación natural al trabajo realizado en [9] y [11]. En [9] se desarrolló la formulación detallada en [5] y se implementó en el entorno *MATLAB* una herramienta matemática que permitía ejecutar el método para redes desequilibradas. En [11] se añadió el modelo del transformador para diferentes grupos de conexión y se desarrolló la formulación para incorporar puestas de tierra al modelo, también usando *MATLAB*.

En esta línea, a lo largo de este trabajo se han desarrollado los cálculos teóricos que permite poner en práctica modelos de red con nudos PV, incorporando transformadores y puestas de tierra al modelo.

Por otro lado, se ha partido de cero programando completamente en Python. La herramienta obtenida se ha validado satisfactoriamente contrastando los resultados obtenidos con modelos equivalentes en el programa OpenDSS.

Por último, se ha demostrado la utilidad de la herramienta para simular redes de media y baja tensión, tanto para perfiles de carga equilibrados como desequilibrados.

Este trabajo es el primero de dos trabajos donde se busca comparar dos algoritmos que resuelven el problema de flujo de cargas en redes desequilibradas a cuatro hilos. El siguiente paso será comparar el algoritmo implementado en este trabajo con el método descrito en [17], que será objeto de un segundo trabajo.

# Apéndice A
# Código completo Python

```
# ------------------------------------------ METODO RECTANGULAR
    ------------------------------------

import numpy as np

#Datos de matriz de impedancia de líneas

line_type = {'OH1': np.array ([[0.540 + 0.777j, 0.049 + 0.505j, 0.049 + 0.462j, 0.049 + 0.436j ],
                               [0.049 + 0.505j, 0.540 + 0.777j, 0.049 + 0.505j, 0.049 + 0.462j ],
                               [0.049 + 0.462j, 0.049 + 0.505j, 0.540 + 0.777j, 0.049 + 0.505j ],
                               [0.049 + 0.436j, 0.049 + 0.462j, 0.049 + 0.505j, 0.540 + 0.777j ]]) ,
           'OH2': np.array ([[1.369 + 0.812j, 0.049 + 0.505j, 0.049 + 0.462j, 0.049 + 0.436j ],
                               [0.049 + 0.505j, 1.369 + 0.812j, 0.049 + 0.505j, 0.049 + 0.462j ],
                               [0.049 + 0.462j, 0.049 + 0.505j, 1.369 + 0.812j, 0.049 + 0.505j ],
                               [0.049 + 0.436j, 0.049 + 0.462j, 0.049 + 0.505j, 1.369 + 0.812j ]]) ,
           'OH3': np.array ([[2.065 + 0.825j, 0.049 + 0.505j, 0.049 + 0.462j, 0.049 + 0.436j ],
                               [0.049 + 0.505j, 2.065 + 0.825j, 0.049 + 0.505j, 0.049 + 0.462j ],
                               [0.049 + 0.462j, 0.049 + 0.505j, 2.065 + 0.825j, 0.049 + 0.505j ],
                               [0.049 + 0.436j, 0.049 + 0.462j, 0.049 + 0.505j, 2.065 + 0.825j ]]) ,
           'UG1': np.array ([[0.211 + 0.747j, 0.049 + 0.673j, 0.049 + 0.651j, 0.049 + 0.673j ],
                               [0.049 + 0.673j, 0.211 + 0.747j, 0.049 + 0.673j, 0.049 + 0.651j ],
                               [0.049 + 0.651j, 0.049 + 0.673j, 0.211 + 0.747j, 0.049 + 0.673j ],
                               [0.049 + 0.673j, 0.049 + 0.651j, 0.049 + 0.673j, 0.211 + 0.747j ]]) ,
           'UG2': np.array ([[0.314 + 0.762j, 0.049 + 0.687j, 0.049 + 0.665j, 0.049 + 0.687j ],
                               [0.049 + 0.687j, 0.314 + 0.762j, 0.049 + 0.687j, 0.049 + 0.665j ],
                               [0.049 + 0.665j, 0.049 + 0.687j, 0.314 + 0.762j, 0.049 + 0.687j ],
                               [0.049 + 0.687j, 0.049 + 0.665j, 0.049 + 0.687j, 0.314 + 0.762j ]]) ,
           'UG3': np.array ([[0.871 + 0.797j, 0.049 + 0.719j, 0.049 + 0.697j, 0.049 + 0.719j ],
                               [0.049 + 0.719j, 0.871 + 0.797j, 0.049 + 0.719j, 0.049 + 0.697j ],
                               [0.049 + 0.697j, 0.049 + 0.719j, 0.871 + 0.797j, 0.049 + 0.719j ],
                               [0.049 + 0.719j, 0.049 + 0.697j, 0.049 + 0.719j, 0.871 + 0.797j ]]) }

#Defino la clase line. Argumentos: Impedancia, Longitud, bus entrada, bus salida

class line:
    def __init__(self, Z, long, BUS_0, BUS_1):
        self.Z = line_type[Z]*long/1000
        self.long = long
        self.I = None
        self.connections = [BUS_0, BUS_1]

        #La función check permite comprobar la caída de tensión en cada línea.
    def check(self):
        res = (np.array(self.connections[0].U) -
        np.array(self.connections[1].U)) - np.dot(self.Z, self.I)
        return res

#Defino la clase bus. Argumentos: ref, potencia activa, reactiva, tensión, tipo
#Nudos PQ: U = None, Nudos PV,
#Nudos Slack: P,Q = None U = U_ref
```

```
#Nudos PV: P, V_esp

class bus:
    def __init__(self, ref, P, Q, U, tipo, Rg, V_esp=False):
        self.ref = ref
        self.P = P
        self.Q = Q
        self.U = U
        self.V_esp = V_esp
        self.tipo = tipo
        self.rg = Rg
        self.I = [0, 0, 0, 0]
        self.connections = list()

      #La función check permite comprobar la PLK en cada nudo.

    def check(self):
        sum_I = np.array([0, 0, 0, 0], dtype=complex)
        sum_I += np.array(self.I)
        for line in self.connections:
            if line.connections[0] == self:
                sum_I -= np.array(line.I)
            if line.connections[1] == self:
                sum_I += np.array(line.I)
        if self.rg != False:
            sum_I[3] -= self.U[3]/self.rg
        res_P = np.array([0, 0, 0])
        res_Q = np.array([0, 0, 0])
        for phase in range(3):
            S = (self.U[phase])*np.conjugate(self.I[phase])
            res_P[phase] = np.real(S) - self.P[phase]
            res_Q[phase] = np.imag(S) - self.Q[phase]
        return sum_I, res_P, res_Q

class transformer:
    def __init__(self, BUS_0, BUS_1, Y_t, r_a, r_b, r_c, R_gt):


        #Submatrices de admitancia del modelo del transformador

        self.Y_pp = np.array([[2*Y_t, -Y_t,   -Y_t,   0],
                              [-Y_t, 2*Y_t,   -Y_t,   0],
                              [-Y_t,  -Y_t, 2*Y_t,    0],
                              [0,       0,      0    ,0]])

        self.Y_ps = np.array([[-r_a*Y_t,      0,        r_c*Y_t,    (r_a-r_c)*Y_t],
                              [r_a*Y_t,    -r_b*Y_t,      0     ,  (r_b-r_a)*Y_t],
                              [   0,       r_b*Y_t,   -r_c*Y_t,    (r_c-r_b)*Y_t],
                              [   0,         0,          0,             0       ]])


        self.Y_sp = np.array([[-r_a*Y_t,          r_a*Y_t,          0,          0],
                              [   0,           -r_b*Y_t,          r_b*Y_t,    0],
                              [r_c*Y_t,           0,           -r_c*Y_t,      0],
                              [(r_a-r_c)*Y_t, (r_b-r_a)*Y_t, (r_c-r_b)*Y_t,  0]])

        self.Y_ss = np.array([[ r_a**2*Y_t,          0,              0,                  -r_a**2*Y_t],
                              [   0,           r_b**2*Y_t,          0,                  -r_b**2*Y_t],
                              [   0,             0,           r_c**2*Y_t,               -r_c**2*Y_t],
                              [-r_a**2*Y_t, -r_b**2*Y_t,    -r_c**2*Y_t,  (r_a**2+r_b**2+r_c**2)*Y_t+1/R_gt]])

        self.I = [0,0,0,0]
        self.connections = [BUS_0, BUS_1]

        self.Y_trafo = np.zeros((8,8), dtype=complex)
        self.Y_trafo[0:4,0:4] = self.Y_pp
        self.Y_trafo[0:4,4:8] = -self.Y_ps
        self.Y_trafo[4:8,0:4] = self.Y_sp
        self.Y_trafo[4:8,4:8] = self.Y_ss
```

```
class grid:
    def __init__(self):
        #Buses y líneas se inicializan con una lista vacía
        self.buses = list()
        self.lines = list()
        self.transformers = list()

    def add_bus(self, ref, P, Q, U, tipo, Rg, V_esp=False):

        self.buses.append(bus(ref, P, Q, U, tipo, Rg, V_esp))

    def add_line(self, Z, long, bus_0, bus_1):
        for bus in self.buses:
            if bus.ref == bus_0:
                BUS_0 = bus
            if bus.ref == bus_1:
                BUS_1 = bus
        self.lines.append(line(Z, long, BUS_0, BUS_1))
        BUS_0.connections.append(self.lines[-1])
        BUS_1.connections.append(self.lines[-1])

    def add_transformer(self, bus_0, bus_1, Y_t, r_a, r_b, r_c, R_gt):
        for bus in self.buses:
            if bus.ref == bus_0:
                BUS_0 = bus
            if bus.ref == bus_1:
                BUS_1 = bus
        self.transformers.append(transformer(BUS_0, BUS_1, Y_t,
        r_a, r_b, r_c, R_gt))

        BUS_0.connections.append(self.transformers[-1])
        BUS_1.connections.append(self.transformers[-1])

    def initialize(self,U_0):

        #Recorre todos los buses excepto el nudo slack y copia la tensión inicial
        for bus in self.buses[1:]:
                if bus.U == None:
                    bus.U = np.copy(U_0)

                for index in range(3):
                    bus.I[index] = np.complex(bus.P[index],
                    -bus.Q[index])/np.conjugate(bus.U[index]
                    - bus.U[3])
                bus.I[3] = -np.sum(bus.I)

    def generate_Y(self):

        self.Y = np.zeros((len(self.buses)*4, len(self.buses)*4),
        dtype = complex)

        #Bucle para colocar los términod Y_ij

        for bus in self.buses:
            for line in bus.connections:
                for line_con in line.connections:
                        if bus.ref != line_con.ref and
                        hasattr(line, 'Z') == True:
                            self.Y[bus.ref*4 : bus.ref*4 + 4,
                            line_con.ref*4 : line_con.ref*4 + 4]
                            += -np.linalg.inv(line.Z)

        #Ahora calculamos los elementos de la diagonal principal Y_ii

        for index in range(0, self.Y.shape[0], 4):
            aux = np.zeros([4, 4], dtype = complex)

            for index2 in range(0, self.Y.shape[1], 4):
                aux -= self.Y[index : index + 4, index2 : index2 + 4]

        #Rellena la diagonal principal con esos elementos
```

```
            self.Y[index : index + 4, index : index + 4] = aux


        #Términos de admitancias del transformador
        if isinstance(self.buses[0].connections[0], transformer):
            self.Y[:4,:4] += self.transformers[0].Y_pp
            self.Y[:4,4:8] += self.transformers[0].Y_ps
            self.Y[4:8,:4] -= self.transformers[0].Y_sp
            self.Y[4:8,4:8] -= self.transformers[0].Y_ss


        # Ahora se separan los terminos real e imaginario,
        Bij Gij, el imaginario delante
        self.Y_B = np.zeros((self.Y.shape[0]*2, self.Y.shape[1]*2),
        dtype = complex)
        for index_row, row in enumerate(self.Y):
            for index_col, item in enumerate(row):

                self.Y_B[(index_row)*2, (index_col)*2] = -np.imag(item)
                self.Y_B[(index_row)*2 + 1, (index_col)*2 + 1] = np.imag(item)
                self.Y_B[(index_row)*2, (index_col)*2 + 1] = np.real(item)
                self.Y_B[(index_row)*2 + 1, (index_col)*2] = np.real(item)

        #Se coloca el término de la puesta a tierra
        for bus in self.buses:
            if bus.rg != False:

                self.Y_B[bus.ref*8 + 6, bus.ref*8 + 7] += 1/bus.rg
                self.Y_B[bus.ref*8 + 7, bus.ref*8 + 6] += 1/bus.rg

        #Se elimina la primera fila, nudo slack
        self.Y_B = self.Y_B[2*4:]

        return self.Y_B


    def generate_jacobian_matrices(self,epsilon_PV):
        #Genera las matrices que forman el jacobiano y varían en cada iteración: D_I,D_V

        self.D_I = np.zeros((self.Y.shape[0]*2, self.Y.shape[1]*2))
        self.D_V = np.zeros((self.Y.shape[0]*2, self.Y.shape[1]*2))


        #Definimos la variable epsilon que sera diferente para nudos PV y nudos PQ
        for bus in self.buses:
            if bus.tipo == 'PV':
                epsilon = epsilon_PV
            else:
                epsilon = 1

            D_I =[Ibi Iai ; Iai*eps+(1-eps)*2fi 2Iai Ibi*eps+(1-eps)*2*ei]
            for pointer, item in enumerate(bus.I[:-1]):

                self.D_I[(bus.ref)*8 + pointer*2, (bus.ref)*8
                + pointer*2] = np.imag(item)
                self.D_I[(bus.ref)*8 + pointer*2, (bus.ref)*8 +
                pointer*2 + 1] = np.real(item)
                self.D_I[(bus.ref)*8 + pointer*2 + 1, (bus.ref)*8
                + pointer*2] = np.real(item)*epsilon
                self.D_I[(bus.ref)*8 + pointer*2 + 1, (bus.ref)*8
                + pointer*2 + 1] = -np.imag(item)*epsilon

                self.D_I[(bus.ref)*8 + pointer*2, (bus.ref)*8 + 6]
                = -np.imag(item)
                self.D_I[(bus.ref)*8 + pointer*2, (bus.ref)*8 + 7]
                = -np.real(item)
                self.D_I[(bus.ref)*8 + pointer*2 + 1, (bus.ref)*8 + 6]
                = -np.real(item)*epsilon
                self.D_I[(bus.ref)*8 + pointer*2 + 1, (bus.ref)*8 + 7]
                = np.imag(item)*epsilon
```

```
            for pointer, item in enumerate(bus.U[:-1]):
                self.D_I[(bus.ref)*8 + pointer*2 + 1, (bus.ref)*8
                + pointer*2] += (1-epsilon)*2*(np.imag(item)
                - np.imag(bus.U[-1]))
                self.D_I[(bus.ref)*8 + pointer*2 + 1, (bus.ref)*8
                + pointer*2 + 1] += (1-epsilon)*2*(np.real(item)
                - np.real(bus.U[-1]))

                self.D_I[(bus.ref)*8 + pointer*2 + 1, (bus.ref)*8 + 6]
                += -(1-epsilon)*2*(np.imag(item) - np.imag(bus.U[-1]))
                self.D_I[(bus.ref)*8 + pointer*2 + 1, (bus.ref)*8 + 7]
                += -(1-epsilon)*2*(np.real(item) - np.real(bus.U[-1]))

                #Se calcula D_v =[ei fi : eps*fi eps*(-ei)]
            for pointer, item in enumerate(bus.U[:-1]):

                self.D_V[(bus.ref)*8 + pointer*2, (bus.ref)*8 + pointer*2]
                = np.real(item) - np.real(bus.U[-1])
                self.D_V[(bus.ref)*8 + pointer*2, (bus.ref)*8 + pointer*2
                + 1] = np.imag(item) - np.imag(bus.U[-1])
                self.D_V[(bus.ref)*8 + pointer*2 + 1, (bus.ref)*8 +
                pointer*2] = epsilon*(np.imag(item) - np.imag(bus.U[-1]))
                self.D_V[(bus.ref)*8 + pointer*2 + 1, (bus.ref)*8 +
                pointer*2 + 1] = -epsilon*(np.real(item) -
                np.real(bus.U[-1]))

                self.D_V[(bus.ref)*8 + 6:(bus.ref)*8 + 8, (bus.ref)*8 +
                pointer*2:(bus.ref)*8 + pointer*2 + 2] = np.eye(2)
                self.D_V[(bus.ref)*8 + 6:(bus.ref)*8 + 8, (bus.ref)*8 +
                6:(bus.ref)*8 + 8] = np.eye(2)

        #Elimina las filas y columnas del nudo slack

        self.D_I = self.D_I[2*4:,2*4:]
        self.D_V = self.D_V[2*4:,2*4:]

        return self.D_I, self.D_V

    def compute_residuals(self):
        U = list()
        I = list()
        res_S = list()

        #Nudo Slack
        for item in self.buses[0].U:
            U.append(np.imag(item))
            U.append(np.real(item))

        #Nudos PQ y PV
        for bus in self.buses[1:]:

            for item in bus.U:
                U.append(np.imag(item))
                U.append(np.real(item))

            for item in bus.I:
                I.append(np.real(item))
                I.append(np.imag(item))

            if bus.tipo == 'PQ':

                for index in range(3):

                    #delta_P = P_esp - ei*Iai - fi*Ibi
                    res_S.append(bus.P[index] -
                    np.real(bus.U[index])*np.real(bus.I[index]) -
                    np.imag(bus.U[index])*np.imag(bus.I[index]))

                    #delta_Q = Q_esp - fi*Iai + ei*Ibi
                    res_S.append(bus.Q[index] -
                    np.imag(bus.U[index])*np.real(bus.I[index]) +
```

```
                    np.real(bus.U[index])*np.imag(bus.I[index]))

                res_S.append(0)
                res_S.append(0)

            if bus.tipo == 'PV':

                for index in range(3):

                    #delta_P = P_esp - ei*Iai - fi*Ibi
                    res_S.append(bus.P[index] - np.real(bus.U[index])*np.real(bus.I[index])
                    -np.imag(bus.U[index])*np.imag(bus.I[index]))

                    #delta_V^2 = V_esp^2 - ei^2-fi^2
                    res_S.append(bus.V_esp[index]**2-
                    (np.real(bus.U[index]))**2-(np.imag(bus.U[index]))**2)

                res_S.append(0)
                res_S.append(0)


        #Fórmula (5) del paper de Esther

        res_I = I - np.dot(self.Y_B, U)
        res_S = np.array(res_S, dtype = complex)

        self.residual = np.concatenate((res_I, res_S))

        return self.residual, np.max(np.abs(self.residual))

    def next_step(self):

        Y_bus = self.generate_Y()
        Y_bus = Y_bus[:,2*4:]
        D_I, D_V = self.generate_jacobian_matrices()
        residual, res = self.compute_residuals()

        #Calculo el Jacobiano

        J = np.zeros((D_I.shape[0] + D_V.shape[0], Y_bus.shape[1]+D_I.shape[1]), dtype = complex)

        J[0:Y_bus.shape[0], 0:Y_bus.shape[1]] = Y_bus
        J[0:Y_bus.shape[0], Y_bus.shape[1]:] = -np.eye(D_V.shape[0], D_V.shape[1])
        J[Y_bus.shape[0]:, 0:D_I.shape[1]] = D_I
        J[Y_bus.shape[0]:, Y_bus.shape[1]:] = D_V

        delta = np.linalg.solve(J, residual)

        delta_U = delta[:(len(self.buses)-1)*8]
        delta_I = delta[(len(self.buses)-1)*8:]

        #Actualizo las tensiones e intensidades

        index = 0
        for bus in self.buses[1:]:
            for index2 in range(len(bus.U)):
                bus.U[index2] += np.complex(delta_U[index + 1], delta_U[index])
                index += 2
        index = 0
        for bus in self.buses[1:]:
            for index2 in range(len(bus.U)):
                bus.I[index2] += np.complex(delta_I[index],
                delta_I[index + 1])
                index += 2

        return delta, delta_U, delta_I, J

    def augmented_rectangular(self, options):

        #Hay que definir la tolerancia y el numero maximo de operaciones
        como argumentos
```

```
        tol , n_iter = options

        #Llamo al atributo que crea la matriz de admitancias

        self .generate_Y()

        #Parto de un residuo = 10
        res = 10
         iteration = 0

        self .x = list ()
        self . lista_residuos = list ()

        #Aplico el algoritmo NR rectangular aumentado
        while res > tol and iteration < n_iter :
            self .generate_Y()
            self . generate_jacobian_matrices ()
            Corriente , res = self .compute_residuals ()


            self . next_step ()
             iteration += 1
            print (' Iteration : ' + str ( iteration ) + '. Residual: ' +
            str (res ))


            self .x.append( iteration )
            self . lista_residuos .append(res)


        # Asignamos valores a las corrientes en líneas
        for line in self . lines :
            line .I = np.dot(np. linalg .inv( line .Z),
            (np. array ( line . connections [0]. U) -
            np. array ( line . connections [1]. U)))
        for trafo in self . transformers :
             tensiones_trafo = np. zeros ((8) , dtype = complex)
             tensiones_trafo [0:4] = trafo . connections [0]. U
             tensiones_trafo [4:8] = trafo . connections [1]. U
            trafo .I = np.dot( trafo .Y_trafo , np. array ( tensiones_trafo ))
            trafo .I = np. array ( trafo .I [4:8])
        return self . lista_residuos

    def print_results ( self , name):

        TENSIONES = [[] for _ in range(len( self .buses))]
        INTENSIDADES = [[] for _ in range(len( self .buses))]

        for index, bus in enumerate( self .buses):
            TENSIONES[index] = bus.ref, (np.round(np.abs(bus.U[0]), 4)),
            (np.round(np.angle(bus.U[0], deg = True), 4)),
            (np.round(np.abs(bus.U[1]), 4)), (np.round(np.angle(bus.U[1],
            deg = True), 4)), (np.round(np.abs(bus.U[2]), 4)),
            (np.round(np.angle(bus.U[2], deg = True), 4)),
            (np.round(np.abs(bus.U[3]), 4)), (np.round(np.angle(bus.U[3],
            deg = True), 4))

            INTENSIDADES[index] = bus.ref, (np.round(np.abs(bus.I [0]) , 4)),
            (np.round(np.angle(bus.I [0], deg = True), 4)),
            (np.round(np.abs(bus.I [1]) , 4)), (np.round(np.angle(bus.I [1],
            deg = True), 4)) ,( np.round(np.abs(bus.I [2]) , 4)),
            (np.round(np.angle(bus.I [2], deg = True), 4)),
            (np.round(np.abs(bus.I [3]) , 4)), (np.round(np.angle(bus.I [3],
            deg = True), 4))


        import xlsxwriter

        with xlsxwriter .Workbook(name) as workbook:
            worksheet = workbook.add_worksheet('Tensiones - Raúl')
            for row_num, data in enumerate(TENSIONES):
```

```
                worksheet.write_row(row_num, 0, data)

            worksheet = workbook.add_worksheet('Corrientes - Raúl')

            for row_num, data in enumerate(INTENSIDADES):
                worksheet.write_row(row_num, 0, data)

    def plot(self):

        import matplotlib.pyplot as plt

        x0 = np.arange(1, len(self.buses))

        plot_U =  [[],[],[],[]]

        for bus in self.buses[1:]:
            plot_U[0].append(abs(bus.U[0]))
            plot_U[1].append(abs(bus.U[1]))
            plot_U[2].append(abs(bus.U[2]))
            plot_U[3].append(abs(bus.U[3]))

        import seaborn as sns
        sns.set()
        #Plot Tensiones

        fig, ax = plt.subplots()
        ax.plot(x0, plot_U[0], linewidth=2.0, label='Fase A')
        ax.plot(x0, plot_U[1], linewidth=2.0, label='Fase B')
        ax.plot(x0, plot_U[2], linewidth=2.0, label='Fase C')
        #ax.plot(x0, plot_U[3], linewidth=2.0, label='Neutro')
        ax.set_xlabel('Nudo')
        ax.set_ylabel('Tensión(V)')
        ax.set_title("Evolución de Tensiones en la Red BT")
        ax.set_ylim(210,240)
        plt.xticks(range(0, len(self.buses),1))
        ax.legend()  # Add a legend.
        plt.savefig('Tensiones')
```

.

# Bibliografía


[1] Comisión Europea. Energía limpia para todos los europeos: desbloquear el potencial de crecimiento de europa. 11 2016.

[2] Red Eléctrica de España. Compromisos de sostenibilidad: Objetivo 2030. Web https://www.ree.es/es/sostenibilidad/compromiso-con-la-sostenibilidad/objetivos-2030, Accedido el 30 Noviembre 2021.

[3] Ministerio para la Transición Ecológica y el Reto Demográfico. Plan nacional integrado de energÍa y clima 2021-2030. 02 2019.

[4] Mohammed Albadi. *Power Flow Analysis*. 03 2019.

[5] Antonio Gomez-Exposito and Esther Romero-Ramos. Augmented rectangular load flow model. *Power Engineering Review, IEEE*, 22:60–60, 03 2002.

[6] Jose Vargas Cantero. Método de newton raphson. 10 2018.

[7] Abaali Lhoussine, Talbi El Hachmi, and Rachid Skouri. Comparison of newton raphson and gauss seidel methods for power flow analysis. 09 2018.

[8] Antonio Gómez-Expósito, Antonio J Conejo, and Claudio Cañizares. *Electric energy systems: analysis and operation*. CRC press, 2018.

[9] Javier Sánchez Reyes. Formulación y programación de un flujo de cargas ampliado para redes trifásicas a cuatro hilos, universidad de sevilla, 2017.

[10] Álvaro Rodríguez del Nozal, Esther Romero-Ramos, and Ángel Luis Trigo-García. Accurate assessment of decoupled oltc transformers to optimize the operation of low-voltage networks. *Energies*, 12(11), 2019.

[11] Álvaro M. Lorente Robles. Incorporación de transformadores y cargas en triángulo a flujo de cargas extendido para sistemas trifásicos desequilibrados, universidad de sevilla, 2020.

[12] Python.org. Documentación acerca del lenguaje de programación python. Web https://www.python.org/about/, Accedido el 30 Noviembre 2021.

[13] Anaconda.org. Documentación acerca del del entorno anaconda. Web https://www.anaconda.com/about-us, Accedido el 30 Noviembre 2021.

[14] K. Strunz, Ehsan Abbasi, Robert Fletcher, Nikos Hatziargyriou, Reza Iravani, and Géza Joos. *TF C6.04.02 : TB 575 – Benchmark Systems for Network Integration of Renewable and Distributed Energy Resources*. 04 2014.

[15] Álvaro Rodríguez del Nozal, Esther Romero-Ramos, and Ángel Luis Trigo-García. Accurate assessment of decoupled oltc transformers to optimize the operation of low-voltage networks. *Energies*, 12(11), 2019.

[16] Ministerio de Ciencia y Tecnología. Itc-bt-19. 2019.

[17] Debora Penido, Leandro Araujo, Sirley Carneiro, Jose Pereira, and Paulo Garcia. Three-phase power flow based on four-conductor current injection method for unbalanced distribution networks. *Power Systems, IEEE Transactions on*, 23:494 – 503, 06 2008.