UNIVERSIDAD DE OVIEDO

Escuela de
Ingeniería de Minas, Energía y Materiales de Oviedo

Máster en Ingeniería de Minas

Trabajo Fin de Máster

Aplicación en tiempo real de técnicas de cálculo de flujo de cargas en redes de distribución

Autor: Jaime Olascoaga Durán

Tutor: José Manuel Cano Rodríguez

Cotutor: Carlos Hiram Rojas García

Oviedo, julio de 2018

Índice de contenidos

1. Introducción y objetivos 3

2. Estructura de un Sistema Eléctrico de Potencia 4

2.1 Sistemas Eléctricos de Potencia 4

2.1 Centrales de generación 6

2.2 Red de transporte 7

2.5 Red de distribución 8

2.3 Subestaciones 8

2.4 Centro de Control Eléctrico 9

3. Modelización de los Sistemas Eléctricos de Potencia 10

3.1 Sistema por unidad 11

3.2 Cambio de base en valores por unidad 15

3.3 Líneas 16

3.4 Transformadores 18

3.4.1 Transformador ideal 18

3.4.2 Transformador real 19

3.4.3 Transformador regulable 24

4. Flujo de cargas 29

4.1 Formulación del problema 29

4.2 Método de Newton-Raphson 40

5. Caso de Estudio 45

5.1 Presentación de la red sintética 46

5.2 Modelización de la red sintética 48

5.2.1 Selección de las bases del sistema 49

5.2.2 Modelización de las líneas eléctricas 50

5.2.3 Modelización de los transformadores con tomas 52

5.3 Comprobación del modelo de la red de distribución en PowerWorld 58

5.4 Planteamiento del cálculo de flujo de cargas mediante Newton-Raphson..... 60

5.5 Implementación del cálculo de flujo de cargas en Matlab y Python..... 69

5.5 Implementación del cálculo de flujo de cargas en tiempo real en Python 3 73

5.5.1 Obtención medidas de potencia en tiempo real con PowerMeter 74

5.5.2 Código para el cálculo de flujo de cargas en Python 3 74

5.6 Aplicación cálculo flujo de cargas en tiempo real con Raspberry Pi 3 Model B+ ... 77

5.6.1 Presentación de la Raspberry Pi 77

5.6.2 Resultados obtenidos..... 79

6. Conclusiones 82

7. Bibliografía..... 83

8. ANEXOS..... 85

ANEXO I – Obtención de las ecuaciones de flujo de cargas..... 86

ANEXO II – Implementación en Matlab del método de Newton-Raphson 88

ANEXO III - Implementación en Python 3 del método de Newton-Raphson..... 98

ANEXO IV - Implementación en Python 3 del cálculo de flujo de cargas en tiempo real 111

1. Introducción y objetivos

En este trabajo se implementará una técnica capaz de realizar el cálculo de flujo de cargas en una red eléctrica de distribución. Esta técnica será probada en una red de distribución sintética en la que la medida de potencia de algunos de los nodos será obtenida desde dispositivos reales. De esta forma, se podrá garantizar la capacidad del algoritmo de flujo de cargas desarrollado para funcionar en tiempo real.

Para ello, se implementará en Matlab el método de Newton-Raphson para el cálculo del flujo de cargas de la red de distribución sintética. Este algoritmo será validado con PowerWorld, un software de análisis y simulación de sistemas de energía eléctrica.

El objetivo de este trabajo es hacer funcionar, una vez validado, el algoritmo desarrollado de Newton-Raphson en un dispositivo autónomo de bajo coste (Raspberry Pi). Por esta razón, el algoritmo también se programará en un lenguaje abierto (Python) que no necesita de ninguna licencia propietaria, como en el caso de Matlab, y que puede correr en prácticamente cualquier tipo de hardware. Este algoritmo se modificará de tal forma que finalmente será capaz de utilizar medidas de potencia de algunos nodos obtenidas en tiempo real.

La idea de implementar el algoritmo en la Raspberry Pi no tiene que ver solo con el bajo coste del dispositivo, sino por el hecho de que es un elemento que podría formar de un equipo más grande (por ejemplo, algún generador o carga que utilizase los resultados del flujo de cargas para generar sus referencias), resultando así en una aplicación de gran interés.

2. Estructura de un Sistema Eléctrico de Potencia

Se comienza describiendo en este apartado qué es un sistema eléctrico de potencia y los elementos básicos que lo componen. Dentro de los sistemas eléctricos de potencia es de especial interés para este trabajo el conocimiento de las redes de distribución, ya que sobre una red de este tipo se aplicará la técnica de cálculo de flujo de cargas.

Además de presentar los diferentes elementos que componen un sistema eléctrico de potencia, se incluye una breve descripción del funcionamiento y operación del sistema eléctrico español, con el fin de mostrar la complejidad de estos sistemas de enorme tamaño.

2.1 Sistemas Eléctricos de Potencia

Un **sistema eléctrico de potencia** es el conjunto de elementos que operan de forma coordinada en un determinado territorio para satisfacer la demanda de energía eléctrica de los consumidores [1].

El objetivo de un sistema eléctrico de potencia es, por tanto, la generación, transmisión, y distribución de energía eléctrica a los consumidores, de tal forma que se entregue esa energía de la manera más segura y económica posible.

Los sistemas eléctricos de potencia se estructuran en centros de producción (generación), de transporte (red de alta tensión), de distribución (red de media y baja tensión), y de consumo. También se incluyen dentro de los sistemas eléctricos de potencia los sistemas asociados de protección y control del mismo.

Para mayor claridad se incluye un esquema del sistema eléctrico peninsular, en el que se recogen todos los elementos básicos que lo constituyen.

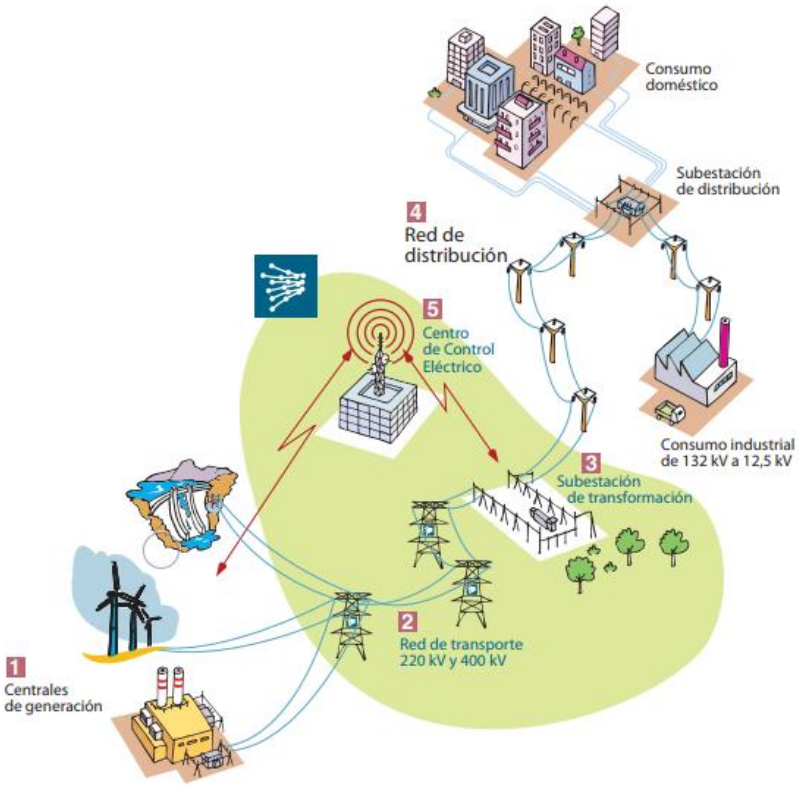


Fig. 1- Esquema del sistema eléctrico peninsular. Fuente: [1]

De acuerdo con la numeración del esquema, se distinguen los siguientes elementos:

1. Centrales o plantas de generación donde se produce la electricidad (centrales nucleares, hidroeléctricas, de ciclo combinado, parques eólicos, etc.).
2. Líneas de transporte de la energía eléctrica de alta tensión (AT).
3. Estaciones transformadoras (subestaciones) que reducen la tensión o el voltaje de la línea (Alta tensión/Media tensión, Media tensión/Baja tensión),
4. Líneas de distribución de media y baja tensión que llevan la electricidad hasta los puntos de consumo,
5. Centro de control eléctrico desde el que se gestiona y opera el sistema de generación y transporte de energía.

En función de la cantidad de energía a transportar y de la distancia a recorrer, cada parte de la red conduce la electricidad a una tensión u otra.

El voltaje de la energía eléctrica, una vez generada, es elevado a alta tensión para reducir las pérdidas de energía que se producen en el transporte, y posteriormente se va

transformando a media y baja tensión para acercarla al consumidor final a través de las redes de distribución. En función de su voltaje existen:

- Las líneas de alta tensión (AT), entre 380.000 y 132.000 V. Se utilizan para transportar grandes cantidades de energía a largas distancias.
- Las líneas de media tensión (MT), entre 132.000 y 1.000 V.
- Y las líneas de baja tensión (BT), que llevan la energía hasta el punto de consumo, a una tensión inferior a los 1.000 V, ya que los equipos domésticos y algunos industriales funcionan con un voltaje de 380 o 220 V.

En las siguientes páginas se describen brevemente cada uno de los elementos del sistema eléctrico que se mencionaron anteriormente de acuerdo con [2].

2.1 Centrales de generación

Las centrales de generación son las encargadas de producir energía eléctrica. Junto a estas centrales se encuentra la subestación de generación que sirve para conectar dichas centrales a la red de alta tensión.

Existen diferentes medios para producir energía eléctrica, desde los convencionales de origen térmico (centrales de carbón, de fuel, nucleares o los nuevos ciclos combinados de gas natural), hasta las tecnologías más novedosas, entre las que se engloban las llamadas energías alternativas o renovables que aprovechan fuentes de energía como el sol o el viento, entre otras.

Fig. 2- Central (nuclear) de generación eléctrica. Fuente: [2]

2.2 Red de transporte

La red de transporte de alta tensión es una red mallada compuesta por líneas y subestaciones de tensión superior a 220.000 voltios (220 kV), que cubre todo el territorio español y es básica para llevar la energía eléctrica desde las centrales de generación hasta las áreas de distribución. Esta configuración mallada garantiza una gran fiabilidad, existiendo caminos alternativos para evacuar y recibir energía eléctrica en el caso de que algunas líneas fallen.

Generalmente, las zonas donde se sitúan las centrales eléctricas no coinciden con los lugares en los que se consume la energía, por lo que es necesario transportarla a través de miles de kilómetros de conductores de alta tensión. La mayor parte de ellos son conductores aéreos, y en ciertos casos, cables subterráneos o incluso submarinos.

El transporte de electricidad entre largas distancias se efectúa a través de las líneas de alta tensión con el fin de reducir las pérdidas y lograr una mayor eficiencia energética.

Las líneas de alta tensión y más de 400 estaciones transformadoras pertenecen a la empresa Red Eléctrica de España, la cual gestiona su ampliación y desarrollo, así como su mantenimiento. Las líneas de media y baja tensión, por el contrario, son propiedad de distintas compañías eléctricas que son las que distribuyen la electricidad hasta el consumidor final.

La red de transporte de Red Eléctrica está compuesta por más de 43.000 kilómetros de líneas de alta tensión, más de 5.000 posiciones de subestaciones y más de 85.000 MVA de capacidad de transformación [3]. Estos activos configuran una red mallada, fiable y segura, que ofrece unos índices de calidad de servicio de máximo nivel al sistema eléctrico nacional.

Red de transporte peninsular y no peninsular de Red Eléctrica (Datos acumulados a 31 de diciembre del 2017)

Km de circuito	2013	2014	2015	2016	2017
400kV	20.639	21.094	21.184	21.619	21.728
220kV	19.053	19.192	19.386	19.479	19.507
150 - 132 - 110kV	272	272	398	523	523
< 110kV	2.014	2.015	2.023	2.025	2.035
Total	**41.978**	**42.572**	**42.989**	**43.646**	**43.793**

Tabla 1. Kilómetros de la red de transporte de Red Eléctrica. Fuente: [3]

2.5 Red de distribución

La red de distribución de media y baja tensión lleva la energía desde los centros de distribución hasta el consumidor final, adecuándola al nivel de tensión que éste necesite. En la red de media tensión, ésta varía desde los 1.000 a los 30.000 voltios. Las líneas de baja tensión tienen menos de 1.000 voltios.

El consumo puede ser de tipo industrial o doméstico:

- La industria pesada utilizada 33.000 voltios.
- El uso público (ferrocarril, metro, etc) utiliza de 15.000 a 25.000 voltios.
- La industria ligera usa de 380 a 415 voltios.
- Los consumidores domésticos usan 220 o 240 voltios.

La configuración de la red de distribución es bien distinta a la de transporte. En un primer nivel y a nivel regional se extiende una red, todavía de alta tensión, llamada red de reparto y que puede tener aún una estructura mallada como la red de transporte. Desde las subestaciones de esta red cuelga a su vez una red de media tensión que se acerca ya más al consumo más repartido. Por razones económicas, esta red presenta una estructura a veces algo mallada, pero se opera siempre radialmente. Finalmente, desde esta red se vuelve a reducir la tensión (subestaciones de distribución) para alimentar en baja tensión a los consumidores domésticos, comerciales, etc. Los fallos en las redes de distribución son los causantes de la mayoría de los cortes de suministro en los consumidores finales.

2.3 Subestaciones

Las subestaciones de transformación están situadas por toda la red, siendo las encargadas de adecuar la tensión según los distintos escalones de la red de transporte, transformándola de alta a media o baja tensión.

Las subestaciones son necesarias para el funcionamiento del sistema porque conectan entre sí varias líneas, bien directamente, si son de la misma tensión o mediante transformadores, si son de diferentes tensiones.

En la subestación, se recoge toda la información relativa al funcionamiento de los equipos y elementos de la red y se envía a los centros de control para su correcta operación.

En el sistema eléctrico, además de las subestaciones de transformación cuyo responsable es Red Eléctrica, existen subestaciones de generación y distribución. Las de generación conectan las centrales eléctricas a la red elevando la tensión hasta los 400 kV y las de distribución reducen la tensión conforme se acercan a zonas industriales o núcleos de población.

Fig. 3- Subestación de transformación. Fuente: [2]

2.4 Centro de Control Eléctrico

El Centro de Control Eléctrico de Red Eléctrica (CECOEL) emite las instrucciones de operación del sistema de producción y transporte con el fin de garantizar el equilibrio constante entre la generación y el consumo de electricidad. El CECOEL es el responsable de la operación y supervisión coordinada en tiempo real de las instalaciones de generación y transporte del sistema eléctrico español.

Su función consiste en lograr que, en cada instante, durante las 24 horas del día, y todos los días del año, la generación programada en las centrales eléctricas coincida con el consumo real de electricidad. En caso de que difiera, se envían las órdenes oportunas a las centrales para que se ajusten sus producciones, aumentando o disminuyendo la generación de energía.

Fig. 4- Centro de Control Eléctrico de Red Eléctrica. Fuente: [2]

3. Modelización de los Sistemas Eléctricos de Potencia

Se ha visto que un sistema eléctrico de potencia comprende la producción de energía eléctrica en centrales eléctricas, su transporte y distribución mediante las líneas eléctricas, y el consumo de energía que tiene lugar en las industrias, viviendas y comercios.

En este apartado se presentan los modelos de los elementos básicos que constituyen un sistema eléctrico de potencia, suponiendo régimen estacionario equilibrado. Concretamente, los elementos básicos que se consideran son: líneas eléctricas y transformadores.

Estos modelos serán empleados más adelante en este trabajo para realizar la modelización de una red distribución sobre la que se realizará el cálculo de flujo de cargas. Asimismo, se introduce en este apartado el sistema por unidad y la selección y cambio de bases en el análisis de redes eléctricas, operaciones que se utilizarán también para el caso de estudio de la red de distribución.

El régimen estacionario hace referencia a la modelización de la red eléctrica mediante un circuito en régimen estacionario sinusoidal. Por tanto, sólo se consideran variables eléctricas y le son de aplicación el método fasorial de resolución de circuitos. El término equilibrado se refiere a un sistema trifásico balanceado o equilibrado, el cual se revuelve siempre como un circuito monofásico equivalente.

En este trabajo se realizará el análisis de flujo de cargas de una red eléctrica de distribución, donde los generadores y los consumos, o cargas, se formulan con ecuaciones no lineales en régimen estacionario sinusoidal.

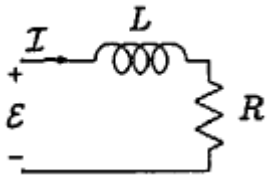


Fig. 5- Ejemplo de régimen estacionario de una red eléctrica. Fuente: [4]

3.1 Sistema por unidad

Se va a presentar el sistema por unidad, el cual es el sistema de unidades empleado para trabajar con sistemas eléctricos de potencia dado las grandes ventajas que proporciona.

En el análisis de redes eléctricas, debido a la existencia de elementos con diferentes niveles de tensión, potencias nominales y demás valores de sus parámetros, es necesario realizar previamente un adimensionamiento y normalización de todas las magnitudes eléctricas y parámetros del sistema [4].

Este escalado se realiza dividiendo el valor del parámetro (impedancia [Z], admitancia [Y]) o de la variable (tensión [V], potencia [P, Q], intensidad [I]) por un valor base o de referencia, transformando así el circuito eléctrico con valores en unidades físicas (kV, MW, kΩ…) a *valores por unidad (p.u.)*.

$$\boxed{\text{valor}_{\text{p.u.}} = \frac{\text{valor de la variable}}{\text{valor base}}}$$

Por ejemplo, si se selecciona una base de voltaje de 120 kV, los voltajes de 108, 120 y 126 kV equivaldrán a 0.90, 1.00 y 1.05 en por unidad.

$$\text{valor}_{\text{p.u.}} = \frac{108\ \text{kV}}{120\ \text{kV}} = 0.90$$

$$\text{valor}_{\text{p.u.}} = \frac{120\ \text{kV}}{120\ \text{kV}} = 1.00$$

$$\text{valor}_{\text{p.u.}} = \frac{126\ \text{kV}}{120\ \text{kV}} = 1.05$$

A continuación, se expresan las transformaciones a p.u. de algunos parámetros y variables eléctricas:

$$U(\text{V}) \rightarrow U_{p.u.} = \frac{U}{U_B}\ (\text{p.u.}) \quad ; \quad I(\text{A}) \rightarrow I_{p.u.} = \frac{I}{I_B}\ (\text{p.u.})$$

$$P(\text{W}) \rightarrow P_{p.u.} = \frac{P}{S_B}\ (\text{p.u.}) \quad ; \quad Q(\text{var}) \rightarrow Q_{p.u.} = \frac{Q}{S_B}\ (\text{p.u.})$$

$$Z(\Omega) \rightarrow Z_{p.u.} = \frac{Z}{Z_B}\ (\text{p.u.})$$

La realización del análisis de los sistemas eléctricos en p.u. presenta notables ventajas, entre las que se pueden destacar [5]:

- Simplificación de los cálculos manuales y reducción de los errores computacionales.
- Se obtienen valores acotados, con lo que los errores se hacen más evidentes.
- Desaparecen del problema las relaciones de transformación.
- No es necesario distinguir entre magnitudes de fase y de línea.

Los valores base, que se hacen coincidir con los valores nominales en la medida de lo posible, son los módulos de las magnitudes eléctricas principales: U_B, I_B, S_B, Z_B.

Lo más habitual es tomar como magnitudes base la tensión en un punto del sistema y una potencia base común a todo el sistema (U_B y S_B). Para la selección de las bases en una red eléctrica se debe proceder de la siguiente manera:

1. Se selecciona una tensión base (U_B) por cada nivel de tensión nominal que haya en la red, teniendo en cuenta que las tensiones base deben tener la misma relación de transformación que los transformadores. Es decir, cada devanado del transformador define una zona de valores de tensión base (U_{B1}, U_{B2}).

2. Se utiliza una potencia base única para toda la red (S_B).

Para mayor claridad, se expone un pequeño ejemplo recogido en [6] para ver cómo se seleccionan las bases de una red eléctrica. El primer transformador (T1) tiene una relación de transformación 13.2/132 kV y 5 MVA de potencia. Las tensiones del segundo transformador (T2) son 132/69 kV, y su potencia es 10 MVA.

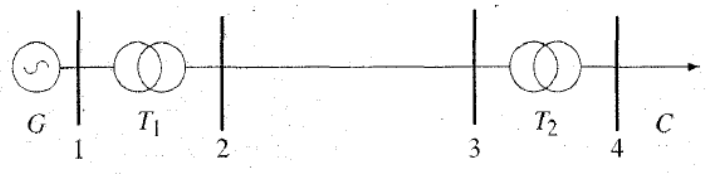


Fig. 6- Esquema unifilar de la red del ejemplo. Fuente [6]

Se adoptará una potencia base de valor S_B = 10MVA (común para toda la red). La elección de la base de potencia es arbitraria y se puede seleccionar cualquier valor, aunque es práctica habitual utilizar la potencia del mayor transformador de la red.

Debido a la existencia de los dos transformadores, se pueden distinguir en la red tres zonas con diferentes tensiones base. Estas zonas se muestran en la siguiente figura.

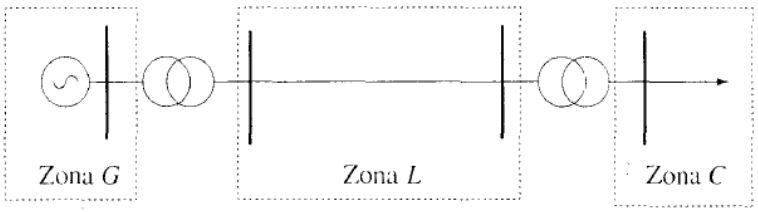


Fig. 7- Zonas de tensión en la red del ejemplo. Fuente: [6]

De esta forma, las tensiones base para las zonas G, L y C son:

- $U_{BG} = 13.2$ kV
- $U_{BL} = 132$ kV
- $U_{BC} = 69$ kV

Con la selección de estas bases, las cuales satisfacen la relación de transformación de los transformadores, se consigue eliminar precisamente la relación de transformación en el cálculo en p.u.

$$\frac{U_{BG}}{U_{BL}} = \frac{U_G}{U_L} = \frac{13.2\ \text{kV}}{132\ \text{kV}}$$

$$\frac{U_{BL}}{U_{BC}} = \frac{U_L}{U_C} = \frac{132\ \text{kV}}{69\ \text{kV}}$$

Una vez establecida la pareja de valores base (S_B , U_B), el resto de bases (I_B , Z_B) quedan definidas a través de las ecuaciones eléctricas básicas, que en un sistema monofásico son:

$$S_B = U_B \cdot I_B \qquad ; \qquad U_B = Z_B \cdot I_B$$

$$\boxed{I_B = \frac{S_B}{U_B}}$$

$$\boxed{Z_B = \frac{{U_B}^2}{S_B}}$$

Sin embargo, en las redes trifásicas es más habitual adoptar como tensión base en cada zona la tensión de línea (U_L) y como potencia base una potencia trifásica ($S_{3\phi}$) común para todo el sistema: U_B y S_B. La intensidad base de línea y la impedancia base se obtienen de forma similar a las de un sistema monofásico, pero teniendo en cuenta las siguientes relaciones trifásicas:

$$S_{3\phi} = 3 \cdot S_{1\phi}$$

$$S_{3\phi} = \sqrt{3} \cdot U_L \cdot I_L$$

$$U_L = \sqrt{3} \cdot U_{Fase}$$

De este modo, la impedancia base tiene una expresión idéntica a la de las bases monofásicas. Utilizando las relaciones anteriores:

$$\boxed{Z_B} = \frac{{U_{FaseB}}^2}{S_{1\phi B}} = \frac{\left(\frac{U_L}{\sqrt{3}}\right)^2}{\frac{S_{3\phi B}}{3}} = \frac{{U_{LB}}^2}{S_{3\phi B}} \equiv \boxed{\frac{{U_B}^2}{S_B}}$$

Mientras que la intensidad base se obtiene a partir de la relación trifásica siguiente:

$$\boxed{I_B = \frac{S_B}{\sqrt{3} \cdot U_B}}$$

En nuestro ejemplo, las impedancias base de cada zona serían:

$$Z_{BG} = \frac{{U_{BG}}^2}{S_B} = 17{,}424\ \Omega$$

$$Z_{BL} = \frac{{U_{BL}}^2}{S_B} = 1742{,}4\ \Omega$$

$$Z_{BC} = \frac{{U_{BC}}^2}{S_B} = 476{,}1\ \Omega$$

Y las intensidades base:

$$I_{BG} = \frac{S_B}{\sqrt{3} \cdot U_{BG}} = 437{,}387\ \text{A}$$

$$I_{BL} = \frac{S_B}{\sqrt{3} \cdot U_{BL}} = 43{,}739 \text{ A}$$

$$I_{BC} = \frac{S_B}{\sqrt{3} \cdot U_{BC}} = 83{,}674 \text{ A}$$

Lógicamente, los resultados de los cálculos que se obtengan estarán en valores p.u. y se requiere la operación contraria para pasar de p.u. a valores en unidades físicas.

La utilización de valores por unidad aporta grandes ventajas en el análisis de sistemas eléctricos de potencia, simplificando enormemente los cálculos, sobre todo cuando existen transformadores. La elección de valores base adecuados significa la desaparición de la relación de transformación entre los devanados del transformador. Se observa también que desaparece el factor $\sqrt{3}$ en las ecuaciones de sistemas trifásicos, es decir, se obtiene directamente un circuito monofásico en p.u.

3.2 Cambio de base en valores por unidad

Normalmente, los parámetros de los elementos de la red eléctrica suelen venir expresados en p.u., o en tanto por ciento (0.5 p.u. ≡ 50%), referidos a la base definida por sus valores nominales (potencia nominal, tensión nominal). Esta base, no tiene por qué coincidir con la que se utilice en un lugar de la red donde se quiere realizar un análisis concreto de la misma y, por ello, hay que realizar un cambio de base para unificar los datos de todos los elementos.

Los valores de un elemento que están referidos a unas bases U_{B1} y S_{B1} se pueden referir a unas nuevas bases U_{B2} y S_{B2} con las siguientes relaciones:

$$U_{p.u.2} = U_{p.u.1} \cdot \frac{U_{B1}}{U_{B2}}$$

$$S_{p.u.2} = S_{p.u.1} \cdot \frac{S_{B1}}{S_{B2}}$$

$$Z_{p.u.2} = Z_{p.u.1} \cdot \frac{{U_{B1}}^2}{{U_{B2}}^2} \cdot \frac{S_{B2}}{S_{B1}}$$

3.3 Líneas

Las líneas eléctricas son los elementos básicos que constituyen las redes eléctricas, cuya función es el transporte de la energía eléctrica entre dos puntos. Su funcionamiento viene caracterizado por cuatro parámetros: resistencia (R), inductancia (L), capacidad (C) y conductancia (G) [4]. Por tanto, las líneas se modelan mediante estos cuatro parámetros circuitales:

- Impedancia serie o longitudinal: $Z = R + j\omega L = R + jX\ (\Omega/km)$
- Impedancia shunt o transversal: $Y = G + j\omega C = G + jB(S/km)$

Donde X es la reactancia y B es la susceptancia.

A continuación, se presentan los diferentes modelos de la línea que se pueden construir a partir de ellos. Estos modelos dependen de la longitud de la línea (L):

Línea corta (L < 80 km)

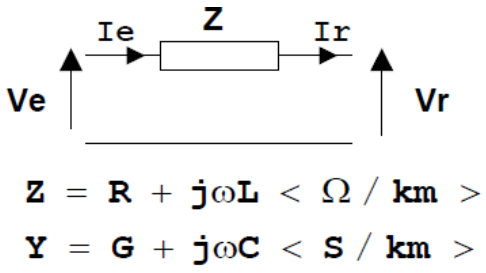


Fig. 8- Modelo línea de longitud corta. Fuente: [7]

Línea media (80 < L < 240 km)

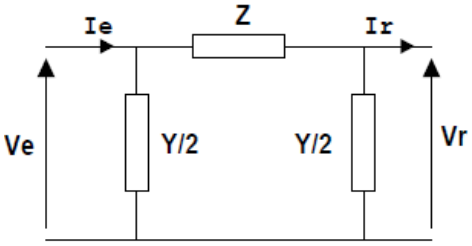
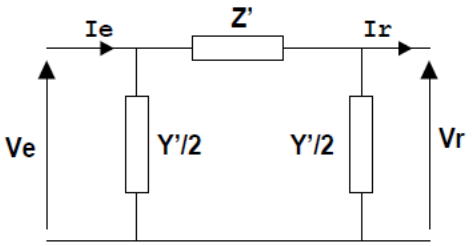


Fig. 9- Modelo línea de longitud media. Fuente: [7]

Línea larga (L > 240 km)



$$Y' = Y\frac{\tanh s\,(\gamma\ell)}{\gamma\ell}$$

$$z' = z\frac{\sinh(\gamma\ell)}{\gamma\ell}$$

$$\gamma = \sqrt{Zy} \text{ y } \ell = \text{longitud}$$

Fig. 10- Modelo línea de longitud larga. Fuente: [7]

En las líneas de longitud media se incluye la admitancia en paralelo, que representa el efecto capacitivo. Si se divide esta admitancia en dos partes iguales y cada una se coloca en un extremo de la línea, se obtiene el llamado modelo π de la línea. Generalmente se utiliza el valor de la susceptancia (B) como valor de la admitancia en paralelo (Y).

En las líneas de longitud larga se considera que los parámetros de la línea no están agrupados sino distribuidos uniformemente a lo largo de la misma.

En el caso de estudio que se presenta más adelante, las líneas de la red de distribución apenas superan los 5 kilómetros de longitud, por lo que las líneas de esta red se modelarán de acuerdo con el modelo de línea corta, en el que solo se tiene en cuenta la impedancia longitudinal o en serie ($Z = R + jX$).

Por razones económicas, interesa que cada línea sea capaz de transportar la máxima energía posible en condiciones adecuadas para el consumo. Ese valor máximo está limitado por varios factores, como las características propias de la red en la que esté inmersa, o los límites tecnológicos de los propios elementos que forman la línea [4]:

- La intensidad en cualquier punto de la línea no puede superar la corriente máxima admisible por los conductores (límite térmico): $I < I_{max}$.
- La tensión en todos los puntos debe mantenerse dentro de unos márgenes, que eviten defectos de aislamiento y para ofrecer a los consumidores un valor de tensión que permita el correcto funcionamiento de los equipos conectados: $U_{min} < U < U_{max}$.

3.4 Transformadores

Otro de los elementos fundamentales en las redes eléctricas son los transformadores de potencia, que son utilizados para elevar, reducir o regular los niveles de tensiones en la red. Los transformadores son los enlaces entre los centros de generación del sistema eléctrico de potencia y las líneas de transmisión, y entre líneas de diferentes niveles de tensión.

En este apartado se presenta el modelo eléctrico de los transformadores, introduciendo primero el modelo del transformador ideal, a partir del cual se deriva el modelo del transformador real y del transformador regulable. La red de distribución analizada en este trabajo tiene varios transformadores regulables, por lo que es necesario introducir en estos capítulos previos el modelo eléctrico de dichos transformadores.

3.4.1 Transformador ideal

Un transformador de potencia está constituido básicamente por dos devanados o bobinas arrolladas sobre un núcleo de material ferromagnético, de tal forma que están enlazadas por el mismo flujo magnético **[8]**.

Los dos devanados son denominados devanado primario y secundario. Como norma general los primario y secundario son arrollamientos con un número distinto de vueltas y uno de ellos es la "entrada" donde se conecta un generador o línea de alimentación y el otro es la salida por la que se pretende obtener un valor de tensión diferente. La tensión aplicada a la entrada provoca un flujo magnético que se canaliza por el núcleo de la máquina y que finalmente induce una tensión en el segundo devanado. La relación entre las tensiones de entrada y salida está determinada por la relación entre el número de espiras de ambos bobinados.

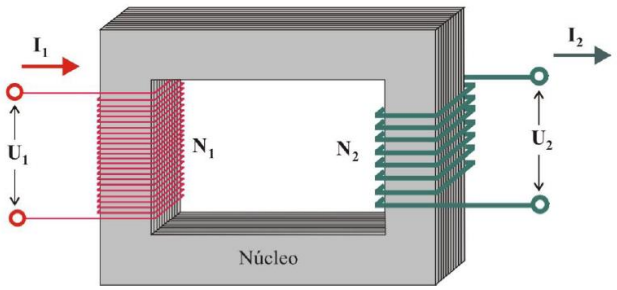


Fig. 11- Transformador monofásico ideal. Fuente: [9]

El transformador monofásico ideal, que responde al circuito eléctrico de la Figura 12, se define a partir de los siguientes supuestos:

- Los devanados tienen resistencia nula.
- El flujo magnético está totalmente confinado en el núcleo magnético, es decir, no hay flujo de dispersión.
- El núcleo tiene una reluctancia magnética nula.
- No hay pérdidas por histéresis ni corrientes parásitas en el núcleo.

En el transformador se produce una transformación energética. En el primario se introduce una potencia (o energía) y por el secundario se extrae otra potencia (o energía) que se ha de diferenciar de la primera tan sólo en las pérdidas internas de la máquina (realmente bajas 2-3% en grandes máquinas). Si se considera que la potencia eléctrica es el producto de la tensión por la corriente y se desprecian las pérdidas (transformador ideal) se llega a que en un transformador la relación entre corrientes de primario y secundario ha de ser inversa a la relación de tensiones.

$$\boxed{S_p = S_s} \rightarrow \boxed{U_p \cdot I_p = U_s \cdot I_s}$$

$$\boxed{r_t = \frac{N_p}{N_s} = \frac{U_p}{U_s} = \frac{I_s}{I_p}}$$

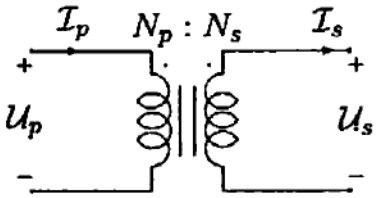


Fig. 12- Circuito eléctrico del transformador monofásico ideal. Fuente: [4]

3.4.2 Transformador real

Cuando a un transformador real se le aplica una tensión en uno de sus devanados mientras que en el otro se le conecta una carga, se observa que las potencias (activas y

reactivas) en los dos devanados no son iguales, y la relación entre las tensiones primaria y secundaria no coincide exactamente con la relación de transformación N_p/N_s.

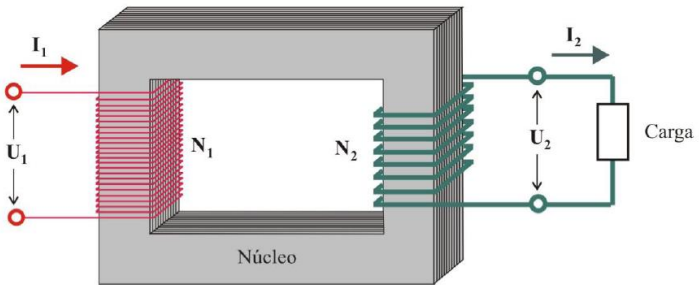


Fig. 13- Transformador en carga. Fuente: [9]

Tan pronto como se cierra el circuito del secundario, al estar este bobinado sometido a tensión circula una corriente por él y por la carga. Dicha corriente crea, de igual manera que la del primario una cierta cantidad de flujo de dispersión. El flujo de dispersión es aquella fracción del flujo magnético que no se canaliza a través del núcleo, sino que se cierra sobre sí mismo a través del aire.

Esto plantea la necesidad de representar el transformador mediante un modelo más cercano a la realidad (Figura 14), en el que se consideren las pérdidas óhmicas y los flujos de dispersión en los devanados mediante la inclusión de las resistencias (R_p y R_s) y reactancias (X_p y X_s) en serie. La rama formada por la resistencia R_{Fe} y la reactancia X_m representan las pérdidas que tienen lugar en el propio núcleo ferromagnético por corrientes de Foucault y fenómenos de histéresis y el hecho de que la reluctancia magnética del circuito magnético no es nula [4].

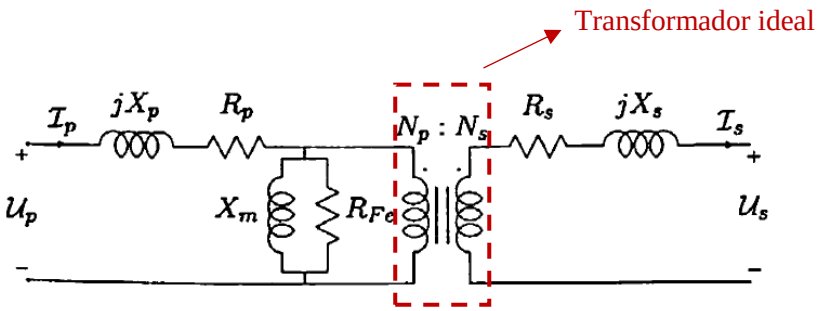


Fig. 14- Circuito eléctrico del transformador real. Fuente: [4]

Existen dos ensayos normalizados que permiten obtener las caídas de tensión, pérdidas y parámetros del modelo del transformador real [8]:

- Ensayo de vacío: Consiste en aplicar la tensión de fase nominal al primario del transformador manteniendo el secundario en circuito abierto (en vacío). En estas condiciones se mide mediante el vatímetro (representado por el elemento circular rojo conectado en paralelo al primario) la potencia total absorbida y así como la corriente de vacío mediante el amperímetro (elemento circular amarillo conectado en serie). Si se conoce la corriente total de vacío I_0 medida por el amperímetro, la tensión de alimentación U_p y las pérdidas indicadas por el vatímetro, mediante la aplicación de las leyes básicas de los circuitos eléctricos es posible deducir el valor de la resistencia de pérdidas en el núcleo R_{Fe} (normalmente denominadas pérdidas en el hierro) y el valor del módulo de la reactancia magnetizante X_m.

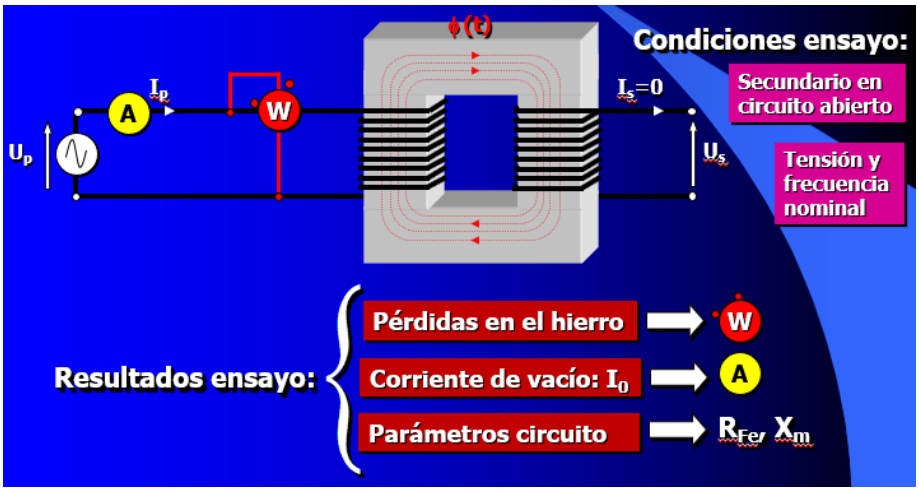


Fig. 15- Ensayo de vacío de un transformador. Fuente: [8]

- Ensayo de cortocircuito: Para poder realizar esta prueba es necesario poder alimentar el transformador con una fuente de alimentación variable en tensión (normalmente un autotransformador). Asimismo, al igual que en el ensayo de vacío, se utiliza un vatímetro y un amperímetro o cualquier instrumento digital capaz de medir la potencia y la corriente absorbida por la máquina. A diferencia del ensayo de vacío éste tiene lugar con el secundario conectado en cortocircuito. Por este motivo, es por lo que es necesario aplicar una fuente de tensión variable ya que si se aplicase la tensión asignada a la máquina se dañaría de forma permanente o actuarían sus

protecciones. El ensayo se desarrolla subiendo la tensión de la fuente desde 0 hasta que la corriente circulante por el primario sea la corriente nominal. En ese instante se registran las pérdidas medidas por el vatímetro. Puesto que para alcanzar la corriente nominal hará falta una tensión muy reducida las pérdidas magnéticas se podrán considerar despreciables, correspondiendo, por tanto, la potencia a la disipada en la resistencia de los conductores del primario y secundario (denominadas "pérdidas en el cobre"). El otro parámetro del transformador que se obtiene de este ensayo es la impedancia de cortocircuito $Z_{cc} = R_{cc} + jX_{cc}$.

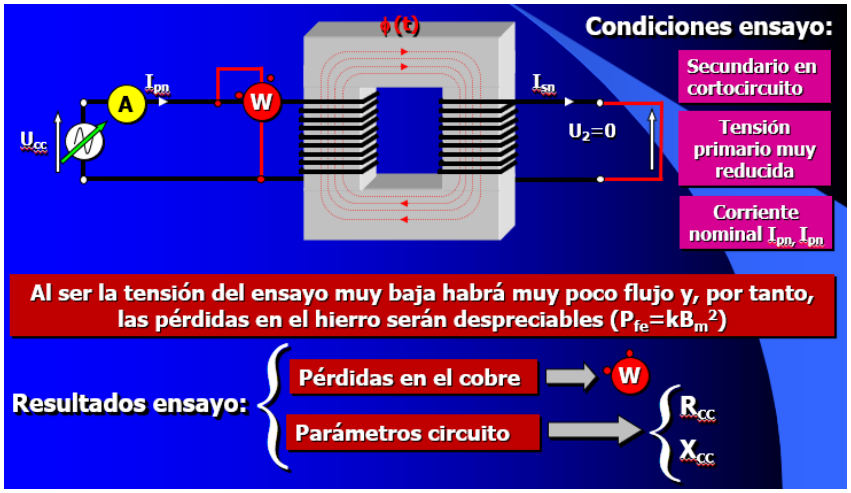


Fig. 16- Ensayo de cortocircuito de un transformador. Fuente: [8]

En condiciones cercanas a las nominales de trabajo del transformador se puede despreciar la corriente de vacío (la que circula por la rama paralela (R_{Fe}, X_m) del modelo del transformador) por ser muy inferior a la corriente de carga y, en consecuencia, puede eliminarse la rama paralela del circuito de la Figura 14. Si este circuito resultante lo pasamos a valores por unidad el modelo del transformador se reduce al circuito en serie de la siguiente figura, en el que aparece únicamente la impedancia de cortocircuito ($Z_{cc,pu} = R_{cc,pu} + jX_{cc,pu}$):

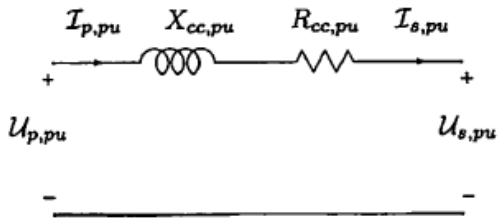


Fig. 17- Circuito eléctrico del transformador monofásico real en p.u. Fuente: [4]

Una vez presentado el modelo del transformador real, se presenta a continuación un ejemplo de [4] para demostrar la ventaja de los cálculos en por unidad.

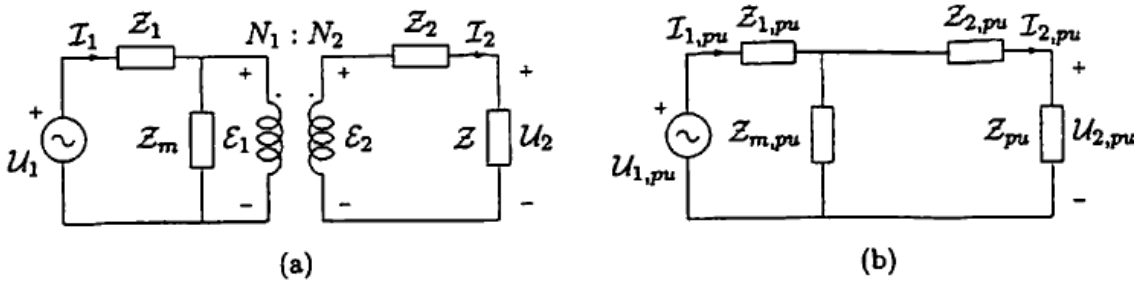


Fig. 18- Ejemplo de la ventaja del uso de valores por unidad. Fuente: [4]

La figura (a) se corresponde con el modelo de la Figura 14, donde se han agrupado las resistencias y reactancias en impedancias. Vamos a demostrar que gracias al uso de valores por unidad se puede llegar al circuito (b), donde no existe el acoplamiento del transformador y, por tanto, se elimina la relación de transformación, simplificando enormemente los cálculos.

Se seleccionan como valores base la potencia S_B, la tensión base U_{B1} para el lado primario y la tensión base U_{B2} para el lado secundario, manteniendo la relación de transformación:

$$\frac{U_{B1}}{U_{B2}} = \frac{\varepsilon_1}{\varepsilon_2} = \frac{N_1}{N_2} = \text{rt} \qquad (1)$$

Las intensidades base quedan definidas entonces:

$$I_{B1} = \frac{S_B}{U_{B1}}$$

$$I_{B2} = \frac{S_B}{U_{B2}}$$

Y las impedancias base:

$$Z_{B1} = \frac{{U_{B1}}^2}{S_B}$$

$$Z_{B2} = \frac{{U_{B2}}^2}{S_B}$$

Aplicando estas relaciones a las ecuaciones obtenidas de utilizar la Ley de Tensiones de Kirchhoff en ambos lados del transformador:

$$U_1 = I_1 \cdot Z_1 + \varepsilon_1 \quad ; \quad \varepsilon_2 = I_2 \cdot Z_2 + U_2$$

$$\downarrow$$

$$\frac{U_1}{U_{B1}} = \frac{I_1}{I_{B1}} \cdot \frac{Z_1}{Z_{B1}} + \frac{\varepsilon_1}{U_{B1}}$$

$$\frac{\varepsilon_2}{U_{B2}} = \frac{I_2}{I_{B2}} \cdot \frac{Z_2}{Z_{B2}} + \frac{U_2}{U_{B2}}$$

Y teniendo en cuenta (1),

$$\frac{\varepsilon_1}{U_{B1}} = \frac{\varepsilon_2}{U_{B2}}$$

Se llega finalmente a la siguiente expresión:

$$U_{1,pu} = I_{1,pu} \cdot Z_{1,pu} + I_{2,pu} \cdot Z_{2,pu} + U_{2,pu}$$

Que responde al modelo de la figura (b), en el que no existe el acoplamiento del transformador. La gran ventaja de usar los valores por unidad es que no se requieren cálculos para referir una impedancia de un lado del transformador a otro, ya que lo valores de las impedancias ($Z_{1,pu}$, $Z_{2,pu}$) son iguales vistos desde cualquier lado del transformador.

3.4.3 Transformador regulable

Los transformadores regulables son de especial interés en la explotación de redes eléctricas porque estos transformadores tienen la capacidad de modificar, en carga o desenergizados, el módulo o ángulo de los voltajes de línea, dentro de pequeños márgenes **[4]**.

Casi todos los transformadores tienen derivaciones en los devanados para ajustar la relación de transformación. Sus circuitos especiales permiten hacer los cambios sin interrumpir la corriente.

Por tanto, los transformadores de regulación, desde el punto de vista de la variable a controlar, pueden ser de dos clases:

- Transformadores de regulación de tensión
- Transformadores de regulación de ángulo o fase

En este apartado solo se van a describir solamente los transformadores de regulación de tensión, pues en la red de distribución radial que se estudiará más adelante basta con tener en cuenta solo este tipo de transformadores. No es necesario tener en cuenta el desfase que introducen los transformadores, dado que basta con aplicar el giro correspondiente una vez obtenidos los resultados.

La regulación de tensión se consigue añadiendo cambiadores de tomas bajo carga a un transformador normal, que modifican el número de espiras del devanado de alta tensión.

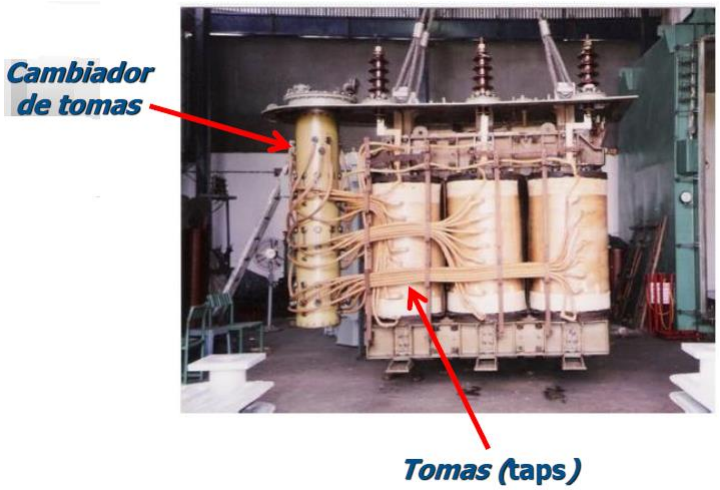


Fig. 19- Transformador regulable con tomas. Fuente: [7]

El esquema más sencillo consiste en un transformador donde uno de sus devanados tiene varias tomas correspondientes a distintos números de espiras. De esta manera, el transformador realiza la modificación del módulo de la tensión, supuesta fija U_p, mediante la modificación del número de espiras de el lado de alta tensión (N_p) y, por tanto, la variación de relación de espiras N_p/N_s. Teniendo en cuenta esto, un transformador de regulación por tomas responde al siguiente modelo:

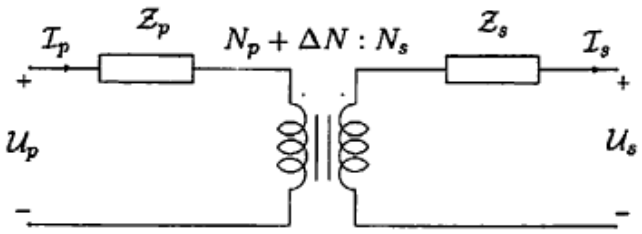


Fig. 20- Circuito de transformador con regulación por tomas. Fuente: [4]

Pasando este circuito a valores por unidad (con tensiones base manteniendo la relación de transformación nominal N_p/N_s), se obtiene el modelo de la Figura 21, donde y_{ik} es la admitancia (inversa de la impedancia 1/Z) de cortocircuito del transformador en p.u., igual a la obtenida sin modificación de tomas, y donde a representa la regulación entre las tensiones primaria y secundaria: $a = 1 + \Delta N_p/N_p$.

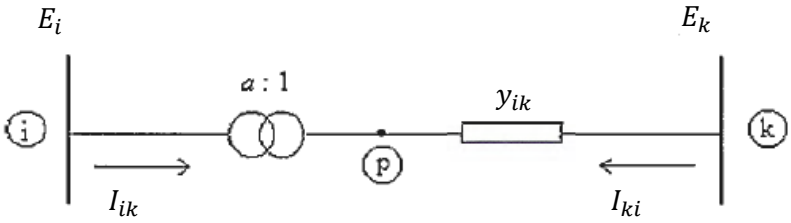


Fig. 21- Circuito de transformador con regulación por tomas en p.u. Fuente: [10]

La admitancia de cortocircuito y_{ik} está del lado del secundario pues es el lado donde la tensión del transformador coincide con la nominal (relación de transformación a:1). Esto se explica simplemente porque del paso del valor de y_{ik} en ohmios al sistema por unidad se utilizaron los valores nominales del transformador.

Se puede obtener un circuito equivalente al de la Figura 21 sin acoplamiento, resultando un modelo más simple. El procedimiento es el siguiente **[10]**:

Conocemos la relación de transformación:

$$\frac{E_i}{E_p} = a$$

De donde se obtiene que

$$E_i = a \cdot E_p \tag{1}$$

El valor de E_p se deduce aplicando la Ley de Tensiones de Kirchhoff al lazo que va desde el secundario del transformador hasta el nodo *k:*

$$E_p = E_k - \frac{I_{ki}}{y_{ik}} \tag{2}$$

Sustituyendo (2) en (1)

$$E_i = aE_k - a\frac{I_{ki}}{y_{ik}} \tag{3}$$

Por otro lado, sabemos que las intensidades están relacionadas a través de la relación de transformación:

$$I_{ki} = -aI_{ik} \tag{4}$$

El signo negativo representa que ambas corrientes tienen sentidos opuestos. El objetivo ahora es obtener expresiones de I_{ik} y I_{ki} que relacionen todos los parámetros del circuito de la Figura 21. Despejando I_{ki} de (3):

$$I_{ki} = y_{ik}E_k - \frac{1}{a}y_{ik}E_i \tag{5}$$

A partir de esta expresión (5) y utilizando la relación (4) llegamos a la expresión para la corriente I_{ik}:

$$I_{ik} = \frac{1}{a^2}y_{ik}E_i - \frac{1}{a}y_{ik}E_k \tag{6}$$

Agrupando en forma matricial (5) y (6)

$$\begin{pmatrix} I_{ik} \\ I_{ki} \end{pmatrix} = \begin{pmatrix} \frac{1}{a^2}y_{ik} & -\frac{1}{a}y_{ik} \\ -\frac{1}{a}y_{ik} & y_{ik} \end{pmatrix} \cdot \begin{pmatrix} E_i \\ E_k \end{pmatrix}$$

La matriz que relaciona las corrientes de las líneas con las tensiones nodales es la matriz de admitancias: $[I] = [Y] \cdot [E]$.

En la matriz $[Y]$, los elementos fuera de la diagonal representan la admitancia (con signo contrario) que hay entre el nodo *i* y el nodo *k*.

A esta admitancia la denominamos

$$\boxed{\mathrm{A} = \frac{1}{a} y_{ik}}$$

Por su parte, en los elementos de la diagonal de la matriz de admitancias:

- El elemento (1,1) representa la suma de todas las admitancias conectadas al nodo *i*. Se deduce que este elemento es la suma de A más otra admitancia que se desconoce por ahora y que está conectada al nodo *i*. A esta admitancia desconocida la llamamos B.
- El elemento (2,2) representa la suma de todas las admitancias conectadas al nodo *k*. Se deduce que este elemento es la suma de A más otra admitancia que se desconoce por ahora y que está conectada al nodo *k*. A esta admitancia desconocida la llamamos C.

Solo falta determinar el valor de las admitancias B y C. De acuerdo con lo expuesto anteriormente:

$$\frac{1}{a^2} y_{ik} = \mathrm{A} + \mathrm{B} \quad \rightarrow \quad \boxed{\mathrm{B} = \frac{1-a}{a^2} y_{ik}}$$

$$y_{ik} = \mathrm{A} + \mathrm{C} \quad \rightarrow \quad \boxed{\mathrm{C} = \frac{a-1}{a} y_{ik}}$$

Con los valores de las admitancias A, B y C puede construirse el circuito equivalente de la Figura 21. Este circuito es el modelo que se utilizará en el caso de estudio para trabajar con los transformadores con tomas. Por la forma que adquiere, a este modelo se le conoce como modelo en π del transformador con tomas.

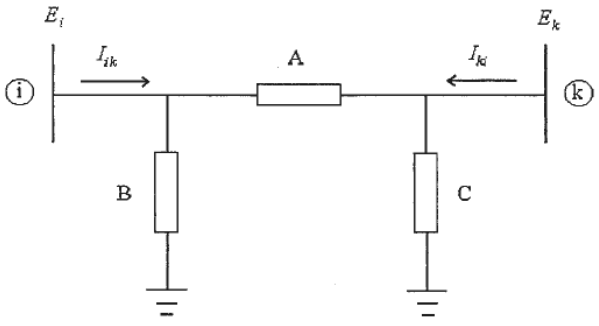


Fig. 22- Modelo en π del transformador con tomas. Fuente: [10]

4. Flujo de cargas

Este trabajo se centra en el cálculo de flujo de cargas de una red de distribución sintética. Hasta el momento, se han ido introduciendo los conceptos necesarios para abordar el futuro análisis de esta red, empezando por la descripción de las redes de distribución hasta llegar a los modelos eléctricos de los elementos que la componen (líneas y transformadores). El último paso antes de comenzar el caso de estudio es introducir el problema de flujo de cargas.

El problema conocido como *flujo de cargas* (*load flow* o *power flow* en inglés) consiste en obtener las condiciones de funcionamiento de un sistema eléctrico de potencia en régimen permanente.

El flujo de cargas consiste básicamente en obtener las tensiones complejas (módulo y ángulo de fase $\rightarrow V\angle\theta$) en todas las barras o nudos eléctricos de la red. A partir de estos resultados se pueden calcular inmediatamente todas las magnitudes de interés, como los flujos de potencia activa y reactiva que fluyen en cada línea, corrientes circulantes, caídas de tensión, pérdidas, etc.

En la operación diaria de un sistema eléctrico de potencia, el flujo de cargas constituye la base del análisis de seguridad del sistema. Esta herramienta se ejecuta periódicamente para identificar posibles problemas de sobrecargas o tensiones inaceptables en la red, como consecuencia de la evolución de la carga, o cuando ocurre algún cambio brusco en la topología de la red [4].

4.1 Formulación del problema

En capítulos anteriores se mencionó que las subestaciones de los sistemas eléctricos de potencia hacían de enlace entre las centrales de generación y los consumidores finales. Estos consumidores demandan o consumen potencia de la red eléctrica. Obviamente, estas potencias no son constantes, sino que varían en cada instante de tiempo. Los generadores (centrales de generación) de la red eléctrica son los encargados de inyectar a la red la potencia demandada por los consumidores.

Debido a la naturaleza cambiante de la demanda surge la necesidad de encontrar en cada instante las tensiones en cada nudo o barra de la red (generalmente subestaciones) y el

flujo de potencia a través de las líneas eléctricas, es decir, la condición de funcionamiento en cada instante de la red eléctrica. Esto es lo que se conoce como flujo de cargas. La siguiente figura da cuenta de la naturaleza cambiante de la demanda. Se trata del perfil de potencias demandadas de un día concreto en el sistema eléctrico español peninsular, información que se puede consultar en tiempo real en la página web de Red Eléctrica.

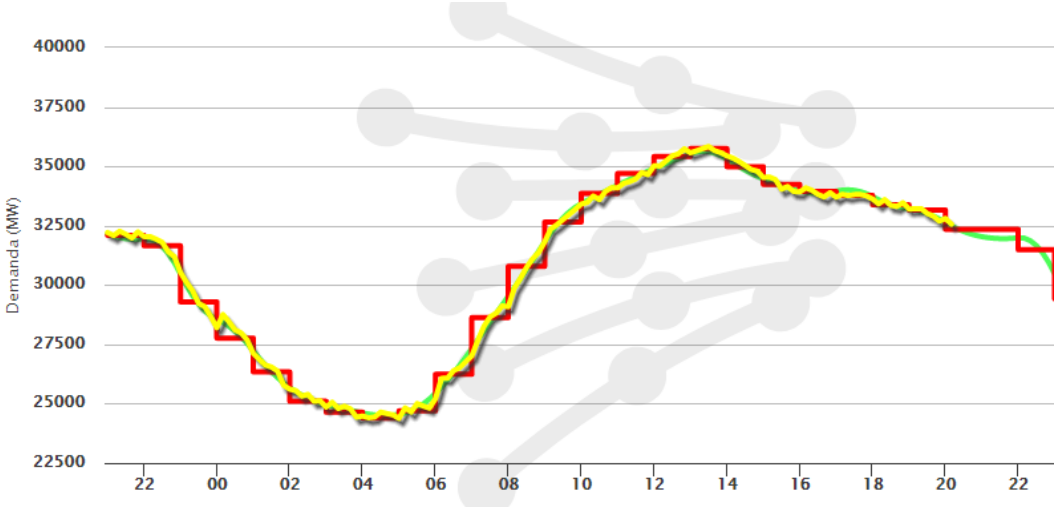


Fig. 23- Perfil de cargas diario del sistema eléctrico peninsular. Fuente: [11]

Aunque los sistemas eléctricos de potencia son básicamente lineales, su estudio mediante herramientas de análisis circuital convencionales es difícil, ya que la información se presenta en forma de potencias, no de impedancias (cargas) y fuentes de tensión (generadores).

De aquí en adelante se van a deducir las ecuaciones del flujo de cargas, de forma que se consiga construir un modelo matemático de la red. Algunas consideraciones previas que debe tenerse en cuenta es que el flujo de cargas es un estudio en régimen permanente, donde se asume que las potencias demandadas son conocidas y sus valores constantes.

Empezamos considerando una red eléctrica pequeña con tres nodos, tal como se muestra en la siguiente figura. En el nodo 1 hay un generador que suministra a la red una potencia activa P_{G1} y una potencia reactiva Q_{G1}. En el nodo 3 también hay un generador que inyecta a la red una potencia P_{G3} y P_{G3}. En este nodo hay a su vez un consumo de potencia, es decir, hay una carga que se está demandando potencia activa P_{D3} y reactiva Q_{D3}. De la misma manera, en el nodo 2 hay una carga conectada que demanda potencia activa y reactiva (P_{D2} , Q_{D2}). Los nodos se encuentran conectados entre así a través de las líneas eléctricas L_1, L_2 y L_3.

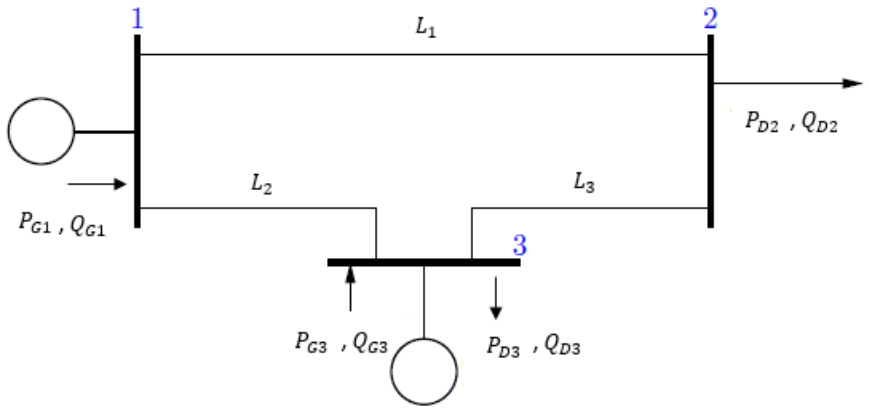


Fig. 24- Ejemplo de pequeña red eléctrica con tres nodos. Fuente: Elaboración Propia

Queda de manifiesto que en un mismo nodo puede existir tanto demanda como inyección de potencia. Si ahora se considera un nodo aislado *i* de una red cualquiera para el estudio del flujo de potencias se puede observar que en cada nodo concreto la potencia inyectada a la red es la resta algebraica de la potencia inyectada menos la potencia demandada en ese nodo (ver Figura 25).

$$P_i = P_{Gi} - P_{Di}$$

$$Q_i = Q_{Gi} - Q_{Di}$$

Además, la potencia inyectada en cada nodo (P_i , Q_i) se reparte entre el resto de los nodos (*k*, *j*, *m*) a los que se encuentre conectado a través de líneas eléctricas.

$$P_i = P_{ik} + P_{ij} + P_{im} \quad , \quad Q_i = Q_{ik} + Q_{ij} + Q_{im}$$

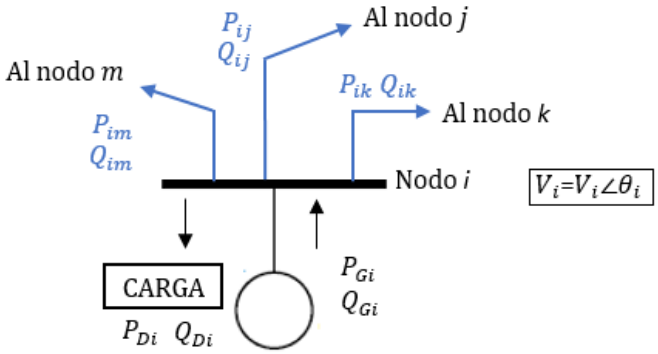


Fig. 25- Ejemplo de flujo de potencia en un nodo concreto de la red. Fuente: Elaboración Propia

Se puede concluir entonces diciendo que la potencia inyectada en el nodo *i* de una red eléctrica atiende a la siguiente expresión:

(1)

$$S_i = V_i \cdot I_i^* = P_i + jQ_i$$

En la que se ha introducido un nuevo término S_i, que es la potencia compleja, la cual engloba a la potencia activa P_i y reactiva Q_i, definidas anteriormente como:

$$P_i = P_{Gi} - P_{Di}$$
$$Q_i = Q_{Gi} - Q_{Di} \quad (2)$$

Antes de continuar con la formulación matemática de las ecuaciones del flujo de cargas es necesario introducir primero la matriz de admitancias de barras [Y]. Las características de esta matriz son las siguientes:

- La matriz [Y] modela la interconexión de las distintas barras o nodos del sistema.
- Sólo incluye las líneas y los nodos del sistema.
- No incluye las admitancias de los generadores ni las cargas (modeladas estas últimas como potencias constantes).
- La dimensión de la matriz [Y] es (*N* x *N*), siendo *N* el número de nodos de la red.
- [Y] es una matriz simétrica.
- [Y] es una matriz dispersa (hasta el 90-95% de sus elementos pueden llegar a ser 0), pues no todas las barras del sistema están conectadas entre sí.
- Los elementos de la diagonal (i=k) Y_{ii} se obtienen como la suma algebraica de todas las admitancias conectadas al nodo *i*.
- Los elementos fuera de la diagonal ($i \neq k$) $Y_{ik} = Y_{ki}$ se corresponden con las admitancias, de signo contrario, conectadas entre el nodo *i* y el nodo *k*.

Para demostrar cómo se construye esta matriz retomamos la red de la Figura 24. Para este ejemplo las líneas eléctricas de esta red se van a modelar de acuerdo con el modelo en π para líneas de mediana longitud (80km < L < 240 km). En el caso de líneas cortas se despreciarían las admitancias en paralelo, siendo el cálculo menos laborioso.

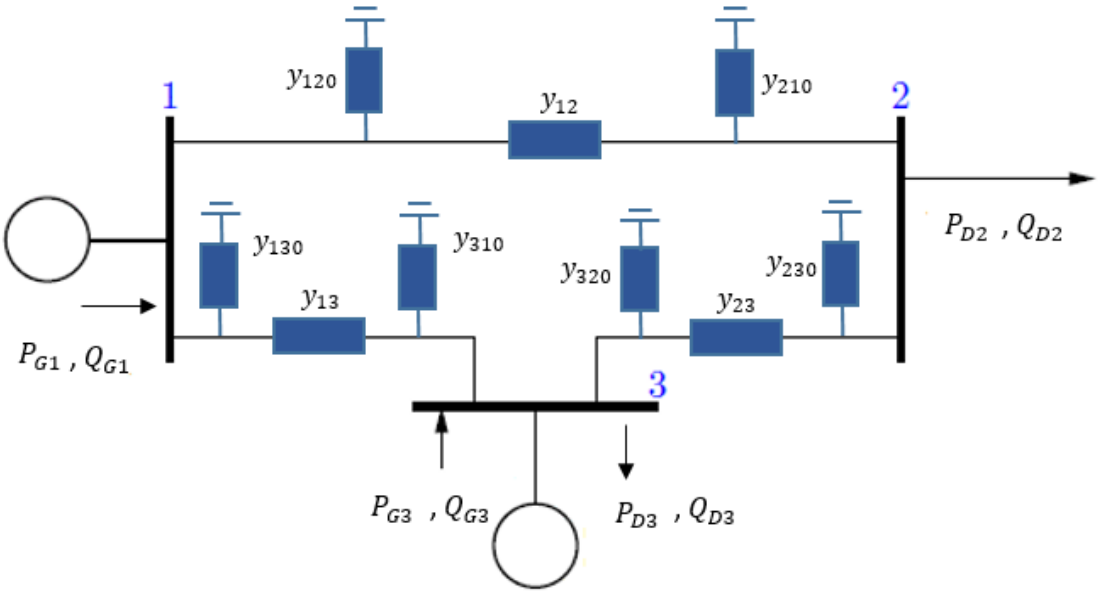


Fig. 26- Modelo de la red de la Fig.24 para el cálculo de la matriz de admitancias. Fuente: Elaboración Propia

Si imaginamos las potencias inyectadas como corrientes inyectadas y aplicamos la Ley de Corrientes de Kirchoff en cada nodo se obtienen las siguientes ecuaciones:

$$I_1 = y_{120}V_1 + y_{12}(V_1 - V_2) + y_{130}V_1 + y_{13}(V_1 - V_3)$$

$$I_2 = y_{210}V_2 + y_{12}(V_2 - V_1) + y_{230}V_2 + y_{23}(V_2 - V_3)$$

$$I_3 = y_{310}V_3 + y_{13}(V_3 - V_1) + y_{320}V_3 + y_{23}(V_3 - V_2)$$

Reagrupando términos y expresando las ecuaciones en forma matricial:

$$\begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix} = \begin{pmatrix} (y_{120} + y_{12} + y_{130} + y_{13}) & -y_{12} & -y_{13} \\ -y_{21} & (y_{210} + y_{12} + y_{230} + y_{23}) & -y_{23} \\ -y_{31} & -y_{32} & (y_{310} + y_{13} + y_{320} + y_{23}) \end{pmatrix} \cdot \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix}$$

Al ser simétrica, los términos y_{12} y y_{21} tienen el mismo valor. Lo mismo ocurre con el resto de los elementos fuera de la diagonal de la matriz.

Esta estructura responde a [I] = [Y]·[V], y se puede reescribir de la siguiente manera:

$$\begin{pmatrix} I_1 \\ I_2 \\ I_3 \end{pmatrix} = \begin{pmatrix} Y_{11} & Y_{12} & Y_{13} \\ Y_{21} & Y_{22} & Y_{23} \\ Y_{31} & Y_{32} & Y_{33} \end{pmatrix} \cdot \begin{pmatrix} V_1 \\ V_2 \\ V_3 \end{pmatrix} \qquad (3)$$

Estando formados los elementos de la diagonal ($i = k$) por la suma de todas las admitancias conectadas al nodo *i* (donde *i* son las filas de la matriz y *k* las columnas):

- $Y_{11} = y_{120} + y_{12} + y_{130} + y_{13}$
- $Y_{22} = y_{210} + y_{12} + y_{230} + y_{23}$
- $Y_{33} = y_{310} + y_{13} + y_{320} + y_{23}$

Los elementos fuera de la diagonal ($i \neq k$) están formados por las admitancias conectadas entre el nodo *i* y el nodo *k*, con signo contrario.

- $Y_{12} = Y_{21} = -y_{12}$
- $Y_{13} = Y_{31} = -y_{13}$
- $Y_{23} = Y_{32} = -y_{23}$

Después de introducir la matriz de barras [Y] continuamos con la obtención de las ecuaciones del flujo de cargas. Las ecuaciones en forma matricial [I] = [Y]·[V] recogidas en (3) se pueden escribir en forma de sumatorio:

$$I_i = \sum_{k=1}^{N} Y_{ik} \cdot V_k \qquad (4)$$

Donde N es el número de barras o nodos del sistema considerado. Esta ecuación expresa la corriente inyectada en el nodo *i*. Ahora se retoma la expresión de la potencia compleja de (1):

$$S_i = V_i \cdot I_i^* = P_i + jQ_i$$

Reordenando términos y sustituyendo el término de la corriente (4) en la expresión de la potencia compleja se llega a lo siguiente:

$$P_i + jQ_i = V_i \cdot I_i^* = V_i \cdot \left[\sum_{k=1}^{N} Y_{ik} \cdot V_k\right]^* \qquad k = 1, 2, \ldots, N$$

Si se continúa desarrollando esta expresión se llega finalmente a las ecuaciones de flujo de cargas. En el ANEXO I se muestra el procedimiento para obtener estas ecuaciones:

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

$$Q_i = |V_i| \sum_{k=1}^{N} |V_k| \ (G_{ik} \sin(\theta_{ik}) - B_{ik} \cos(\theta_{ik})) \qquad (5)$$

k =1, 2, …, N

Son las ecuaciones del flujo de cargas en su forma polar. Representan la suma compleja de las potencias de los *k* nodos conectadas al nodo *i*. Es decir, proporcionan la potencia inyectada en el nodo *i*, la cual fluye a través de las *k* líneas conectadas a este nodo.

Siendo:

N: Número de barras o nudos de las que consta el sistema

$\boldsymbol{Y_{ik}}$: Admitancia entre las barras *i* y *k* (elemento Y_{ik} de la matriz [Y])

$$Y_{ik} = G_{ik} + jB_{ik}$$

$\boldsymbol{G_{ik}}$: Conductancia o parte real de la admitancia entre las barras *i* y *k* (parte real del elemento Y_{ik} de la matriz [Y])

$\boldsymbol{B_{ik}}$: Susceptancia o parte imaginaria de la admitancia entre las barras *i* y *k* (parte imaginaria del elemento Y_{ik} de la matriz [Y])

$\boldsymbol{\theta_{ik}}$: Ángulo formado entre las tensiones de la barra *i* y la barra *k* ($\theta_i - \theta_k$)

Las ecuaciones del flujo de cargas presentan ciertas características:

- Son ecuaciones algebraicas, pues se considera el sistema eléctrico de potencia como un sistema estático en el que se calculan las potencias en un instante dado. Es decir, no existen derivadas ni cambios en el tiempo en las ecuaciones.
- Son ecuaciones no lineales que requieren métodos iterativos para su resolución. En efecto, en las ecuaciones están presentes senos y cosenos que convierten las ecuaciones en no lineales.
- La potencia de los nodos P, Q se expresan en términos de V, θ y de los elementos de la matriz [Y].

Estas ecuaciones de potencia proporcionan las potencias complejas en cada nodo del sistema. Como se mencionó al inicio de este apartado, las potencias generadas y demandadas en el sistema son valores conocidos y que, por tanto, tienen que coincidir con los valores proporcionados por las ecuaciones (5). A estas potencias conocidas se les llama potencias especificadas y se denotan con un superíndice:

$$P_i^{esp} = P_{Gi}^{esp} - P_{Di}^{esp}$$

$$Q_i^{esp} = Q_{Gi}^{esp} - Q_{Di}^{esp}$$

La potencia específica del nodo i es la resta algebraica de la potencia específica generada y la potencia específica demandada. Como el requisito es que $P_i^{esp} = P_i$ y $Q_i^{esp} = Q_i$, se debe cumplir

$$P_i^{esp} - P_i = 0$$

$$Q_i^{esp} - Q_i = 0$$

Siendo P_i y Q_i las ecuaciones de flujo de cargas (5):

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

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

k =1, 2, …, N

Atendiendo a estas ecuaciones de potencia, se observa que cada nudo aporta dos ecuaciones (P_i , Q_i) y cuatro incógnitas (V_i , V_k , θ_i , θ_k), por lo que deben especificarse dos magnitudes por nudo para que las ecuaciones anteriores puedan resolverse. En función de las condiciones de contorno impuestas, se distinguen dos tipos principales de nudos **[4]**:

- Nudos de consumo o nudos PQ: son nudos donde se conoce el consumo de potencia activa (P_{Di}^{esp}) y reactiva (Q_{Di}^{esp}), siendo nula la potencia generada ($P_{Gi} = Q_{Gi} = 0$). Las restricciones son, por tanto,

$$P_i^{esp} = \cancel{P_{Gi}^{esp}} - P_{Di}^{esp} = - P_{Di}^{esp}$$
$$Q_i^{esp} = \cancel{Q_{Gi}^{esp}} - Q_{Di}^{esp} = - Q_{Di}^{esp}$$

  quedando como incógnitas los dos módulos y ángulos de la tensión nodal respectiva (V_i , V_k , θ_i , θ_k). La gran mayoría de los nudos de una red, sobre todo en niveles de menor tensión, son de este tipo.

- Nudos de generación o nudos PV: son nudos donde un generador regula la tensión a un valor especificado (V_i^{esp}) e inyecta una potencia activa (P_{Gi}^{esp}) determinada previamente por consideraciones económicas. Las restricciones resultantes, que tienen en cuenta el posible consumo local, son:

$$P_i^{esp} = P_{Gi}^{esp} - P_{Di}^{esp}$$
$$V_i = V_i^{esp}$$

  quedando Q_i y θ_i como incógnitas.

Ahora bien, si únicamente se consideran ambos tipos de nudos, todas las potencias activas inyectadas deberían especificarse de antemano, lo cual es imposible porque las pérdidas de potencia en la red, que también deben ser aportadas por los generadores, no se conocen hasta que se obtienen los flujos de potencia por cada elemento del sistema eléctrico de potencia. Por tanto, la potencia activa de al menos un generador no puede ser especificada y debe calcularse al final del proceso. Afortunadamente, esta incógnita adicional se compensa con el hecho de que, cuando se trabaja con fasores, uno de los ángulos de fase puede tomarse libremente como origen de fases. Para simplificar los cálculos, se toma como

origen de fases precisamente el nudo de generación cuya potencia se deja libre, sin asignar. Este nudo, que suele ser un generador importante con capacidad para regular frecuencia, se denomina nudo *slack*.

A modo de resumen, en la siguiente tabla se recogen los valores conocidos y las incógnitas de cada tipo de nudo.

Tipo de Barra	Conocidos	desconocidos
Slack	$\|V\|,\ \theta$	$P,\ Q$
de Generación	$P,\ \|V\|$	$Q,\ \theta$
de Carga	$P,\ Q$	$\|V\|,\ \theta$

Tabla 2. Datos conocidos e incógnitas en el cálculo del flujo de cargas. Fuente: [7]

Si se considera que n_D es el número de nodos de consumo (PQ), entonces el número de nodos de generación, sin contar el nudo slack, será $n_G = N - n_D - 1$, siendo *N* el número total de nodos. En base a la clasificación de los nudos realizada anteriormente, las ecuaciones que intervienen en el problema de flujo de cargas (5) se plantearán en los siguientes nudos:

$$P_i = |V_i| \sum_{k=1}^{N} |V_k| \ (G_{ik} \cos(\theta_{ik}) + B_{ik} \sin(\theta_{ik})) \qquad k = 1, 2, \ldots, N$$

$$i = 2, \ldots, N$$

Suponiendo que el nudo slack es el primero i=1

$$Q_i = |V_i| \sum_{k=1}^{N} |V_k| \ (G_{ik} \sin(\theta_{ik}) - B_{ik} \cos(\theta_{ik})) \qquad k = 1, 2, \ldots, N$$

$$i = 2, \ldots, n_D$$

Es decir, las ecuaciones de potencia activa se plantean en todos los nudos menos en el nudo slack, mientras que las potencias reactivas se plantean en todos los nudos de consumo. Dicho de otra manera, los nudos de generación (PV) aportan una ecuación, sin incluir el nudo slack, y los nudos de consumo (PQ) aportan dos ecuaciones.

La solución del problema de flujo de cargas consiste en encontrar los desfases θ_i de todos los nudos menos el slack (i = 2, …, N) y los módulos de tensiones V_i en todos los nudos de generación sin incluir el slack (i = 2, …, n_D). Para mayor claridad la Tabla 2

representa un buen resumen para saber los nudos en los que se deben plantear las ecuaciones de potencia. Una vez conocidos los fasores de tensión en todas las barras es posible calcular el resto de las variables de la red.

Al fijar la tensión compleja (módulo y ángulo) del nudo slack y dejar libre su potencia compleja, implica que las dos ecuaciones respectivas de este nudo (P y Q) no intervienen en el proceso de cálculo. Estas ecuaciones servirán una vez resuelto el problema para hallar precisamente la potencia compleja de dicho nudo.

Dado que las ecuaciones de potencia no son lineales, su solución requiere inevitablemente de métodos iterativos, por lo que es necesario adoptar unos valores iniciales adecuados para las variables del problema (V_i y θ_i) que hagan converger el proceso iterativo hacia un punto físicamente viable.

Debido a las características especiales del problema de flujo de cargas, donde se sabe de antemano que las tensiones se mueven en una banda relativamente estrecha alrededor de su valor nominal, y que los desfases entre nudos adyacentes se mueven en márgenes pequeños por motivos de estabilidad, hacen que el denominado *perfil plano* sea casi siempre la mejor opción para iniciar el proceso iterativo. Dicho perfil consiste en utilizar como valores iniciales $\theta_i^0 = 0$ para todos los nudos y $V_i^0 = 1$ p.u. para todos los nudos de consumo.

4.2 Método de Newton-Raphson

El método de Newton-Rapshon se basa en ir obteniendo sucesivamente nuevos valores mediante aproximaciones de primer orden de las funciones no lineales involucradas.

$$y = f(x_1, x_2) = \begin{bmatrix} f_1(x_1, x_2) \\ f_2(x_1, x_2) \\ \vdots \\ f_N(x_1, x_2) \end{bmatrix}$$

Las funciones involucradas $f(x_1, x_2)$ son las ecuaciones de potencia (5) y las incógnitas del problema de flujo de cargas son las tensiones complejas en cada barra (V_i y θ_i). Este sistema de ecuaciones con dos incógnitas se puede recoger en forma matricial:

$$y = \begin{bmatrix} P(x_1, x_2) \\ Q(x_1, x_2) \end{bmatrix} = \begin{bmatrix} P_2(x_1, x_2) \\ \vdots \\ P_N(x_1, x_2) \\ Q_2(x_1, x_2) \\ \vdots \\ Q_N(x_1, x_2) \end{bmatrix} \quad ; \quad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \theta \\ V \end{bmatrix} = \begin{bmatrix} \theta_2 \\ \vdots \\ \theta_N \\ V_2 \\ \vdots \\ V_N \end{bmatrix} \tag{1}$$

Las funciones y las incógnitas están expresadas para todos los nudos del sistema sin incluir la barra slack, que se considera el primer nodo. Por eso los subíndices van de 2, …, N, siendo N el número de nodos del sistema y P_i , Q_i las ecuaciones de potencia:

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

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

Asumiendo ahora una estimación inicial de la solución del sistema de ecuaciones $x_1^{(0)}$, $x_2^{(0)}$ y expandiendo en Series de Taylor una función cualquiera $f(x_1, x_2)$:

$$y = f\left(x_1^{(0)}, x_2^{(0)}\right) + \frac{df}{dx_1}\Big|_{x_1^{(0)}} \left(x - x_1^{(0)}\right) + \frac{df}{dx_2}\Big|_{x_2^{(0)}} \left(x - x_2^{(0)}\right) + \cdots +$$

Despreciando las derivadas de orden mayor que 2 y reordenando términos:

$$y - f\left(x_1^{(0)}, x_2^{(0)}\right) = \frac{df}{dx_1}\Big|_{x_1^{(0)}} \left(x - x_1^{(0)}\right) + \frac{df}{dx_2}\Big|_{x_2^{(0)}} \left(x - x_2^{(0)}\right)$$

Escribiendo en forma matricial:

$$[\Delta y^0] = \left[\left.\frac{df}{dx_1}\right|_{x_1^{(0)}} \quad \left.\frac{df}{dx_2}\right|_{x_2^{(0)}}\right] \cdot \begin{bmatrix}\Delta x_1^{(0)} \\ \Delta x_2^{(0)}\end{bmatrix} \tag{2}$$

Donde se han definidos tres nuevos términos, el jacobiano [J], el vector de correcciones que indican la diferencia entre la solución exacta (x_1, x_2) y la solución aproximada $x_1^{(0)}, x_2^{(0)} \rightarrow [\Delta x]$, y $[\Delta y]$ que representa la diferencia entre el valor exacto de la función y el obtenido al evaluar las funciones en $x_1^{(0)}, x_2^{(0)}$.

$$[\Delta y^0] = y - f\left(x_1^{(0)}, x_2^{(0)}\right)$$

$$J^{(0)} = \left[\left.\frac{df}{dx_1}\right|_{x_1^{(0)}} \quad \left.\frac{df}{dx_2}\right|_{x_2^{(0)}}\right]$$

$$\begin{bmatrix}\Delta x_1^{(0)} \\ \Delta x_2^{(0)}\end{bmatrix} = \begin{bmatrix}\left(x - x_1^{(0)}\right) \\ \left(x - x_2^{(0)}\right)\end{bmatrix}$$

Por lo que podemos generalizar y reescribir el sistema de ecuaciones (2) para la iteración *k-ésima* :

$$[\Delta y]^{(k)} = [J]^{(k)} \cdot \begin{bmatrix}\Delta x_1 \\ \Delta x_2\end{bmatrix}^{(k)} \tag{3}$$

Teniendo en cuenta las expresiones de potencia de (1), se puede reescribir (3) para obtener las expresiones matriciales que permiten resolver el problema de flujos de carga del sistema de potencia, para la iteración *k-ésima*:

$$\begin{bmatrix}\Delta P \\ \Delta Q\end{bmatrix}^{(k)} = \underbrace{\begin{bmatrix}H & N \\ M & L\end{bmatrix}^{(k)}}_{\text{Jacobiano [J]}} \begin{bmatrix}\Delta\theta \\ \Delta V/V\end{bmatrix}^{(k)} \tag{4}$$

La primera matriz contiene los residuos de potencia, es decir, las diferencias entre el valor exacto de las potencias inyectadas por las barras ($P_i^{espec.}, Q_i^{espec.}$) y el obtenido al evaluar las funciones de potencia ($P_i\ , Q_i$) en cada iteración [$\theta^{(k)}, V^{(k)}$].

$$\Delta P_i = P_i^{espec.} - P_i$$

$$\Delta Q_i = Q_i^{espec.} - Q_i$$

La segunda matriz del sistema es el Jacobiano, y representa la variación de las potencias activas y reactivas ($P_i\ , Q_i$) con los módulos y ángulos de las tensiones. Se ha dividido el jacobiano en las submatrices *H*, *N*, *M* y *L*. Los términos del jacobiano son los siguientes [4]:

$$H_{ik} = \partial P_i/\partial\theta_k \qquad N_{ik} = V_k\ \partial P_i/\partial V_k$$

$$M_{ik} = \partial Q_i/\partial\theta_k \qquad L_{ik} = V_k\ \partial Q_i/\partial V_k$$

Para i ≠ k	
$H_{ik} = L_{ik} = V_i V_k (G_{ik}\sin\theta_{ik} - B_{ik}\cos\theta_{ik})$ $N_{ik} = -M_{ik} = V_i V_k (G_{ik}\cos\theta_{ik} + B_{ik}\sin\theta_{ik})$	
Para i = k	
$H_{ii} = -Q_i - B_{ii}{V_i}^2$	$L_{ii} = Q_i - B_{ii}{V_i}^2$
$N_{ii} = P_i + G_{ii}{V_i}^2$	$M_{ii} = P_i - G_{ii}{V_i}^2$

Tabla 3. Términos del Jacobiano.

Se observar que la coincidencia de términos entre el Jacobiano y las expresiones de potencia ($\Delta P\ , \Delta Q$) permite un ahorro sustancial de esfuerzo de cálculo.

La última matriz está formada por las diferencias entre el valor de las incógnitas en la iteración *k+1* ($\theta_i^{(k+1)}\ , V_i^{(k+1)}$) y el obtenido en la iteración anterior [$\theta_i^{(k)}, V_i^{(k)}$].

$$\begin{bmatrix} \Delta\theta \\ \Delta V \end{bmatrix}^{(k)} = \begin{bmatrix} \theta \\ V \end{bmatrix}^{(k+1)} - \begin{bmatrix} \theta \\ V \end{bmatrix}^{(k)} \qquad (5)$$

Esta matriz se obtiene resolviendo el sistema (4):

$$\begin{bmatrix} \Delta\theta \\ \Delta V/V \end{bmatrix}^{(k)} = [\,\mathrm{J}\,]^{(k)^{-1}} \begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix}^{(k)}$$

La utilización de $\Delta V/V$ en lugar de ΔV no afecta numéricamente al algoritmo, pero simplifica los términos del jacobiano haciéndolo numéricamente más simétrico. Por ello, cabe resaltar que para obtener la matriz (5) es necesario realizar la siguiente operación:

$$\begin{bmatrix} \Delta\theta \\ \Delta V \end{bmatrix}^{(k)} = \begin{bmatrix} \Delta\theta \\ \Delta V/V \end{bmatrix}^{(k)} \cdot [V]^{(k)}$$

A partir de los resultados del sistema (5) se obtienen los nuevos valores del vector de incógnitas para la siguiente iteración:

$$\begin{bmatrix} \theta \\ V \end{bmatrix}^{(k+1)} = \begin{bmatrix} \theta \\ V \end{bmatrix}^{(k)} + \begin{bmatrix} \Delta\theta \\ \Delta V \end{bmatrix}^{(k)} \qquad (6)$$

Finalmente, podemos concluir que las expresiones matriciales que van a permitir resolver el problema de flujos de cargas son las que se recogen en los sistemas *(4)* y *(6)*.

El proceso iterativo se detendrá cuando se cumpla que los residuos de potencia (ΔP , ΔQ) sean menores que una tolerancia ε suficientemente pequeña, establecida al comienzo del problema.

$$\max\left|\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix}^{(k)}\right| \leq \varepsilon$$

Para valores iniciales $\theta_i^{(0)}, V_i^{(0)}$ próximos a la solución, el método de Newton-Raphson converge cuadráticamente. En el caso particular del flujo de cargas, con independencia del tamaño de la red, el número de iteraciones oscila habitualmente entre 3 y 5 partiendo del perfil plano → $\theta_i^{(0)} = 0$, $V_i^{(0)} = 1$p.u.

A continuación, se indica el algoritmo de cálculo para la resolución del problema de flujo de cargas mediante el método de Newton-Raphson.

Planteamiento del problema: Cálculo del flujo de cargas.

1. Construir el equivalente por fase del sistema con todas las variables y parámetros representados en por unidad en unas bases comunes.
2. Construir la matriz de admitancias de barras [Y].
3. Clasificar las barras.
4. Plantear las ecuaciones de potencia (P_i , Q_i).
5. Calcular los residuos de potencia [ΔP | ΔQ] y los términos del Jacobiano [J].
6. Plantear el sistema de ecuaciones a resolver en forma matricial (4) y (6).
7. Efectuar cálculo iterativo para obtener una solución aproximada mediante el método de Newton-Raphson.
    a. Para la primera iteración inicializar las tensiones no especificadas con un perfil plano ($1\angle 0°$ pu).
    b. Calcular las potencias y los residuos de potencia correspondientes a estos valores iniciales.
    c. Evaluar los elementos del Jacobiano correspondientes a esta iteración.
    d. Resolver el sistema de ecuaciones lineales para obtener [$\Delta\theta$ | $\Delta V/V$].
    e. Utilizar los nuevos valores obtenidos del vector de incógnitas [θ | V] para la siguiente iteración.
    f. Repetir el procedimiento desde el paso b.
    g. Realizar iteraciones hasta que se alcance la convergencia, es decir, que los residuos de potencia sean menores a un error ε, previamente seleccionado.
8. Una vez conocidos los fasores de tensión en todas las barras es posible calcular el resto de variables de red.

5. Caso de Estudio

Mediante la ayuda de software informático se implementará una técnica capaz de resolver el cálculo de flujo de cargas mediante el método de Newton-Raphson en una red de distribución.

Esta técnica será probada en una red de distribución sintética en la que la medida de potencia de algunos de los nodos será obtenida desde dispositivos reales.

Se comenzará modelando la red de distribución para realizar un análisis estático de dicha red en el que se calculará el problema de flujo de cargas en un instante concreto de tiempo para el cual los parámetros de la red adoptan unos valores determinados. Este primer cálculo estático del flujo de cargas se realizará implementando en Matlab el método de Newton-Raphson para el cálculo del flujo de cargas de la red de distribución sintética. Este algoritmo será validado con el programa informático PowerWorld con el fin de comprobar que el modelo empleado de la red proporciona los resultados de flujo de cargas esperados, los cuales ya han sido determinados en **[12]**.

Una vez comprobado la validez del modelo de la red de distribución, se modificará en Matlab el algoritmo de Newton-Raphson. Para simular una situación más cercana a la realidad, ya no se realizará el cálculo de flujo de cargas en un instante determinado de tiempo en el que se conocen los valores puntuales de potencia demandada en los nodos, sino que ahora los valores de las potencias irán fluctuando en el tiempo. Lo que se va a hacer es utilizar una base de datos **[13]** que contiene registros de potencias reales medidas en cada minuto a lo largo de un día completo. Estas potencias se asignarán a los nodos de la red de distribución que se va a estudiar. De esta forma, el algoritmo implementado en Matlab leerá una a una todas las medidas de la base de datos e irá calculando el flujo de cargas para cada una de estas potencias. Concretamente, se calculará el flujo de cargas 1440 veces, que es el número de medidas de potencias en la base de datos (una medida en cada minuto del día).

El algoritmo para el cálculo de flujo de cargas en Matlab se implementará también en lenguaje Python 3. Para los siguientes propósitos de este trabajo se continuará trabajando en Python, pues el objetivo es implementar el algoritmo en la Raspberry Pi.

En este punto el algoritmo estará diseñado para leer las potencias sintéticas de la base de datos anterior. El siguiente paso será comprobar la capacidad del algoritmo desarrollado para funcionar en tiempo real. Para ello, se utilizará el paquete Pymodbus en Python para

leer en tiempo real los datos de potencia de un medidor real. Estas potencias serán asignadas a un nodo concreto de la red, mientras que el resto de los nodos seguirá leyendo las potencias de la base de datos.

5.1 Presentación de la red sintética

La red de distribución sintética sobre la que se va a implementar el algoritmo de Newton-Raphson para resolver el problema de flujo de cargas es una red formada por 9 nodos. Esta red de distribución es una versión simplificada de una red radial de media tensión propiedad de una acería situada en el norte de España, en la que todos los transformadores presentes son transformadores con tomas que permiten regular la tensión del lado de alta tensión [12].

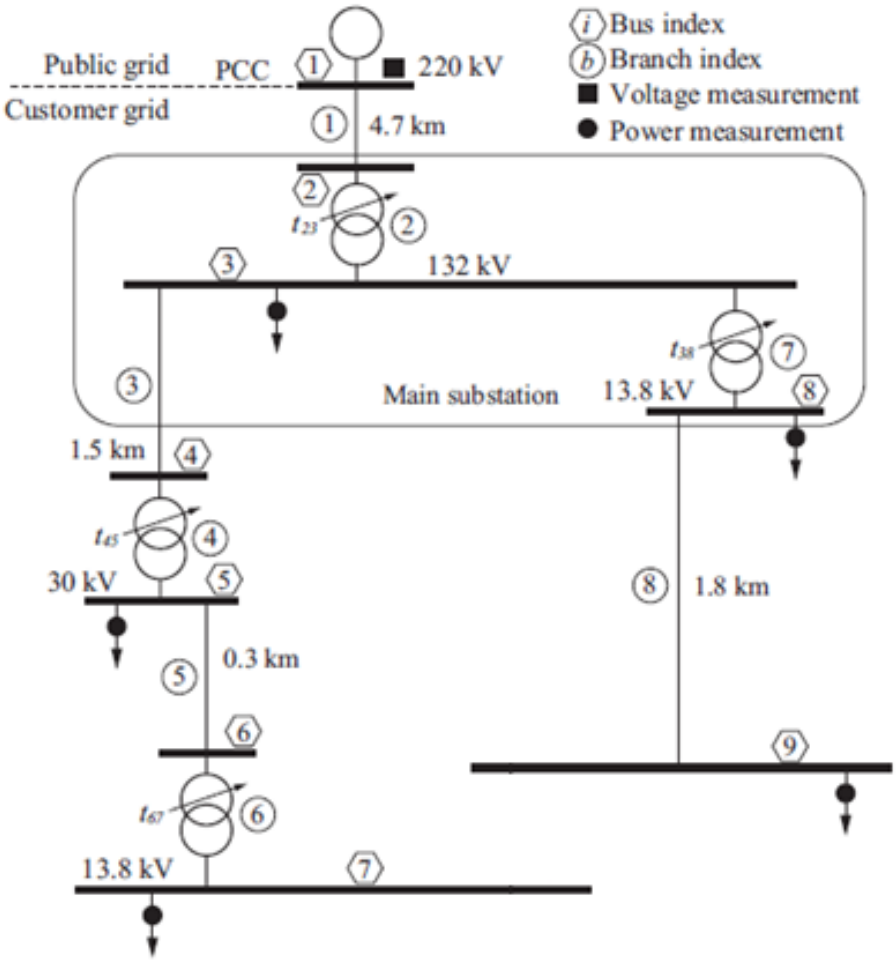


Fig. 27- Red de distribución sintética. Fuente: [12]

En un momento concreto, los parámetros de los elementos de la red son los que se incluyen en las siguientes tablas:

Parámetros del transformador.

Transf. #	S_n [MWA]	R_{sc} [%]	X_{sc} [%]	a [pu]	θ [deg.]
t_{23}	2 x 270	0,90	12,90	1,0125	-30
t_{45}	3 x 37,5	0,90	9,00	0,9875	0
t_{67}	10	0,95	4,80	0,9250	30
t_{38}	3 x 50	0,92	8,50	0,9750	30

Tabla 4. Fuente: [12]

Parámetros de las líneas.

Linea #	Longitud [km]	Z_{linea} [Ω/km]	Z_{linea} [Ω]	Z_{ik} [pu]
1	4,7	0,025 + j0,240	0,1175 + j1,128	2,43e - 4 + j2,331e - 3
2	-	-	-	1,667e - 3 + j2,3889e - 2
3	1,5	0,161 + j0,151	0,2415 + j0,2265	1,386e - 3 + j1,300e - 3
4	-	-	-	8,000e - 3 + j8,0000e - 2
5	0,3	0,568 + j0,133	0,1704 + j0,0399	1,8933e - 2 + j4,433e - 3
6	-	-	-	9,5000e - 2 + j4,80000e - 1
7	-	-	-	6,133e - 3 + j5,6667e - 2
8	1,8	0,161 + j0,112	0,2898 + j0,2016	1,52174e - 1 + j1,05860e - 1

Tabla 5. Fuente: [12]

Potencias inyectadas.

Barra #	Potencia real, P_i [MW]	Potencia reactiva, Q_i [Mvar]
2	0,0	0,0
3	84,0	26,0
4	0,0	0,0
5	34,0	12,0
6	0,0	0,0
7	4,9	12,6
8	52,0	39,0
9	2,7	-3,4

Tabla 6. Fuente: [12]

5.2 Modelización de la red sintética

Para realizar el cálculo de flujo de cargas sobre la red de distribución es necesario modelizar previamente los elementos que forman la red. Los parámetros del modelo obtenidos en este apartado se introducirán en PowerWorld para verificar la calidad del mismo.

Las líneas eléctricas se modelarán de acuerdo con el modelo de línea corta explicado en el capítulo 3.3, en el que solo se tiene en cuenta la impedancia serie de la línea. Para los transformadores con tomas se utilizará el modelo en π expuesto en el capítulo 4.4.3.

De esta forma, se busca un modelo de la red de la Figura 27 que tenga el siguiente aspecto, donde el color verde indica el modelo en π de los transformadores y el color azul representa las impedancias de las líneas eléctricas.

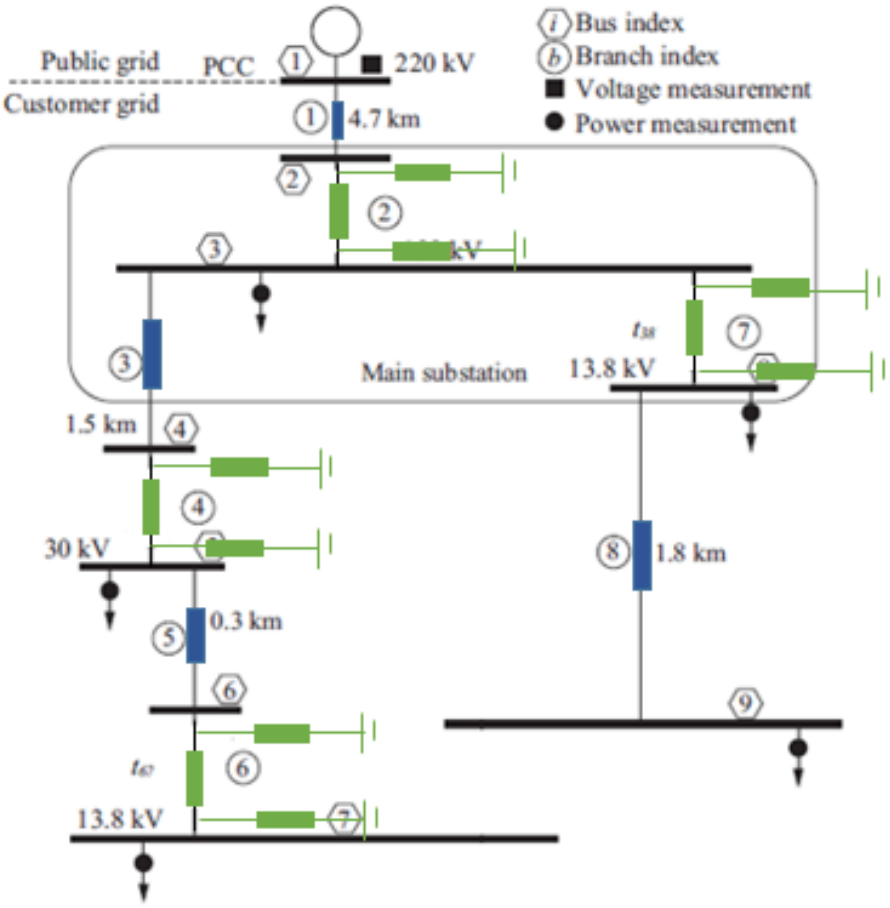


Fig. 28- Modelo de la red de distribución de estudio. Fuente: Elaboración Propia

5.2.1 Selección de las bases del sistema

Antes de modelizar los elementos de la red de distribución se deben seleccionar las bases con las que se van a trabajar para pasar los valores al sistema por unidad.

Se va a utilizar como potencia base común a toda la red $S_B = 100\ MVA$. Se distinguen cuatro zonas con diferentes niveles de tensión delimitadas por los transformadores. Estas zonas se han representado gráficamente en la siguiente figura.

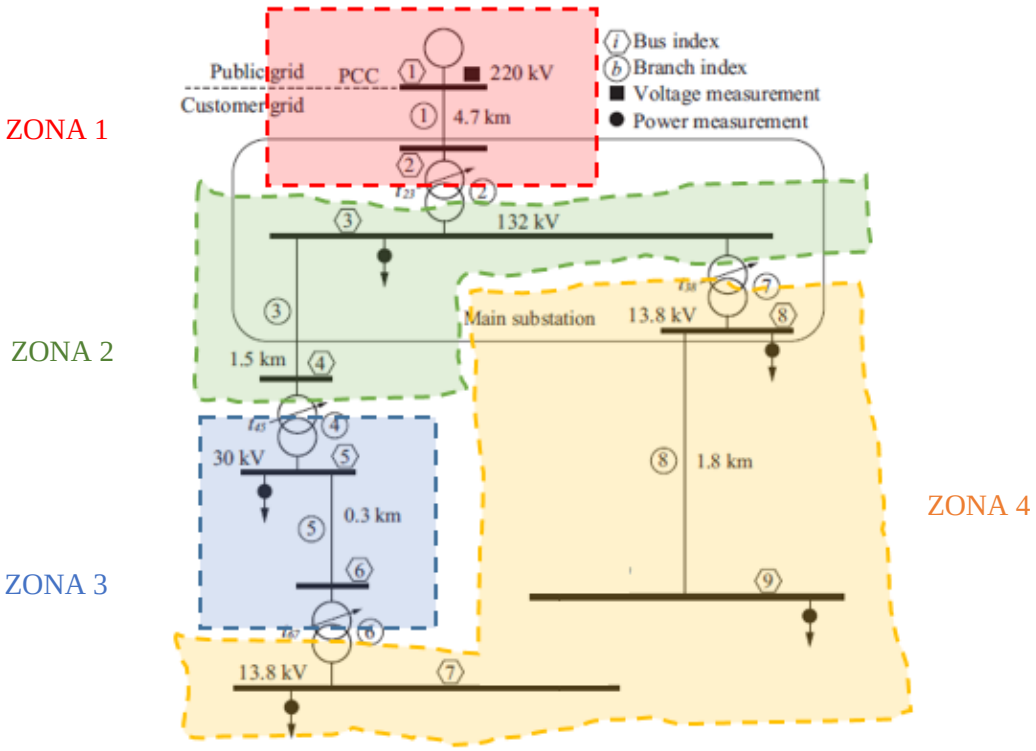


Fig. 29- Zonas de diferente tensión de la red. Fuente: Elaboración Propia

Las tensiones base seleccionadas se hacen coincidir con las tensiones nominales de los transformadores, manteniendo de esta forma la relación de transformación: $U_{B1} = 220kV$, $U_{B2} = 132kV$, $U_{B3} = 30kV$, $U_{B4} = 13{,}8kV$.

Una vez establecidas la potencia y las tensiones base, el cálculo del resto de bases del sistema es inmediato, tal como se explicó en el capítulo 3.1. Los valores de las intensidades e impedancias base de las cuatro zonas de la red se recogen en la siguiente tabla:

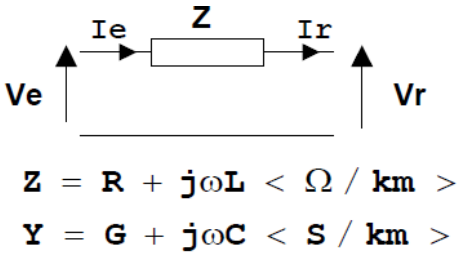
Bases del sistema.

	S_B [MVA]	U_B [kV]	I_B [A]	Z_B [Ω]
Fórmula	-	-	$I_B = \frac{S_B}{\sqrt{3}\,U_B}$	$Z_B = \frac{{U_B}^2}{S_B}$
Zona 1	100	220	262,4319	484,0000
Zona 2	100	132	437,3866	174,2400
Zona 3	100	30	1924,5009	9,0000
Zona 4	100	13,8	4183,6976	1,9044

Tabla 7.

5.2.2 Modelización de las líneas eléctricas

Recordando el modelo de las líneas eléctricas de corta longitud:



Las líneas se modelan como impedancias Z, cuyos valores para la red de distribución se recogen en la tercera columna de la Tabla 5. Para obtener los valores en ohmios de estas impedancias no hay más que multiplicar los valores de las impedancias en Ω/km por los km de la línea. Es decir, multiplicando la segunda y tercera columna de la Tabla 5 se obtienen los valores de la cuarta columna.

Cálculo de los parámetros de cada modelo simplificado.

Equipo	Zona	Modelo	Valores reales
LINEA 1	1	Z_L	$Z_{L1} = (0{,}025 + j0{,}240)\left[\frac{\Omega}{km}\right] \cdot 4{,}7[km] = 0{,}1175 + j1{,}1280\ [\Omega]$
LINEA 3	2	Z_L	$Z_{L3} = (0{,}161 + j0{,}151)\left[\frac{\Omega}{km}\right] \cdot 1{,}5[km] = 0{,}2415 + j0{,}2265\ [\Omega]$
LINEA 5	3	Z_L	$Z_{L5} = (0{,}568 + j0{,}133)\left[\frac{\Omega}{km}\right] \cdot 0{,}3[km] = 0{,}1704 + j0{,}0399\ [\Omega]$
LINEA 8	4	Z_L	$Z_{L8} = (0{,}161 + j0{,}112)\left[\frac{\Omega}{km}\right] \cdot 1{,}8[km] = 0{,}2898 + j0{,}2016\ [\Omega]$

Tabla 8. Cálculo de las impedancias en ohmios de las líneas.

Para obtener los valores en por unidad de las impedancias basta con dividir los valores en ohmios de las impedancias por sus respectivas impedancias bases, según la zona de la red en la que se encuentren las líneas.

Cálculo de los valores en pu.

Equipo	Valores en por unidad
LINEA 1	$Z_{L1\,[pu]} = \frac{Z_{L1}}{Z_{B1}} = \frac{0{,}1175 + j1{,}1280\ [\Omega]}{484\ [\Omega]} = 2{,}43 \cdot 10^{-4} + j2{,}331 \cdot 10^{-3}$
LINEA 3	$Z_{L3\,[pu]} = \frac{Z_{L3}}{Z_{B2}} = \frac{0{,}2415 + j0{,}2265\ [\Omega]}{174{,}24\ [\Omega]} = 1{,}386 \cdot 10^{-3} + j1{,}300 \cdot 10^{-3}$
LINEA 5	$Z_{L5\,[pu]} = \frac{Z_{L5}}{Z_{B3}} = \frac{0{,}1704 + j0{,}0399\ [\Omega]}{9\ [\Omega]} = 1{,}8933 \cdot 10^{-2} + j4{,}433 \cdot 10^{-3}$
LINEA 8	$Z_{L8\,[pu]} = \frac{Z_{L8}}{Z_{B4}} = \frac{0{,}2898 + j0{,}2016\ [\Omega]}{1{,}9044\ [\Omega]} = 1{,}52174 \cdot 10^{-1} + j1{,}05860 \cdot 10^{-1}$

Tabla 9. Cálculo de los valores de las impedancias de las líneas en p.u.

Los datos de esta última tabla son los que se introducirán en PowerWorld para modelar las líneas eléctricas de la red de distribución. En la siguiente imagen se muestra una captura de PowerWorld en la que se están introduciendo los datos de resistencia y reactancia de la línea 1.

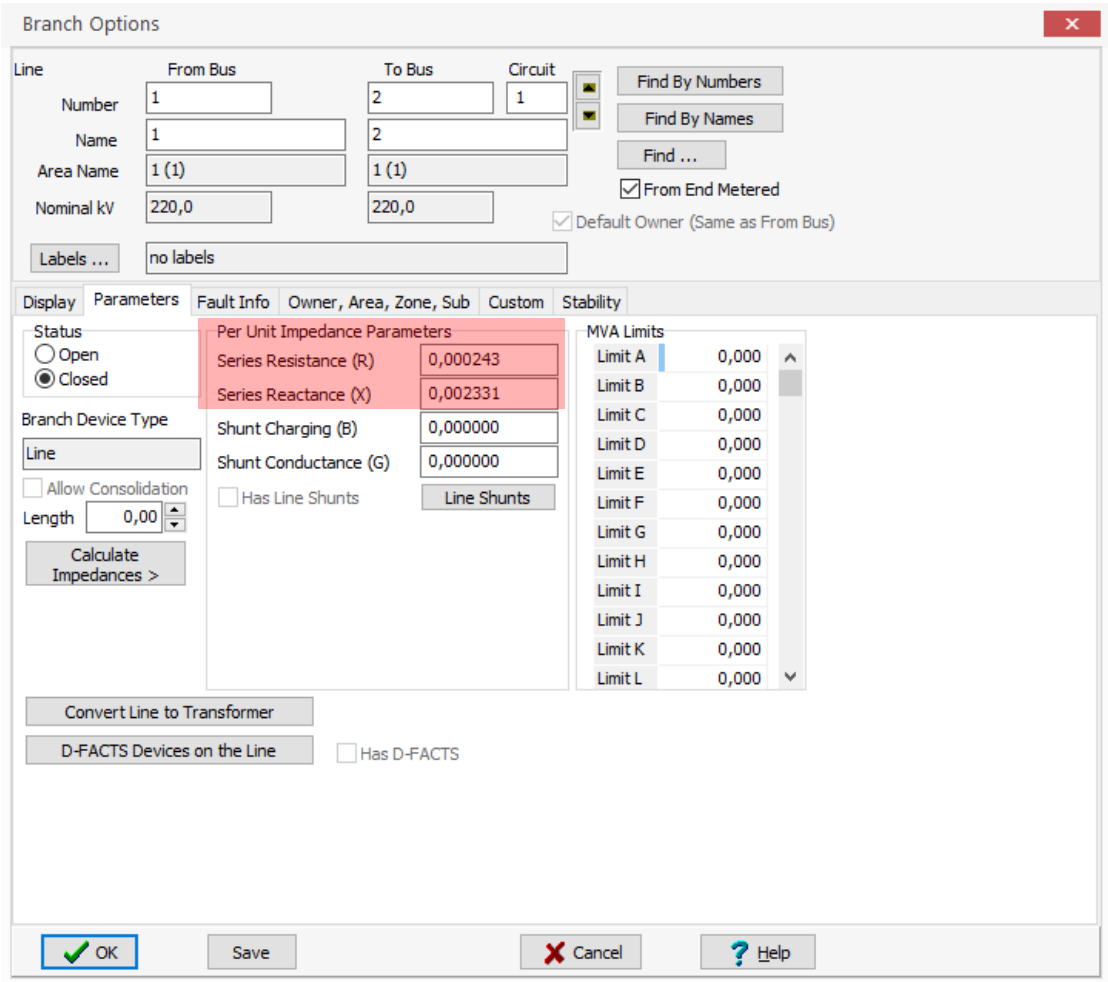
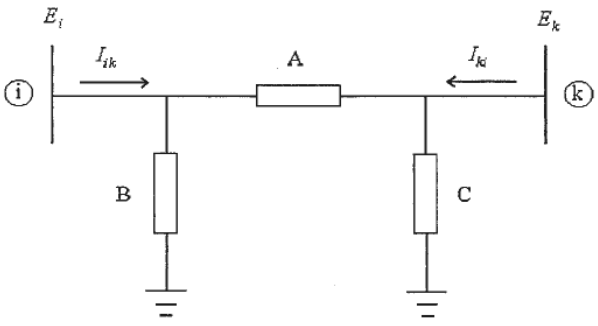


Fig. 30- Ejemplo de datos de líneas introducidos en PowerWorld.

5.2.3 Modelización de los transformadores con tomas

En la red de distribución existen 2 transformadores en la línea 2, 3 transformadores en la línea 4, 1 transformador en la línea 6 y 3 transformadores en la línea 3. Todos ellos son transformadores regulables que responden al modelo en π.



Donde los valores de las admitancias que aparecen en la figura son:

$$\boxed{\mathrm{A} = \frac{1}{a} y_{ik}}$$

$$\boxed{\mathrm{B} = \frac{1-a}{a^2} y_{ik}}$$

$$\boxed{\mathrm{C} = \frac{a-1}{a} y_{ik}}$$

y_{ik} es la inversa de la impedancia de cortocircuito entre el nodo *i* y el nodo *k*. Exceptuando la línea 6 de la red, en la que hay un solo transformador, en el resto de las líneas esta admitancia y_{ik} es la resultante de asociar en paralelo las admitancias de cada uno de los transformadores que haya en cada línea. Por ejemplo, para obtener el modelo en π de la línea 2 (ver Figura 28) es necesario asociar en paralelo las admitancias de los dos transformadores. El valor obtenido (y_{ik}) multiplicado por 1/a se correspondería con el parámetro A del modelo π de esta línea.

Para obtener el modelo en π de los transformadores (ver Figura 28), el programa PowerWorld solo necesita (además de las tomas y los ángulos de fase) la impedancia de cortocircuito de cada transformador individual. Con estos datos el programa calcula automáticamente los parámetros A, B, C del modelo en π.

Sin embargo, para implementar el modelo de la red en Matlab y en Python es necesario introducir las admitancias equivalentes y_{ik} obtenidas de asociar en paralelo los transformadores que hay en cada línea concreta. Por eso en este apartado se va a mostrar el procedimiento para obtener las impedancias de cortocircuito de cada transformador individual (para PowerWorld), así como las impedancias equivalentes de aquellas líneas donde exista más de un transformador (para Matlab y Python).

Los valores de los transformadores (Tabla 4) están referidos a sus valores nominales de placa. Por esa razón, es necesario hacer un cambio de base para adaptar los datos de las resistencias y reactancias de cortocircuito a las bases de nuestra red, tal como se explicó en el capítulo 3.2

Las impedancias base nominales de los transformadores se calculan a partir de sus datos nominales según la expresión:

$$Z_B = \frac{{U_{n2}}^2}{S_n}$$

Se determinan las impedancias base del lado secundario de los transformadores porque lo que nos interesa es llegar a obtener el valor en por unidad de la impedancia de cortocircuito en el lado secundario de los transformadores z_{cc^2}, por las razones explicadas cuando se expuso el modelo en π.

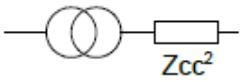


Fig. 31- Impedancia de cortocircuito del lado secundario. Fuente: [7]

La siguiente tabla recoge las impedancias base de cada transformador individual, calculadas como se mencionó anteriormente.

Transf. #	S_n [MWA]	U_{n2} [kV]	Z_B [Ω]
t_{23}	270	132	64,5333
t_{45}	37,5	30	24,0000
t_{67}	10	13,8	19,0440
t_{38}	50	13,8	3,8088

Tabla 10. Impedancias base referidas a los valores nominales de los transformadores.

La resistencias y reactancias de cortocircuito de cada transformador individual, expresada en sus bases nominales, vienen dadas en tanto por ciento en la Tabla 4. Para pasar estos valores a por unidad basta con dividirlos por 100, eso es lo que muestra la quinta y sexta columna de la siguiente tabla.

Transf. #	S_n [MWA]	R_{sc} [%]	X_{sc} [%]	R_{cc} [pu]	X_{cc} [pu]	Z_B [Ω]	Z'_B [Ω]	R'_{cc} [pu]	X'_{cc} [pu]
t_{23}	2 x 270	0,90	12,90	0,0090	0,1290	64,5333	174,2400	3,333E-03	4,7778E-02
t_{45}	3 x 37,5	0,90	9,00	0,0090	0,0900	24,0000	9,0000	2,400E-02	2,4000E-01
t_{67}	10	0,95	4,80	0,0095	0,0480	19,0440	1,9044	9,500E-02	4,8000E-01
t_{38}	3 x 50	0,92	8,50	0,0092	0,0850	3,8088	1,9044	1,840E-02	1,7000E-01

Tabla 11. Cálculo de las impedancias de cortocircuito en por unidad en las bases de la red.

Datos para PowerWorld

Hasta este punto tenemos calculada la impedancia base nominal de los transformadores Z_B y la impedancia de cortocircuito en por unidad en las bases nominales de los transformadores $Z_{cc,pu} = R_{cc,pu} + jX_{cc,pu}$. Además, conocemos las impedancias base

de las diferentes zonas de la red (última columna de la Tabla 7), a las que llamaremos Z'_B. Solo faltan por determinar las impedancias de cortocircuito en p.u. en las bases de la red, a las que denotaremos $Z'_{cc,pu}$ y que se calculan mediante la siguiente expresión:

$$\boxed{Z_{pu} \cdot Z_B = Z'_{pu} \cdot Z'_B} \rightarrow \boxed{Z'_{pu} = Z_{pu} \cdot \frac{Z_B}{Z'_B}}$$

Los resultados de esta operación se recogen en las dos últimas columnas de la Tabla 11 $\rightarrow Z'_{cc,pu} = R'_{cc,pu} + jX'_{cc,pu}$. Estos valores son los que se introducen en PowerWorld para modelar los transformadores regulables, además de las tomas *a* y del desfase propio de cada transformador (estos valores se recogen en la Tabla 4).

En las siguientes imágenes se muestran capturas de PowerWorld en la que se están introduciendo los parámetros calculados para uno de los transformadores de la línea 2.

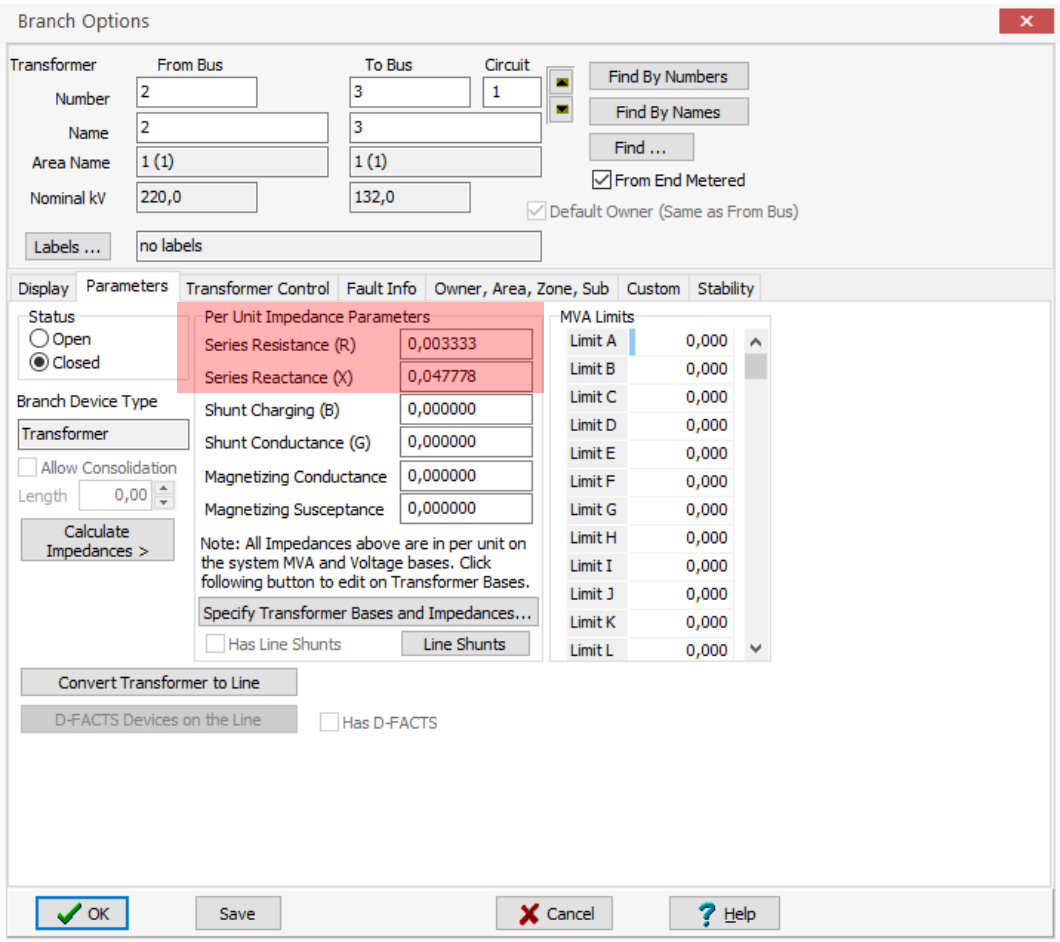


Fig. 32- Introducción de parámetros del transformador en PowerWorld (I).

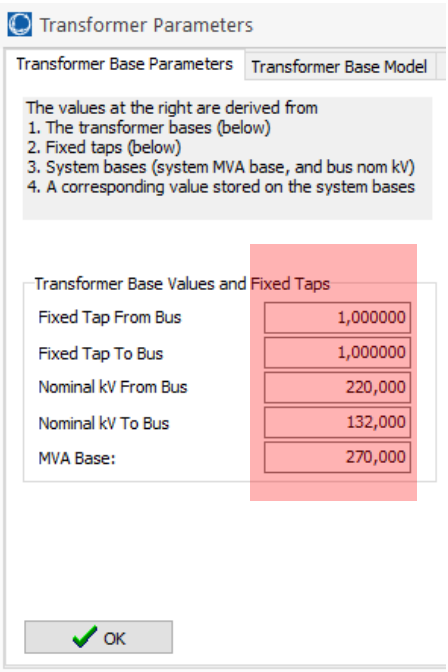


Fig. 33- Introducción de parámetros del transformador en PowerWorld (II).

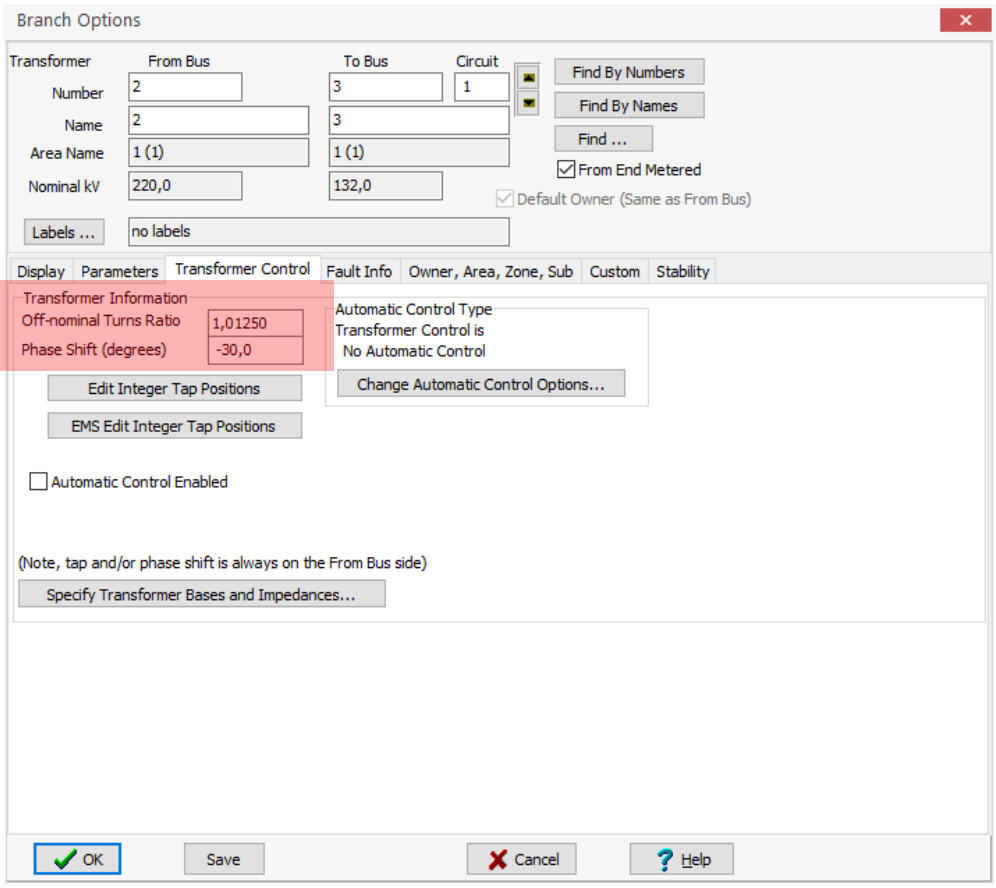


Fig. 34- Introducción de parámetros del transformador en PowerWorld (III).

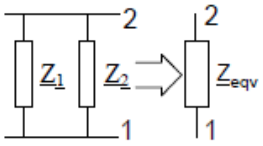
En la siguiente tabla se muestran las resistencias y reactancias de cortocircuito en p.u. teniendo en cuenta las asociaciones en paralelo de las mismas (para Matlab y Python). Esta tabla es idéntica a la Tabla 11 de la quinta columna en adelante. El único cambio está en el cálculo de $R_{cc,pu}$ y $X_{cc,pu}$, donde estos valores se corresponden con las resistencias y reactancias fruto de la asociación en paralelo:

$$R_{eq} = \frac{1}{\frac{1}{R1} + \frac{1}{R2} + \cdots} \quad ; \quad X_{eq} = \frac{1}{\frac{1}{X1} + \frac{1}{X2} + \cdots}$$

$R_{cc\,eq}$ [pu]	$X_{cc\,eq}$ [pu]	Z_B [Ω]	Z'_B [Ω]	$R'_{cc\,eq}$ [pu]	$X'_{cc\,eq}$ [pu]
0,0045	0,0645	64,5333	174,2400	1,667E-03	2,3889E-02
0,0030	0,0300	24,0000	9,0000	8,000E-03	8,0000E-02
0,0095	0,0480	19,0440	1,9044	9,500E-02	4,8000E-01
0,0031	0,0283	3,8088	1,9044	6,133E-03	5,6667E-02

Tabla 12. Cálculo de las impedancias equivalentes de cortocircuito en por unidad en las bases de la red

Datos para Matlab y Python



5.3 Comprobación del modelo de la red de distribución en PowerWorld

Una vez introducidos en PowerWorld los parámetros de todos los elementos que componen la red de distribución, el aspecto es el siguiente:

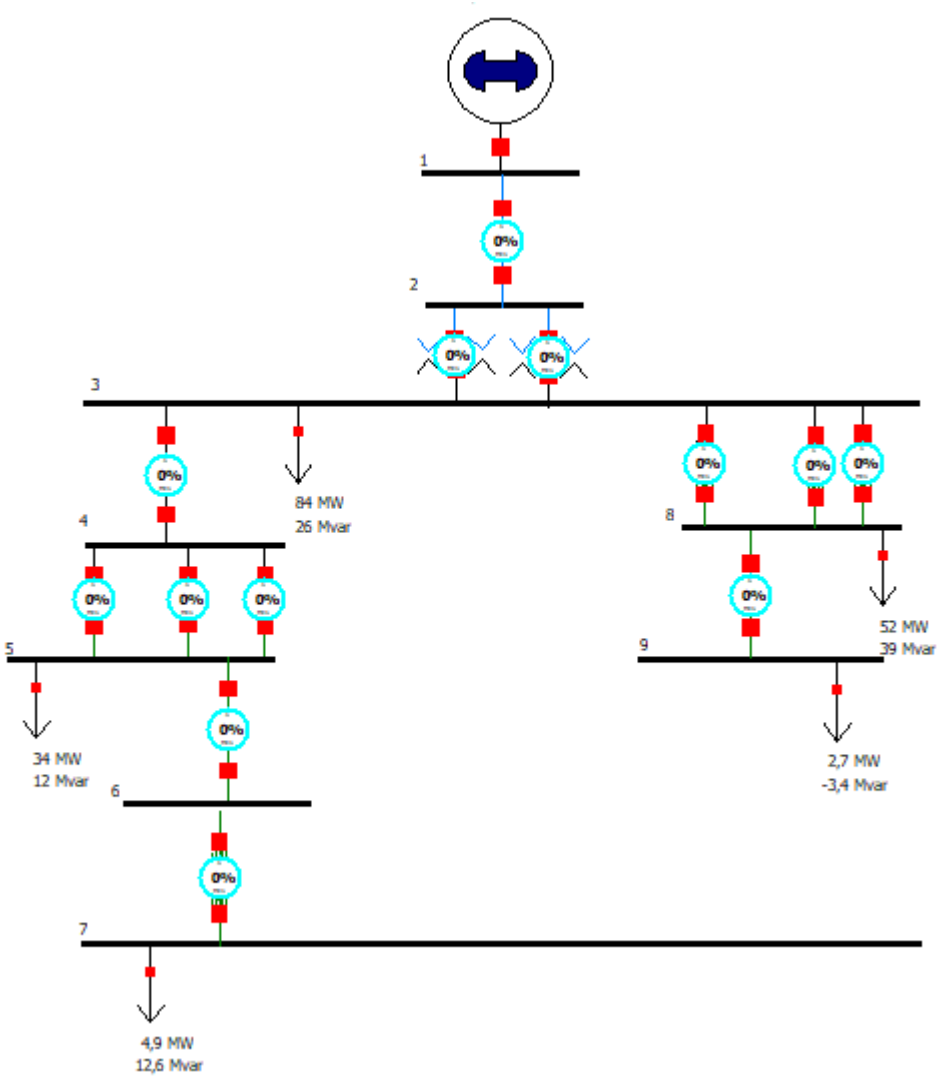
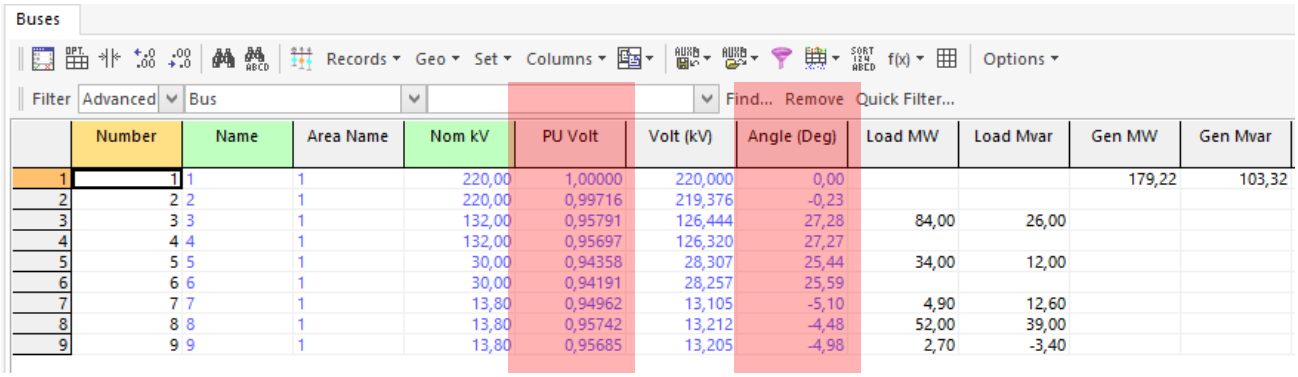


Fig. 35- Modelo de la red de distribución en PowerWorld.

Para comprobar la bondad del modelo y que los datos introducidos son los correctos, se van a comprar los resultados proporcionados por PowerWorld con los resultados reales del flujo de cargas.

Los resultados de PowerWorld son los siguientes:



Buses

Records Geo Set Columns Options

Filter Advanced Bus Find... Remove Quick Filter...

	Number	Name	Area Name	Nom kV	PU Volt	Volt (kV)	Angle (Deg)	Load MW	Load Mvar	Gen MW	Gen Mvar
1	1	1	1	220,00	1,00000	220,000	0,00			179,22	103,32
2	2	2	1	220,00	0,99716	219,376	-0,23				
3	3	3	1	132,00	0,95791	126,444	27,28	84,00	26,00		
4	4	4	1	132,00	0,95697	126,320	27,27				
5	5	5	1	30,00	0,94358	28,307	25,44	34,00	12,00		
6	6	6	1	30,00	0,94191	28,257	25,59				
7	7	7	1	13,80	0,94962	13,105	-5,10	4,90	12,60		
8	8	8	1	13,80	0,95742	13,212	-4,48	52,00	39,00		
9	9	9	1	13,80	0,95685	13,205	-4,98	2,70	-3,40		

Fig. 36- Resultados flujo de cargas PowerWorld.

Resultados reales:

Bus #	$\lvert V_i \rvert$ [pu]	θ_i [deg.]	Bus #	$\lvert V_i \rvert$ [pu]	θ_i [deg.]
2	0.9972	−0.226	6	0.9419	25.588
3	0.9579	27.278	7	0.9496	−5.097
4	0.9570	27.270	8	0.9574	−4.478
5	0.9436	25.436	9	0.9569	−4.980

Fig. 37- Resultado real flujo de cargas PowerWorld. Fuente: [12]

Comparando los módulos y ángulos de las tensiones (V_i, θ_i) en los nodos se puede observar que los resultados coinciden a la perfección. Además, para mayor seguridad dichos resultados se han corroborado también en Matlab. Se concluye entonces que los valores utilizados para construir el modelo de la red de distribución son los adecuados para continuar trabajando sobre esta red.

5.4 Planteamiento del cálculo de flujo de cargas mediante Newton-Raphson

En el apartado 4.2 se citaron los pasos a seguir para resolver el problema de flujo de cargas mediante el método iterativo de Newton-Raphson. Estos pasos son los que se van a seguir en este apartado para dejar planteado el problema de flujo de cargas para la red que se está estudiando, lo cual es la base para implementar el algoritmo de cálculo en Matlab y Python.

1. **Construir el equivalente por fase del sistema con todas las variables y parámetros representados en por unidad en unas bases comunes.**

   El equivalente por fase del sistema se corresponde con la Figura 28.

2. **Construir la matriz de admitancias de barras [Y].**

Como el sistema que estamos estudiando tiene 9 nodos, la matriz [Y] tendrá dimensión 9x9.

$$[Y]=\begin{bmatrix} Y_{11} & Y_{12} & Y_{13} & Y_{14} & Y_{15} & Y_{16} & Y_{17} & Y_{18} & Y_{19} \\ Y_{21} & Y_{22} & Y_{23} & Y_{24} & Y_{25} & Y_{26} & Y_{27} & Y_{28} & Y_{29} \\ Y_{31} & Y_{32} & Y_{33} & Y_{34} & Y_{35} & Y_{36} & Y_{37} & Y_{38} & Y_{39} \\ Y_{41} & Y_{42} & Y_{43} & Y_{44} & Y_{45} & Y_{46} & Y_{47} & Y_{48} & Y_{49} \\ Y_{51} & Y_{52} & Y_{53} & Y_{15} & Y_{55} & Y_{56} & Y_{57} & Y_{58} & Y_{59} \\ Y_{61} & Y_{62} & Y_{63} & Y_{64} & Y_{65} & Y_{66} & Y_{67} & Y_{68} & Y_{69} \\ Y_{71} & Y_{72} & Y_{73} & Y_{74} & Y_{75} & Y_{76} & Y_{77} & Y_{78} & Y_{79} \\ Y_{81} & Y_{82} & Y_{83} & Y_{84} & Y_{85} & Y_{86} & Y_{87} & Y_{88} & Y_{89} \\ Y_{91} & Y_{92} & Y_{93} & Y_{94} & Y_{95} & Y_{96} & Y_{97} & Y_{98} & Y_{99} \end{bmatrix}$$

3. **Clasificar las barras.**

En la siguiente tabla se recoge una clasificación de las 9 barras que intervienen en el sistema. Se incluye el tipo de barra y los valores conocidos y desconocidos de cada una de ellas. Por tratarse de una red de distribución era de esperar que todas las barras fueran de consumo PQ, a excepción de la barra slack.

Valores			
Tipo de barra	**Conocidos**	**Desconocidos**	**Barra**
Slack	$\|V\|,\theta$	P,Q	1
de Carga	P,Q	$\|V\|, \theta$	2
de Carga	P,Q	$\|V\|, \theta$	3
de Carga	P,Q	$\|V\|, \theta$	4
de Carga	P,Q	$\|V\|, \theta$	5
de Carga	P,Q	$\|V\|, \theta$	6
de Carga	P,Q	$\|V\|, \theta$	7
de Carga	P,Q	$\|V\|, \theta$	8
de Carga	P,Q	$\|V\|, \theta$	9

Tabla 13- Clasificación de las barras del sistema.

En la tabla que se muestra debajo se amplía la información de la tabla anterior. En ella se incluyen los valores conocidos de las potencias en las barras (obtenidas de la Tabla 6) y las incógnitas del sistema (|V|, θ).

Barra	**Tipo**	**|V|**	$\boldsymbol{\theta}$	$\boldsymbol{P_G[MW]}$	$\boldsymbol{P_D[MW]}$	$\boldsymbol{Q_G[Mvar]}$	$\boldsymbol{Q_D[Mvar]}$
1	*Slack*	1	0	-	-	-	-
2	de Carga	?	?	0	0	0	0
3	de Carga	?	?	0	84	0	26
4	de Carga	?	?	0	0	0	0
5	de Carga	?	?	0	34	0	12
6	de Carga	?	?	0	0	0	0
7	de Carga	?	?	0	4,9	0	12,6
8	de Carga	?	?	0	52,0	0	39,0
9	de Carga	?	?	0	2,7	0	-3,4

Tabla 14. Incógnitas y Potencias Generadas y Demandadas en cada barra del sistema.

La matriz de incógnitas es, por tanto, la que se muestra a continuación:

$$\begin{bmatrix} \Delta\theta \\ \Delta V/V \end{bmatrix} = \begin{bmatrix} \Delta\theta_2 \\ \Delta\theta_3 \\ \Delta\theta_4 \\ \Delta\theta_5 \\ \Delta\theta_6 \\ \Delta\theta_7 \\ \Delta\theta_8 \\ \Delta\theta_9 \\ \Delta V_2/V_2 \\ \Delta V_3/V_3 \\ \Delta V_4/V_4 \\ \Delta V_5/V_5 \\ \Delta V_6/V_6 \\ \Delta V_7/V_7 \\ \Delta V_8/V_8 \\ \Delta V_9/V_9 \end{bmatrix}$$

Estas incógnitas son las que se van a hallar mediante el estudio de flujo de cargas del sistema. Una vez resuelto el problema y determinados los valores de las incógnitas, a partir de las V_{barras} se podrían calcular todas las magnitudes de interés: I, P, Q, P_{cu}, ΔV %, etc.

4. Plantear las ecuaciones de potencia (P_i , Q_i).

Las ecuaciones de potencia solo se plantean en las barras en las que se conocen las potencias (P ó Q), y nunca en la barra slack, pues es la barra encargada de compensar el sistema. De las barras de carga se conocen tanto la potencia activa como la reactiva (P, Q). De este tipo de barras se obtienen, por tanto, 2 ecuaciones por cada barra.

En nuestro caso, se plantean las ecuaciones de potencia correspondientes a todas las barras exceptuando la barra slack:

$$P_2 , P_3 , P_4 , P_5 , P_6 , P_7 , P_8 , P_9 \,//\, Q_2 , Q_3 , Q_4 , Q_5, Q_6, Q_7, Q_8, Q_9$$

Las potencias activas y reactivas fluyendo desde las barras se calculan de acuerdo a las siguientes expresiones:

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

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

5. Calcular los residuos de potencia [Δ*P* | ΔQ] y los términos del Jacobiano [J].

Los **residuos de potencia** se calculan de la siguiente manera:

$$\Delta P_i = P_i^{espec.} - P_i$$

$$\Delta Q_i = Q_i^{espec.} - Q_i$$

Siendo P_i y Q_i las expresiones planteadas en el apartado anterior.

Las potencias especificadas $P_i^{espec.}$ y $Q_i^{espec.}$ son las potencias exactas inyectadas por cada barra *i* del sistema, y deben ser coincidentes con las que se obtendrán del cálculo P_i y Q_i.

Por lo tanto, se tiene que llegar a que:

$$P_i^{espec.} - P_i = 0$$

$$Q_i^{espec.} - Q_i = 0$$

En nuestro caso, los valores de las potencias especificadas se calculan a partir de los valores recogidos en la Tabla 6 (hay que pasarlos a sistema por unidad).

Por definición, estas potencias se definen como la diferencia entre la potencia generada y la potencia demandada por la barra *i* que se esté estudiando. Así, para nuestro caso llegamos a los siguientes valores:

$$P_3^{espec.} = P_G - P_D = 0 - 0{,}84 = -0.84$$

$$P_5^{espec.} = P_G - P_D = 0 - 0{,}34 = -0{,}34$$

$$P_7{}^{espec.} = P_G - P_D = 0 - 0{,}049 = 0{,}049$$

$$P_8{}^{espec.} = P_G - P_D = 0 - 0{,}52 = -0{,}52$$

$$P_9{}^{espec.} = P_G - P_D = 0 - 0{,}027 = -0{,}027$$

$$Q_3{}^{espec.} = Q_G - Q_D = 0 - 0{,}26 = -0{,}26$$

$$Q_5{}^{espec.} = Q_G - Q_D = 0 - 0{,}12 = -0{,}12$$

$$Q_7{}^{espec.} = Q_G - Q_D = 0 - 0{,}126 = -0{,}126$$

$$Q_8{}^{espec.} = Q_G - Q_D = 0 - 0{,}39 = -0{,}39$$

$$Q_9{}^{espec.} = Q_G - Q_D = 0 - (-0{,}034) = 0{,}034$$

Evaluando las diferencias $(P_i{}^{espec.} - P_i = 0)\ y\ (Q_i^{espec.} - Q_i = 0)$ llegamos a escribir el sistema de ecuaciones que describen nuestro problema de flujo de cargas.

En cuanto al Jacobiano, éste se define de la siguiente manera:

$$[\mathrm{J}] = \begin{bmatrix} H & N \\ M & L \end{bmatrix}$$

Siendo los **términos del Jacobiano** los siguientes:

$$H_{ik} = \partial P_i/\partial\theta_k \qquad N_{ik} = V_k\,\partial P_i/\partial V_k$$

$$M_{ik} = \partial Q_i/\partial\theta_k \qquad L_{ik} = V_k\,\partial Q_i/\partial V_k$$

Donde P_i y Q_i son las ecuaciones de potencia determinadas en el punto 4.

$$\mathrm{J} = \left[\begin{array}{ccc:ccc} H_{22} & \dots & H_{29} & N_{22} & \dots & N_{29} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ H_{92} & \dots & H_{99} & N_{92} & \dots & N_{99} \\ \hdashline M_{22} & \dots & M_{29} & L_{22} & \dots & L_{29} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ M_{92} & \dots & M_{29} & L_{92} & \dots & L_{99} \end{array}\right]$$

De acuerdo con la definición de cada elemento de la matriz podemos escribir el Jacobiano como sigue:

$$J = \left[\begin{array}{ccc|ccc} \frac{\partial P_2}{\partial \theta_2} & \cdots & \frac{\partial P_2}{\partial \theta_9} & V_2\frac{\partial P_2}{\partial V_2} & \cdots & V_9\frac{\partial P_2}{\partial V_9} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial P_9}{\partial \theta_2} & \cdots & \frac{\partial P_9}{\partial \theta_9} & V_2\frac{\partial P_9}{\partial V_2} & \cdots & V_9\frac{\partial P_9}{\partial V_9} \\ \hline \frac{\partial Q_2}{\partial \theta_2} & \cdots & \frac{\partial Q_2}{\partial \theta_9} & V_2\frac{\partial Q_2}{\partial V_2} & \cdots & V_9\frac{\partial Q_2}{\partial V_9} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial Q_9}{\partial \theta_2} & \cdots & \frac{\partial Q_9}{\partial \theta_9} & V_2\frac{\partial Q_9}{\partial V_2} & \cdots & V_9\frac{\partial Q_9}{\partial V_9} \end{array}\right]$$

6. Plantear el sistema de ecuaciones a resolver en forma matricial.

Podemos expresar el sistema de ecuaciones de nuestro problema de flujo de cargas en forma matricial a partir de los cálculos realizados en los apartados anteriores.

De esta manera, se llega a un sistema de ecuaciones expresado en forma matricial que tiene la siguiente forma:

$$\begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix}^{(k)} = \underbrace{\begin{bmatrix} H & N \\ M & L \end{bmatrix}^{(k)}}_{\text{Jacobiano [J]}} \begin{bmatrix} \Delta\theta \\ \Delta V/V \end{bmatrix}^{(k)} \qquad (1)$$

$$\begin{bmatrix} \theta \\ V \end{bmatrix}^{(k+1)} = \begin{bmatrix} \theta \\ V \end{bmatrix}^{(k)} + \begin{bmatrix} \Delta\theta \\ \Delta V \end{bmatrix}^{(k)} \qquad (2)$$

Donde los elementos de la primera y segunda matriz del sistema (1) – matriz de residuos de potencia y Jacobiano – se han determinado en el punto 5 de este apartado del trabajo.

Por su parte, los elementos del vector de incógnitas se plantearon en el punto 3 de este apartado.

Sustituyendo dichos elementos en el sistema matricial de ecuaciones (1) tenemos planteado el sistema de ecuaciones para nuestro problema particular:

$$\begin{bmatrix} \Delta P_2 \\ \Delta P_3 \\ \Delta P_4 \\ \Delta P_5 \\ \Delta P_6 \\ \Delta P_7 \\ \Delta P_8 \\ \Delta P_9 \\ \Delta Q_2 \\ \Delta Q_3 \\ \Delta Q_4 \\ \Delta Q_5 \\ \Delta Q_6 \\ \Delta Q_7 \\ \Delta Q_8 \\ \Delta Q_9 \end{bmatrix} = \begin{bmatrix} \frac{\partial P_2}{\partial \theta_2} & \cdots & \frac{\partial P_2}{\partial \theta_9} & V_2\frac{\partial P_2}{\partial V_2} & \cdots & V_9\frac{\partial P_2}{\partial V_9} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial P_9}{\partial \theta_2} & \cdots & \frac{\partial P_9}{\partial \theta_9} & V_2\frac{\partial P_9}{\partial V_2} & \cdots & V_9\frac{\partial P_9}{\partial V_9} \\ \frac{\partial Q_2}{\partial \theta_2} & \cdots & \frac{\partial Q_2}{\partial \theta_9} & V_2\frac{\partial Q_2}{\partial V_2} & \cdots & V_9\frac{\partial Q_2}{\partial V_9} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial Q_9}{\partial \theta_2} & \cdots & \frac{\partial Q_9}{\partial \theta_9} & V_2\frac{\partial Q_9}{\partial V_2} & \cdots & V_9\frac{\partial Q_9}{\partial V_9} \end{bmatrix} = \begin{bmatrix} \Delta\theta_2 \\ \Delta\theta_3 \\ \Delta\theta_4 \\ \Delta\theta_5 \\ \Delta\theta_6 \\ \Delta\theta_7 \\ \Delta\theta_8 \\ \Delta\theta_9 \\ \Delta V_2/V_2 \\ \Delta V_3/V_3 \\ \Delta V_4/V_4 \\ \Delta V_5/V_5 \\ \Delta V_6/V_6 \\ \Delta V_7/V_7 \\ \Delta V_8/V_8 \\ \Delta V_9/V_9 \end{bmatrix}$$

Residuos de potencia — **Jacobiano** — **Incógnitas**

7. **Efectuar cálculo iterativo para obtener una solución aproximada mediante el método de Newton-Raphson.**

Una vez planteado el sistema de ecuaciones de nuestro problema en forma matricial, el siguiente paso es realizar un proceso iterativo para obtener las soluciones aproximadas de los valores de los ángulos y módulos de la tensión de las barras de nuestro sistema de potencia.

El método de Newton-Raphson es el método de cálculo numérico empleado generalmente para la resolución de los flujos de carga. A continuación, se muestran los pasos a seguir para efectuar el cálculo iterativo.

a. Para la primera iteración inicializar las tensiones no especificadas con un perfil plano (1∠0° pu).

$$[V_1\angle\theta_1, V_2\angle\theta_2, V_3\angle\theta_3, V_4\angle\theta_4, V_5\angle\theta_5, V_6\angle\theta_6, V_7\angle\theta_7, V_8\angle\theta_8, V_9\angle\theta_9]^{(0)}$$

$$[1\angle 0°, 1\angle 0°, 1\angle 0°, 1\angle 0°, 1\angle 0°, 1\angle 0°, 1\angle 0°, 1\angle 0°, 1\angle 0°]$$

b. Calcular las potencias y los residuos de potencia correspondientes a estos valores iniciales.

Esto se corresponde con calcular los elementos de la primera matriz del sistema (1) utilizando como valores iniciales los correspondientes al arranque plano.

$$\begin{bmatrix}\Delta P_2\\ \Delta P_3\\ \Delta P_4\\ \Delta P_5\\ \Delta P_6\\ \Delta P_7\\ \Delta P_8\\ \Delta P_9\\ \Delta Q_2\\ \Delta Q_3\\ \Delta Q_4\\ \Delta Q_5\\ \Delta Q_6\\ \Delta Q_7\\ \Delta Q_8\\ \Delta Q_9\end{bmatrix}^{(0)} = \begin{bmatrix}P_2^{espec.} - P_2^{(0)}\\ P_3^{espec.} - P_3^{(0)}\\ P_4^{espec.} - P_4^{(0)}\\ P_5^{espec.} - P_5^{(0)}\\ P_6^{espec.} - P_6^{(0)}\\ P_7^{espec.} - P_7^{(0)}\\ P_8^{espec.} - P_8^{(0)}\\ P_9^{espec.} - P_9^{(0)}\\ Q_2^{espec.} - Q_2^{(0)}\\ Q_3^{espec.} - Q_3^{(0)}\\ Q_4^{espec.} - Q_4^{(0)}\\ Q_5^{espec.} - Q_5^{(0)}\\ Q_6^{espec.} - Q_6^{(0)}\\ Q_7^{espec.} - Q_7^{(0)}\\ Q_8^{espec.} - Q_8^{(0)}\\ Q_9^{espec.} - Q_9^{(0)}\end{bmatrix}$$

c. Evaluar los elementos del Jacobiano correspondientes a esta iteración.

Ahora se deben calcular los elementos de la segunda matriz del sistema (1) utilizando los valores iniciales del apartado a (arranque plano). Es decir, determinar $[J]^{(0)}$.

d. Resolver el sistema de ecuaciones lineales para obtener $[\Delta\theta \mid \Delta V/V]$.

Una vez calculadas las matrices del apartado c. y d., ya se pueden calcular los elementos de la tercera matriz del sistema (1), resolviendo el sistema de ecuaciones lineales de la siguiente manera:

$$\begin{bmatrix} \Delta\theta \\ \Delta V/V \end{bmatrix}^{(0)} = [\,\mathrm{J}\,]^{(0)^{-1}} \begin{bmatrix} \Delta P \\ \Delta Q \end{bmatrix}^{(0)}$$

e. Utilizar los nuevos valores obtenidos del vector de incógnitas $[\theta \mid V]$ para la siguiente iteración.

Ahora se deben utilizar los valores de partida del apartado a. y los valores obtenidos en el apartado d. para determinar los nuevos valores del vector de incógnitas para comenzar la segunda iteración:

$$\begin{bmatrix} \theta \\ V \end{bmatrix}^{(1)} = \begin{bmatrix} \theta \\ V \end{bmatrix}^{(0)} + \begin{bmatrix} \Delta\theta \\ \Delta V \end{bmatrix}^{(0)}$$

La segunda matriz de este sistema se corresponde con los valores de partida del apartado a.

f. Repetir el procedimiento desde el paso b.

g. Realizar iteraciones hasta que se alcance la convergencia, es decir, que los residuos de potencia sean menores a un error $\varepsilon = 10^{-3}$.

8. Una vez conocidos los fasores de tensión en todas las barras es posible calcular el resto de variables de red.

5.5 Implementación del cálculo de flujo de cargas en Matlab y Python

En el anterior apartado se dejó planteado el problema de flujo de cargas de la red de distribución de 9 nodos. Solo con el planteamiento, sin realizar ninguna iteración, se puede comprobar la gran cantidad de operaciones que conlleva resolver el problema de flujo de cargas por el método de Newton-Raphson.

Para optimizar el proceso de cálculo se ha optado por implementar un algoritmo en Matlab que permita resolver el flujo de cargas de esta red de distribución, para una tolerancia de 10^{-5}. Este algoritmo se recoge en el ANEXO II.

Como se comentó en un inicio, este algoritmo en Matlab está ligeramente modificado para que pueda ser capaz de calcular el flujo de cargas de la red con potencias que van cambiando en el tiempo. Así, los nodos de consumo dejan de tener un valor estático y fijo, que es el caso que se resolvió utilizando Matlab y validado por PowerWorld.

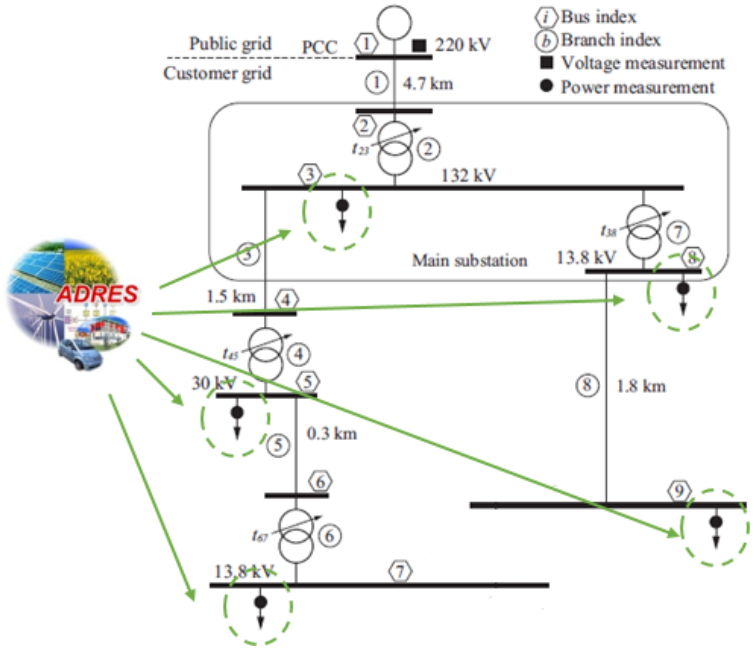


Fig. 38- Esquema lectura de potencias a partir de una base de datos. Fuente: Elaboración Propia

De esta manera, se pretende simular una situación real en la que los consumos o cargas de la red de distribución varían continuamente en el tiempo. Para conseguir esto, se utilizará una base de datos de potencias creada por la Universidad Técnica de Viena en colaboración con otros socios [13]. Esta base de datos, que forma parte del proyecto ADRES-Concept, contiene medidas de potencias reales tomadas en viviendas austriacas. En concreto, se utilizarán 1440 medidas de potencia P y Q correspondientes a 5 puntos diferentes de

medida, que se simularán como los 5 nodos de consumo nuestra red (ver Figura 38). Las 1440 medidas se corresponden con lecturas tomadas cada minuto a lo largo de un día en los cinco puntos de medida. Estas lecturas serán escaladas en el programa para que sean del mismo orden que las medidas sintéticas de partida.

Lo que el algoritmo de Matlab hará es lo siguiente. El programa tiene introducido de antemano los parámetros de la red de distribución que se está estudiando. Los únicos datos que no están introducidos a priori son las potencias o consumos en cada nodo. Una vez iniciado el algoritmo, el programa comenzará a funcionar y cargará el archivo de la base de datos austriaca. Cogerá 5 medidas de P y 5 medidas de Q y se las asignará a los 5 nodos de nuestra red. De esta manera, cada nodo de consumo tendrá asignado una P y una Q, que serán adecuadamente escaladas para que sean del mismo orden que las medidas sintéticas de la red de partida. El programa continuará ejecutándose y para estas condiciones de funcionamiento calculará el flujo de cargas de la red mediante el método de Newton-Raphson, y una vez terminado almacenará los resultados obtenidos (Potencias y tensiones complejas en todos los nodos de la red). Este proceso se repetirá hasta que el programa haya leído las 1440 medidas de la base de datos. Al finalizar, se mostrará una gráfica con el perfil de cargas y el perfil de tensiones correspondientes a estas medidas.

Se incluyen algunas imágenes que muestran el funcionamiento del programa. Durante la ejecución del algoritmo se muestra en la ventana de comandos el número de medida que está tomando de la base de datos. Al finalizar indica el tiempo de ejecución del programa.

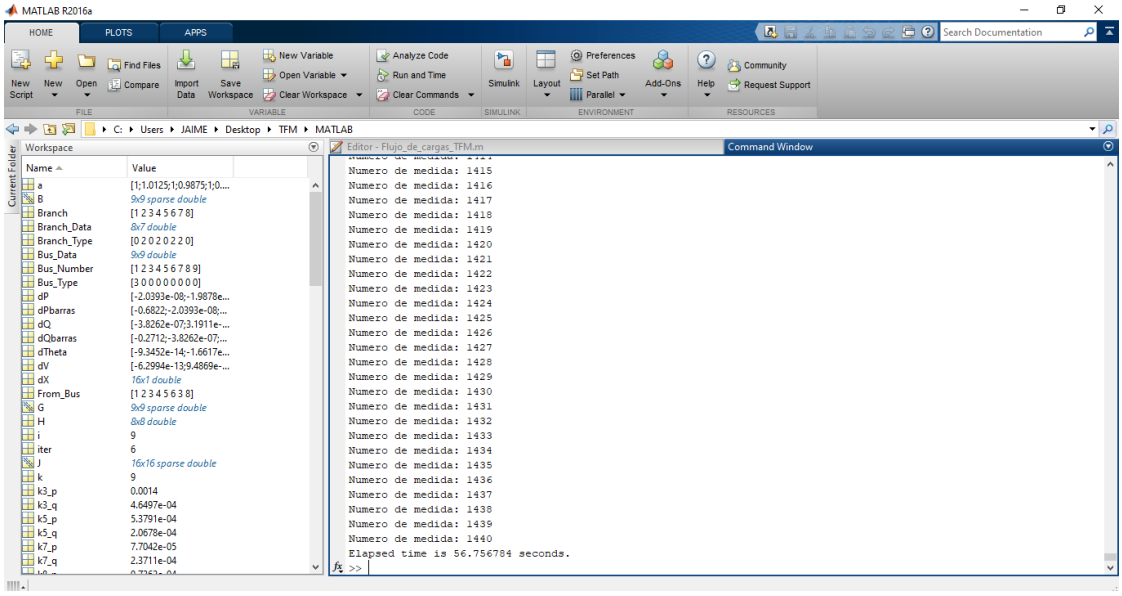


Fig. 39. Funcionamiento del algoritmo de flujo de cargas en Matlab.

El programa también está preparado para mostrar los resultados de flujo de cargas a medida que se van leyendo las potencias. Al ser muchas medidas el programa estaría mostrando resultados sin parar, por eso estas líneas de código están comentadas en el ANEXO II (es decir, no se ejecutan).

Finalmente, se muestran las gráficas que se abren al finalizar el programa. Estas gráficas representan el perfil de cargas y de tensiones diario del nodo 9. Se podrían mostrar los perfiles de cada uno de los cinco nodos de consumo, pero por simplicidad se muestra únicamente el perfil de este nodo. Además, este nodo jugará un papel importante más adelante en el trabajo.

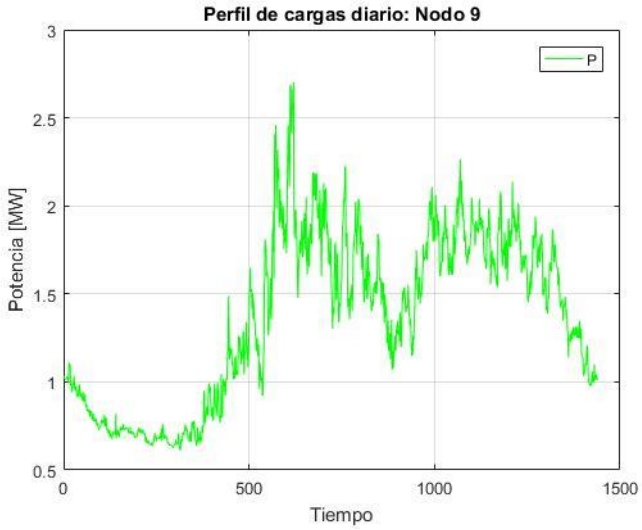


Fig. 40- Perfil diario de potencias activas del nodo 9.

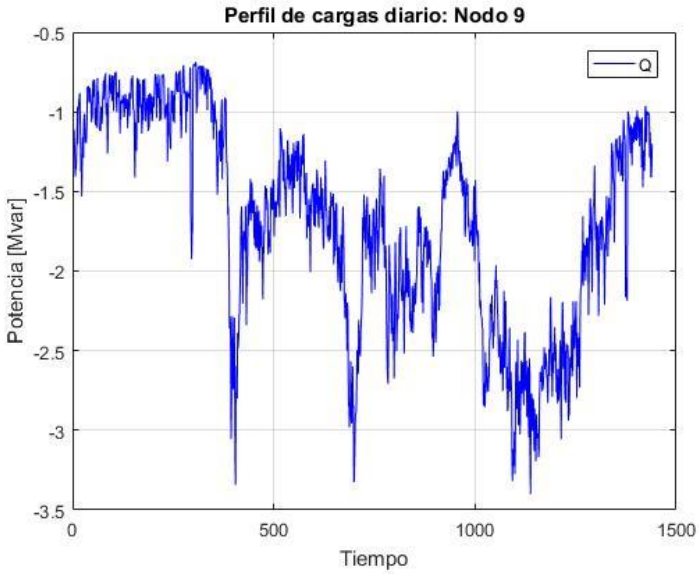


Fig. 41- Perfil diario de potencias reactivas del nodo 9.

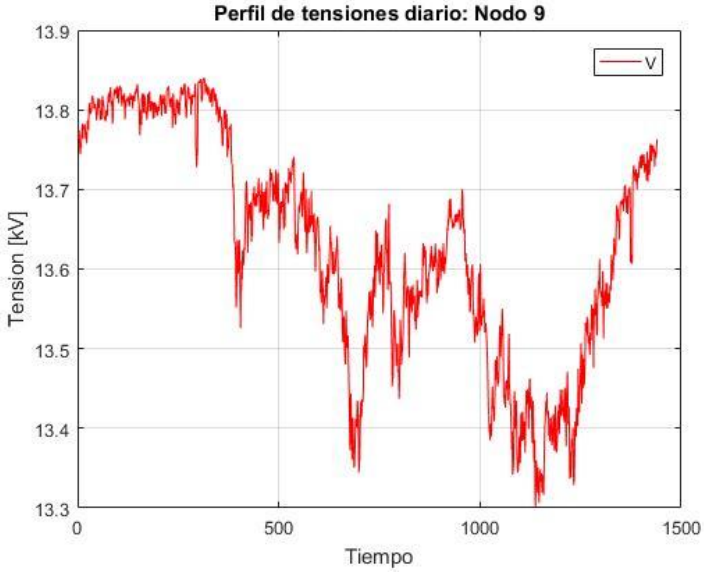


Fig. 42- Perfil de tensiones diario del nodo 9.

5.5 Implementación del cálculo de flujo de cargas en tiempo real en Python 3

El algoritmo de cálculo presentado en el apartado anterior y que se recoge en el ANEXO II se implementó también en lenguaje Python con el objetivo final de ejecutarlo en la Raspberry Pi. Concretamente, se utilizó la versión Python 3.

En el ANEXO III se puede ver el código. Al igual que antes, se han introducido pequeñas modificaciones que van añadiendo complejidad al cálculo de flujo de cargas. Al principio, se utilizó Matlab y PowerWorld para realizar un análisis estático del flujo de cargas de la red de distribución, es decir, las potencias en los nudos de consumo eran constantes y fijas. Más adelante, se diseñó un algoritmo en Matlab que permitía calcular el flujo de cargas cuando las potencias no eran constantes, sino que variaban en el tiempo.

Este programa se implementó también en Python 3. El algoritmo sigue leyendo las potencias del archivo de la base de datos austriaca y asignándolas a los nodos de consumo de nuestra red. Sin embargo, se han realizado ciertos cambios en el nodo 9. A través del uso del paquete Pymodbus en Python se leerán en tiempo real los datos de potencia de un medidor real. Estas potencias deben ser también escaladas para utilizarlas en la red sintética de partida. La siguiente figura da cuenta de esta nueva situación.

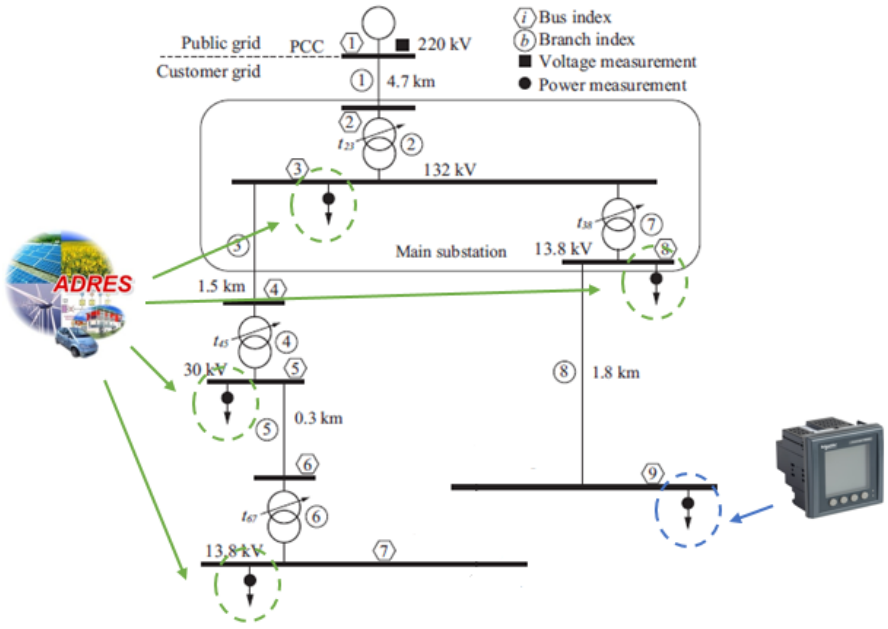


Fig. 43- Nuevo esquema de lectura de potencias de la red de distribución. Fuente: Elaboración Propia

En resumen, se ha pasado de utilizar potencias constantes en los nudos de consumo de la red a utilizar potencias cambiantes en el tiempo. Concretamente, las potencias de los

nudos 3,5,7 y 8 se simulan empleando la base de potencias austriaca, mientras que las potencias en el nodo 9 ahora irán cambiando en tiempo real gracias a las medidas en tiempo real proporcionadas por un medidor real.

5.5.1 Obtención medidas de potencia en tiempo real con PowerMeter

El equipo utilizado para medir potencias y enviar las lecturas en tiempo real es un Power Meter Power Logic PM5560 de Schneider **[14]**. Este equipo está diseñado para medir Energía, Potencia activa y reactiva, Tensión, Corriente, Frecuencia y Factor de potencia.

A través de un protocolo de comunicaciones incorporado en el paquete pymodbus de Python, el algoritmo implementado del ANEXO III se conecta a través de una IP al aparato en cuestión. En función de la conexión a internet el envío de datos entre el Power Meter y el ordenador donde se ejecute el algoritmo será mejor o peor. En el programa de Python está establecido que cada 3 segundos el Power Meter envíe los datos correspondientes al ordenador. Concretamente, los datos recibidos son la fecha, la hora (hasta los milisegundos), la tensión, la intensidad, la potencia activa y la potencia reactiva de una pequeña carga a la que está conectado el equipo.

Fig. 44- Power Meter Power Logic PM5560 de Schneider. Fuente: [14]

5.5.2 Código para el cálculo de flujo de cargas en Python 3

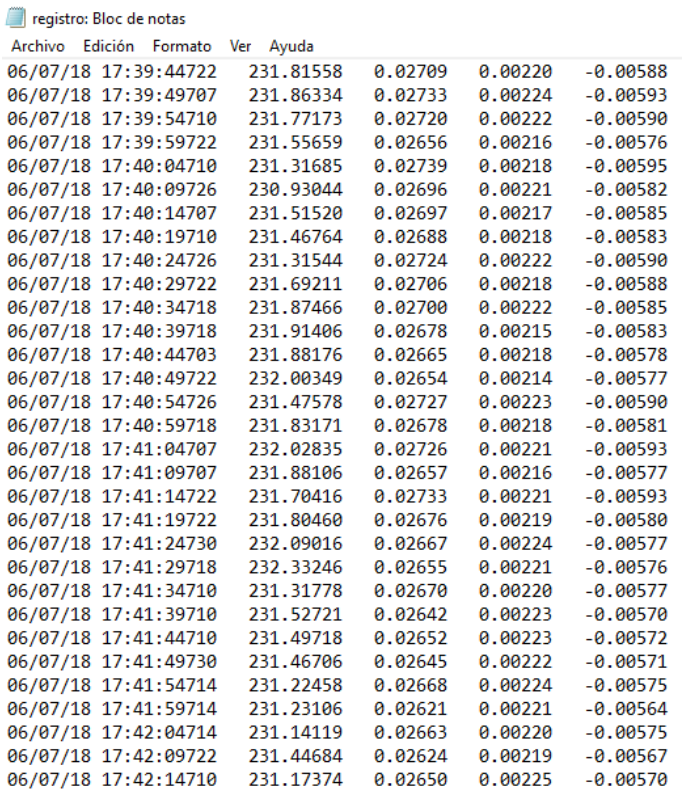
Como ya se ha comentado, el algoritmo implementado en Python 3 se sirve del paquete pymodbus para conectarse al medidor (Power Meter) en tiempo real. El código funciona de la siguiente manera (ANEXO III).

Los parámetros de la red de distribución están introducidos al inicio del programa. Una vez se ejecute dicho programa, el algoritmo asignará a los nodos de consumo unas potencias P y Q, que en el caso de los nodos 3, 5, 7 y 8 serán leídas de la base de datos

austriaca, y en el caso del nodo 9 las potencias serán las enviadas por el Power Meter. Todas las potencias asignadas a los nodos serán escaladas convenientemente para adaptarlas a nuestra red. Con todos estos datos, el programa calculará el flujo de cargas del sistema y almacenará los resultados. Cada 3 segundos se repetirá todo el proceso (lectura y asignación de potencias a los nodos de consumo, cálculo del flujo de cargas, almacenamiento de los resultados).

A medida que se vaya ejecutando el programa, se irán añadiendo en un archivo de texto denominado 'registro.txt' todas las lecturas enviadas por el Power Meter. En este archivo se registran los siguientes campos de izquierda a derecha:

- Fecha (Día/Mes/Año)
- Hora (Hora/Minutos/Milisegundos)
- Tensión [V]
- Intensidad [A]
- Potencia activa [kW]
- Potencia reactiva [kvar]


registro: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
06/07/18 17:39:44722   231.81558   0.02709   0.00220   -0.00588
06/07/18 17:39:49707   231.86334   0.02733   0.00224   -0.00593
06/07/18 17:39:54710   231.77173   0.02720   0.00222   -0.00590
06/07/18 17:39:59722   231.55659   0.02656   0.00216   -0.00576
06/07/18 17:40:04710   231.31685   0.02739   0.00218   -0.00595
06/07/18 17:40:09726   230.93044   0.02696   0.00221   -0.00582
06/07/18 17:40:14707   231.51520   0.02697   0.00217   -0.00585
06/07/18 17:40:19710   231.46764   0.02688   0.00218   -0.00583
06/07/18 17:40:24726   231.31544   0.02724   0.00222   -0.00590
06/07/18 17:40:29722   231.69211   0.02706   0.00218   -0.00588
06/07/18 17:40:34718   231.87466   0.02700   0.00222   -0.00585
06/07/18 17:40:39718   231.91406   0.02678   0.00215   -0.00583
06/07/18 17:40:44703   231.88176   0.02665   0.00218   -0.00578
06/07/18 17:40:49722   232.00349   0.02654   0.00214   -0.00577
06/07/18 17:40:54726   231.47578   0.02727   0.00223   -0.00590
06/07/18 17:40:59718   231.83171   0.02678   0.00218   -0.00581
06/07/18 17:41:04707   232.02835   0.02726   0.00221   -0.00593
06/07/18 17:41:09707   231.88106   0.02657   0.00216   -0.00577
06/07/18 17:41:14722   231.70416   0.02733   0.00221   -0.00593
06/07/18 17:41:19722   231.80460   0.02676   0.00219   -0.00580
06/07/18 17:41:24730   232.09016   0.02667   0.00224   -0.00577
06/07/18 17:41:29718   232.33246   0.02655   0.00221   -0.00576
06/07/18 17:41:34710   231.31778   0.02670   0.00220   -0.00577
06/07/18 17:41:39710   231.52721   0.02642   0.00223   -0.00570
06/07/18 17:41:44710   231.49718   0.02652   0.00223   -0.00572
06/07/18 17:41:49730   231.46706   0.02645   0.00222   -0.00571
06/07/18 17:41:54714   231.22458   0.02668   0.00224   -0.00575
06/07/18 17:41:59714   231.23106   0.02621   0.00221   -0.00564
06/07/18 17:42:04714   231.14119   0.02663   0.00220   -0.00575
06/07/18 17:42:09722   231.44684   0.02624   0.00219   -0.00567
06/07/18 17:42:14710   231.17374   0.02650   0.00225   -0.00570
```

Fig. 45- Archivo de texto creado para registrar las lecturas del medidor.

El programa está preparado para leer 1440 medidas de potencia y, por tanto, realizar 1440 veces el flujo de cargas de la red. Sin embargo, en el anexo se ha escrito que realice 60 medidas. Al finalizar el programa, aparece una ventana como la que se muestra a continuación, que representa gráficamente el perfil de cargas y tensiones diario del nodo 9. Se podría representar el perfil de cualquier nodo, pero resulta más interesante ver la evolución en el nodo que toma las medidas en tiempo real.

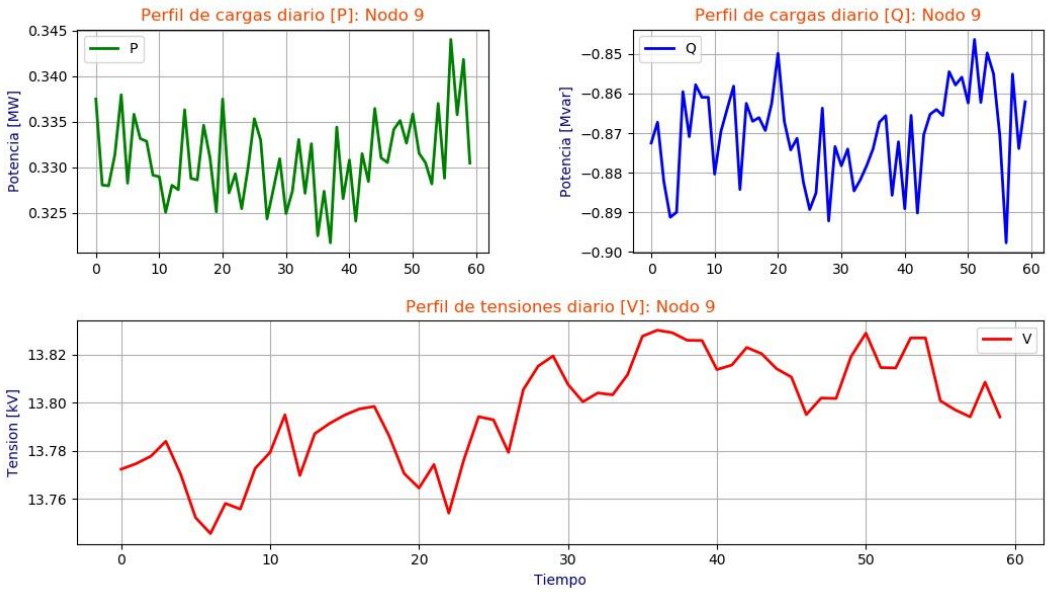


Fig. 46- Perfil de cargas y tensiones diario del nodo 9.

Durante la ejecución del programa se muestra en la consola de comandos el número de medida leída del Power Meter, así como la tensión, potencia activa y reactiva en tiempo real de la carga a la que está conectada el medidor. La consola también muestra al finalizar el programa el tiempo de ejecución del mismo.

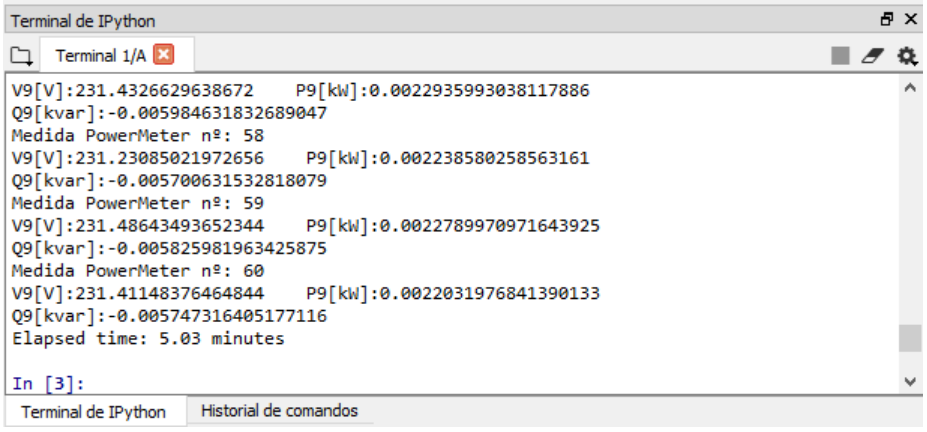


Fig. 47- Medidas en tiempo real a través del Power Meter.

5.6 Aplicación cálculo flujo de cargas en tiempo real con Raspberry Pi 3 Model B+

En este último capítulo se van a presentar los resultados de resolver el flujo de cargas en tiempo real utilizando un pequeño ordenador llamado Raspberry Pi al que se le ha implementado el algoritmo de cálculo del ANEXO III y IV.

Se modificará ligeramente el código del ANEXO III para mostrar en tiempo real la evolución del perfil de cargas y tensiones, y no al final del programa como se hacía hasta ahora. Esto permitirá realizar un análisis más visual y dinámico de los resultados en tiempo real. Se introducirá para ello también el uso de un pequeño control remoto desde el que se puede encender o apagar la carga a la que está conectada el Power Meter para observar en tiempo real cómo se modifican las gráficas de las potencias y las tensiones de la carga, que simulamos como el nodo 9 de nuestra red.

5.6.1 Presentación de la Raspberry Pi

Raspberry Pi es un pequeño ordenador que nació con el propósito de facilitar la enseñanza de los fundamentos básicos de programación en las escuelas. Estos dispositivos están desarrollados por la fundación Raspberry Pi [15].

Para utilizar la Raspberry Pi es necesario conectarla a un monitor a través de un cable HDMI, y a un teclado y ratón a través de los puertos USB de los que dispone. Su aspecto es el siguiente:

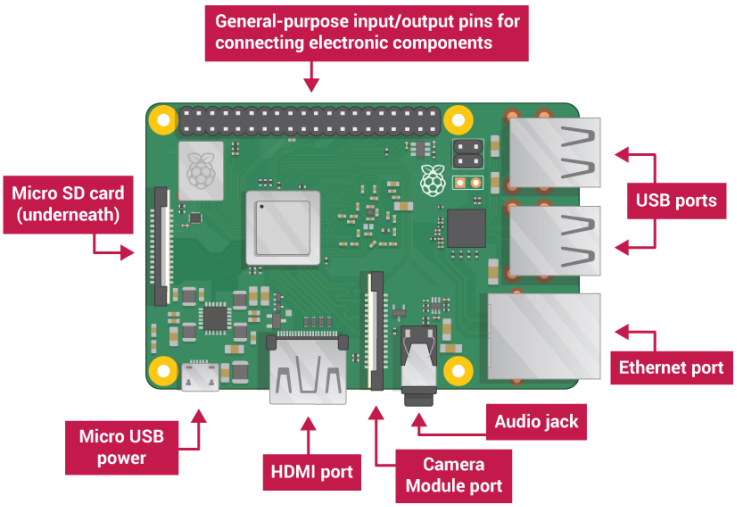


Fig. 48- Aspecto de la Raspberry Pi. Fuente: [15]

- Puertos USB — se utilizan para conectar el ratón y el teclado. También se pueden conectar otros dispositivos, como un lápiz USB.
- Tarjeta de memoria microSD — en ella se almacena el sistema operativo y todos los archivos.
- Puerto Ethernet — utilizado para conectar la Raspberry Pi a internet por cable. La Raspberry Pi también se puede conectar a internet a través de conexión inalámbrica LAN.
- Salida de audio — se pueden conectar auriculares o altavoces
- Puerto HDMI— aquí se conecta el monitor (o proyector) para ver la salida de imagen de la Raspberry Pi. Si el monitor tiene audio se puede escuchar desde aquí.
- Enchufe Micro USB— es donde se conecta una fuente de alimentación. Esta debe ser siempre la última parte de todas, después de haber conectado el resto de los componentes.
- Puertos GPIO — permiten conectar components electrónicos como LEDs a la Raspberry Pi.

El software de la Raspberry Pi es abierto, siendo su sistema operativo oficial una versión adaptada de Debian, denominada Raspbian, aunque permite usar otros sistemas operativos. Raspbian es una distribución del sistema operativo GNU/Linux y por lo tanto libre basado en Debian Strech.

A continuación, se adjuntan un par de imágenes reales de la Raspberry Pi 3 Model B+ con y sin la carcasa de protección.

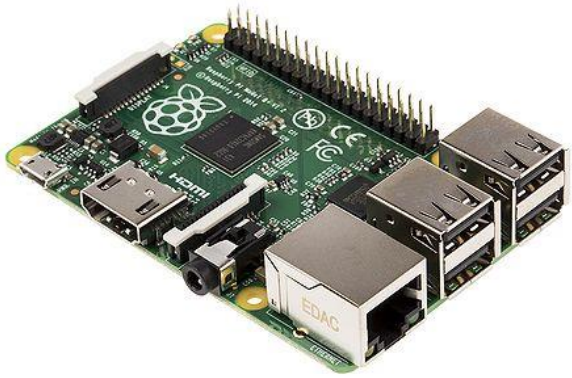


Fig. 49- Raspberry Pi 3 Model B+. Fuente: Elaboración Propia

Fig. 50- Raspberry Pi 3 Model B+ con carcasa. Fuente: Elaboración Propia

Para la puesta en marcha de la Raspberry y la instalación de los paquetes necesarios para hacer funcionar el algoritmo en Python se ha tenido en cuenta **[15]** y **[16]**.

5.6.2 Resultados obtenidos

La Raspberry no deja de ser un ordenador y, por tanto, se le pueden implementar los códigos de Python que se incluyen en los anexos. De hecho, está creada con el objetivo de aprender a programar y por ello vienen incorporados paquetes de programación en el propio dispositivo. Nosotros utilizaremos Python 3.

Los resultados que se incluyen en este apartado se corresponden con el algoritmo del ANEXO IV. Este algoritmo proporciona en tiempo real una gráfica que permite ver la evolución del perfil de cargas y tensiones del nodo 9. Concretamente, la gráfica se actualiza cada 3 segundos, aunque este tiempo puede ser modificado en función de la conexión a internet y la velocidad de envío de datos del Power Meter. Al igual que ocurría con el código del ANEXO III, este programa muestra en la consola de comandos en tiempo real las medidas proporcionadas por el Power Meter (ver Figura 46).

En la Figura 51 se muestran los resultados que se fueron obteniendo en tiempo real, donde los picos y valles más pronunciados de las gráficas se corresponden con la conexión o desconexión de la carga conectada al Power Meter a través de un interruptor remoto.

El interruptor remoto es un aspecto novedoso que se introduce aquí pero que es perfectamente válido para aplicar también al código del ANEXO III. Sin embargo, resulta más útil poder comprobar en directo la evolución de la carga.

El control remoto no es más que un interruptor wifi que permite encender y apagar la carga, y así cambiar la P y la Q del nodo del Power Meter. Este interruptor se controla a través de la aplicación móvil "ewelink", disponible tanto para Android como para iOS.

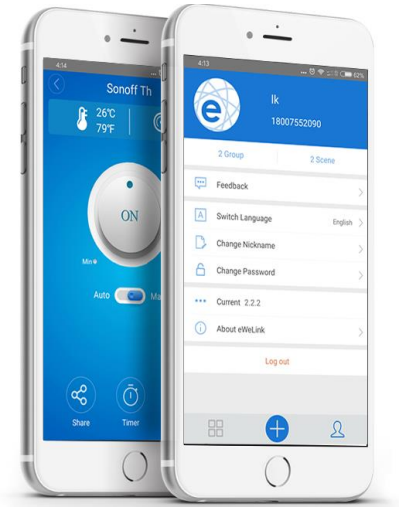


Fig. 51- Aspecto del interruptor wifi "ewelink".

Los siguientes resultados se corresponden al algoritmo del ANEXO IV, detenido tras leer aproximadamente 60 medidas.

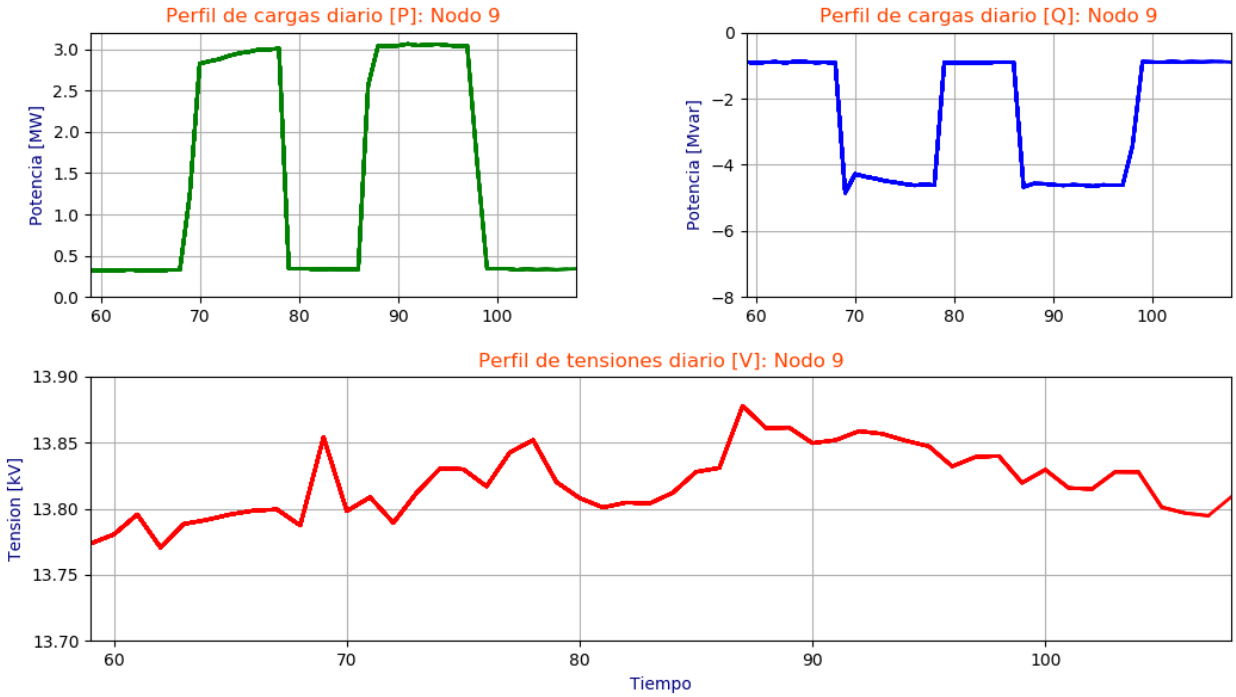


Fig. 52- Resultados algoritmo ANEXO IV utilizando interruptor wifi.

En la siguiente figura se resaltan los cambios de potencia en el nodo 9 introducidos al conectar y desconectar la carga a través del interruptor wifi.

En color negro se representan los momentos en los que se conectó la carga, mientras que en color azul se ilustran las desconexiones de la carga. Se observa cómo al conectar la carga aparecen subidas simultáneas de potencia P y V. La Q también se demanda más (es más negativa) cuando se conecta la carga. Los procesos contrarios ocurren cuando se desconecta la carga.

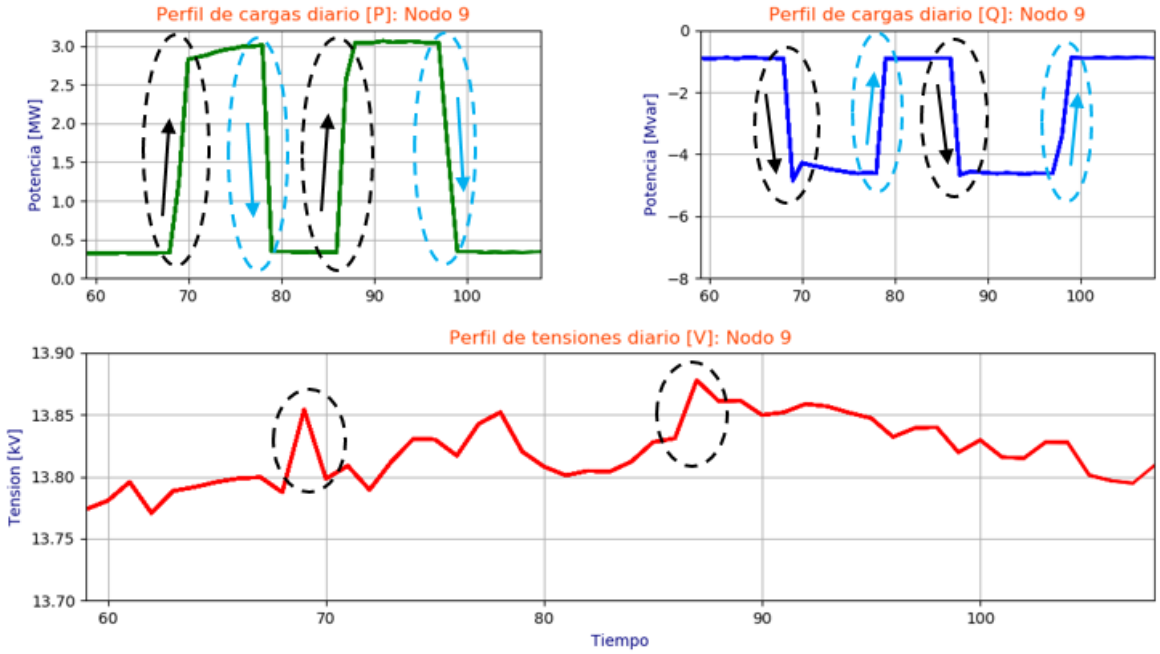


Fig. 53- Perfil de cargas y tensiones al utilizar el interruptor wifi.

6. Conclusiones

Tras la elaboración de este trabajo y los resultados obtenidos se concluye diciendo lo siguiente.

Se ha realizado un modelo eléctrico válido de la red de distribución estudiada, lo cual se ha constatado con el programa PowerWorld. Este modelo ha servido de base para continuar analizando la red.

Se ha implementado tanto en Matlab como en Python 3 un algoritmo para el cálculo de flujo de cargas mediante el método de Newton-Raphson de la red de distribución estudiada.

Se han ido introduciendo mejoras al algoritmo de cálculo hasta el punto de ser aplicable para el cálculo de flujo de cargas en tiempo real. El algoritmo es capaz de asignar a los nodos de consumo de la red de distribución medidas de potencia obtenidas de una base de datos externa, así como de lecturas obtenidas a partir de dispositivos reales.

Se ha desarrollado un algoritmo capaz de interactuar con elementos externos al propio ordenador de trabajo. Los programas que se incluyen en este trabajo se nutren de información del exterior para realizar el cálculo de flujo de cargas en tiempo real. Concretamente, las lecturas de potencia de un nodo particular se obtienen de una carga conectada a un dispositivo real como es el Power Meter Power Logic PM5560 de Schneider, y sus medidas pueden ser modificadas mediante una aplicación móvil que funciona como un interruptor remoto.

Se ha probado el correcto funcionamiento del algoritmo desarrollado en la Raspberry Pi. La ligereza y manejabilidad de este equipo hace factible poder calcular el flujo de cargas de la red de distribución desde cualquier parte con acceso a internet, con las grandes aplicaciones que eso conlleva.

La principal dificultad de este trabajo han sido las labores de programación. Desarrollar el algoritmo de Newton-Raphson en Matlab y después implementarlo en otro lenguaje de programación (Python 3) ha sido laborioso, pues ha requerido familiarizarse con ambos entornos de trabajo, incluso con diferentes sistemas operativos (Windows, Raspbian).

Concluimos diciendo que los algoritmos de cálculo presentados en este trabajo tienen aplicación en el cálculo en tiempo real del flujo de cargas en redes de distribución.

7. Bibliografía

Libros

[4] Gómez Expósito, A. (2002). *Análisis y operación de sistemas de energía eléctrica*. Madrid: McGraw Hill Interamericana.

[5] Grainger, J. and Stevenson, W. (2004). *Análisis de sistemas de potencia*. México: MacGraw-Hill.

[6] Gómez Expósito, A. (2003). *Sistemas eléctricos de potencia*. Madrid: Prentice Hall.

Publicaciones

[10] Barboza, L., Zurn, H. and Salgado, R. (2001). Load tap change transformers: a modeling reminder. *IEEE Power Engineering Review*, 21(2), pp.51-52.

[12] Cano, J., Mojumdar, M., G. Norniella, J. and A. Orcajo, G. (2017). Phase shifting transformer model for direct approach power flow studies. *International Journal of Electrical Power & Energy Systems*, 91, pp.71-79.

[17] Group, W. (1973). Common Format For Exchange of Solved Load Flow Data. *IEEE Transactions on Power Apparatus and Systems*, PAS-92(6), pp.1916-1925.

Recursos de Internet

[1] http://www.ree.es/sites/default/files/downloadable/el_suministro_de_la_electricidad.pdf

[2] http://www.ree.es/es/publicaciones/educacion/de-la-generacion-al-consumo

[3] http://www.ree.es/es/actividades/gestor-de-la-red-y-transportista

[9] http://www.ebah.com.br/content/ABAAAfmXkAL/trabalho-transformadores

[11] http://www.ree.es/es/actividades/demanda-y-produccion-en-tiempo-real

[13] https://www.ea.tuwien.ac.at/projects/adres_concept/EN/

[14] https://www.schneider-electric.es/es/product/METSEPM5560/pm5560-analizador-con-1mod2eth---hasta-63th-h---1%2C1m-4di-2do-52-alarmas---panel

[15] https://www.raspberrypi.org/

[16] https://pythonprogramming.net/introduction-raspberry-pi-tutorials/

Otros Recursos

[7] Apuntes asignatura *Sistemas Eléctricos de Control*. Máster en Ingeniería de Minas. Universidad de Oviedo.

[8] Apuntes asignatura *Electrotecnia*. Grado en Ingeniería de Tecnologías Mineras. Universidad de Oviedo.

8. ANEXOS

ANEXO I – Obtención de las ecuaciones de flujo de cargas.

ANEXO II – Implementación en Matlab del método de Newton-Raphson.

ANEXO III – Implementación en Python 3 del método de Newton-Raphson.

ANEXO IV - Implementación en Python 3 del cálculo de flujo de cargas en tiempo real.

ANEXO I – Obtención de las ecuaciones de flujo de cargas

Se parte de la siguiente expresión,

$$P_i + jQ_i = V_i \cdot I_i^* = V_i \cdot \left[\sum_{k=1}^{N} Y_{ik} \cdot V_k\right]^* \qquad K = 1, 2, \ldots, N$$

La cual se puede reescribir de la siguiente manera,

$$P_i + jQ_i = V_i \cdot I_i^* = \sum_{k=1}^{N} Y_{ik}^* \cdot V_k^*$$

Se introduce V_k dentro del sumatorio,

$$P_i + jQ_i = V_i \cdot I_i^* = \sum_{k=1}^{N} V_i \cdot Y_{ik}^* \cdot V_k^*$$

Se expresan las tensiones y las admitancias en forma rectangular,

$$V_i = V_i\,(\cos\theta_i + j\sin\theta_i)$$

$$V_k^* = V_k\,(\cos\theta_k - j\sin\theta_k)$$

$$Y_{ik}^* = Y_{ik}\,(\cos\theta_k - j\sin\theta_k) = G_{ik} - jB_{ik}$$

Lo que da lugar a la siguiente expresión,

$$P_i + jQ_i = \sum_{k=1}^{N} V_i \cdot V_k\,(\cos\theta_i + j\sin\theta_i)\,(\cos\theta_k - j\sin\theta_k)\;(G_{ik} - jB_{ik}) \qquad (1)$$

$$\cos\theta_i \cdot \cos\theta_k + \sin\theta_i \cdot \sin\theta_k + j\,(\sin\theta_i \cdot \cos\theta_k - \cos\theta_i \cdot \sin\theta_k)$$

$$\cos(\theta_i - \theta_k) \qquad \sin(\theta_i - \theta_k)$$

Se sabe por las identidades trigonométricas que,

$$\cos(\theta_i - \theta_k) = \cos\theta_i \cdot \cos\theta_k + \sin\theta_i \cdot \sin\theta_k$$

$$\sin(\theta_i - \theta_k) = \sin\theta_i \cdot \cos\theta_k - \cos\theta_i \cdot \sin\theta_k$$

Utilizando la siguiente notación → $\theta_{ik} = \theta_i - \theta_k$

$$\cos(\theta_i - \theta_k) = \cos(\theta_{ik})$$

$$\sin(\theta_i - \theta_k) = \sin(\theta_{ik})$$

Se puede reescribir la ecuación (1),

$$P_i + jQ_i = \sum_{k=1}^{N} V_i \cdot V_k \, (\cos\theta_{ik} + j \sin\theta_{ik}) \, (G_{ik} - jB_{ik})$$

Se saca fuera del sumatorio las tensiones V_i,

$$P_i + jQ_i = V_i \sum_{k=1}^{N} V_k \cdot (\cos\theta_{ik} + j \sin\theta_{ik}) \, (G_{ik} - jB_{ik}) \qquad (2)$$

Finalmente, se multiplican los términos que están dentro del sumatorio,

$$(\cos\theta_{ik} + j \sin\theta_{ik}) \, (G_{ik} - jB_{ik}) = \; G_{ik} \cdot \cos\theta_{ik} + B_{ik} \cdot \sin\theta_{ik} + j(\, G_{ik} \cdot \sin\theta_{ik} - B_{ik} \cdot \cos\theta_{ik})$$

Sustituyendo este resultado en (2),

$$P_i + jQ_i = V_i \sum_{k=1}^{N} V_k \cdot G_{ik} \cdot \cos\theta_{ik} + B_{ik} \cdot \sin\theta_{ik} + j(\, G_{ik} \cdot \sin\theta_{ik} - B_{ik} \cdot \cos\theta_{ik})$$

Separando términos reales e imaginarios se llega a las ecuaciones de potencia activa y reactiva del flujo de cargas,

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

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

k =1, 2, …, N

```
%--------------------------------------------------------------------------
%                            FLUJO DE CARGAS
%--------------------------------------------------------------------------
%                           Objetivo del estudio:
% Obtener las condiciones de funcionamiento (tensiones de barra, corrientes
% circulantes, caídas de tensión, pérdidas, etc.) de un Sistema Eléctrico
% de potencia (SEP) en régimen permanente (RP)
% En nuestro caso se van a determinar las tensiones complejas (modulo y angulo)
% y potencias (P,Q) en cada barra del sistema o red
%--------------------------------------------------------------------------
%           Pasos a seguir para el planteamiento del problema:
% 1- Construir el equivalente por fase del sistema con todas las
%    variables y parámetros representados en por unidad en unas bases
%    comunes (PowerWorld)
% 2- Construir la Matriz de Admitancias de barra [Ybarra]
% 3- Clasificar las barras
% 4- Plantear las ecuaciones de potencia
% 5- Resolución de los flujos de carga (Método de Newton-Raphson)
%--------------------------------------------------------------------------
%           Algoritmo de cálculo (Método de Newton-Raphson):
% 1- Inicializar las tensiones con un perfil plano (1<0° pu en todas
%    las barras)
% 2- Calcular R=[dP | dQ]. Si todos los elementos de este vector son
%    menores que la tolerancia detener el proceso. En caso contrario,
%    calcular el Jacobiano [J] y continuar
% 3- Obtener dX=[dTheta | dV/V] resolviendo el sistema: [R]=[J]*[dX]
% 4- Actualizar X=[Theta | V] y volver al paso 2. [X]nueva=[X]antigua+[dX]
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
tic
% Escribiendo "tic" al comienzo del script y "toc" al final permite conocer
% el tiempo que tarda MATLAB en ejecutar el script
%--------------------------------------------------------------------------
% -------------------------------------------------------------------------
%                            BRANCH DATA (CDF [17])
% -------------------------------------------------------------------------
% -------------------------------------------------------------------------
% Branch Type       0 - Transmission line
%                   1 - Fixed tap
%                   2 - Variable tap for voltage control (TCUL, LTC)
%                   3 - Variable tap (turns ratio) for MVAR control
%                   4 - Variable phase angle for MW control (phase shifter)
% -------------------------------------------------------------------------
% Se introducen los datos de todas las lineas de la red
%
% Numero de linea o "branch"
Branch=[1,2,3,4,5,6,7,8];
% Desde la barra "i"
From_Bus=[1,2,3,4,5,6,3,8];
% Hasta la barra "k"
To_Bus=[2,3,4,5,6,7,8,9];
% Tipo de linea de acuerdo con el formado CDF
Branch_Type=[0,2,0,2,0,2,2,0];
% Resistencia [R] de la linea en por unidad
R_pu=[0.000243,0.001667,0.001386,0.008000,0.018933,0.095000,0.006133,0.152174];
% Reactancia [X] de la linea en por unidad
X_pu=[0.002331,0.023889,0.001300,0.080000,0.004433,0.480000,0.056667,0.105860];
```

```
% Tap de los transformadores
Tap=[1,1.0125,1,0.9875,1,0.9250,0.9750,1];  % Tap=1 para líneas
% ---------------------------------------------------------------------------
% Branch | From Bus | To Bus | Branch Type | R[pu] | X[pu] |
Branch_Data=[Branch',From_Bus',To_Bus',Branch_Type',R_pu',X_pu',Tap'];
% ---------------------------------------------------------------------------
% ---------------------------------------------------------------------------
%                       MATRIZ DE ADMITANCIAS [Ybarra]
% ---------------------------------------------------------------------------
% Se renombran los vectores por comodidad de escritura y se convierten
% en vectores columna
R=R_pu';
X=X_pu';
a=Tap';
% Impedancias de las líneas [z]
z=R+X*i;
% Admitancias de las líneas [y]
y=1./z;
% Número de líneas
nl=length(From_Bus);
% Número de barras
nb=max(max(From_Bus),max(To_Bus));
% ---------------------------------------------------------------------------
% Construcción Matriz Admitancias [Y]
Ybarra=zeros(nb,nb);
% Elementos fuera de la diagonal
for i=1:nl
    Ybarra(From_Bus(i),To_Bus(i))= Ybarra(From_Bus(i),To_Bus(i))-y(i)/a(i);
    Ybarra(To_Bus(i),From_Bus(i))=Ybarra(From_Bus(i),To_Bus(i));
end
% Elementos de la diagonal
for i=1:nb
    for k=1:nl
        if From_Bus(k)==i
            Ybarra(i,i)=Ybarra(i,i)+(y(k)/a(k))+[y(k)*(1-a(k))/(a(k)^2)];
        elseif To_Bus(k)==i
            Ybarra(i,i)=Ybarra(i,i)+(y(k)/a(k))+[y(k)*(a(k)-1)/a(k)];
        end
    end
end
% Matriz de Admitancias
Ybarra;
% Como muchos elementos de la matriz son 0, se crea una matriz dispersa
Ybarra=sparse(Ybarra);
%----------------------------------------------------------------------------
% Parte real [G:conductancia] e imaginaria [B:susceptancia] de las
% Admitancias de las barras [Y=G+jB]
G=real(Ybarra);     % Conductancia [G]
B=imag(Ybarra);     % Susceptancia [B]
%----------------------------------------------------------------------------
% Creamos vectores fila de ceros de dimension (1x1440) para ir recogiendo
% los resultados de potencia (P y Q) y de tension (V) del nodo 9. Cada vez
% que se resuelva el flujo de cargas, los resultados de P,Q,V del nodo 9 se
% iran almacenando en estos vectores columna, sustituyendo asi los valores
% originiales de estos vectores, los cuales eran todos cero.
%
% Estos vectores nos serviran para dibujar el perfil diario de cargas y
% tensiones al final del programa.
P_plot=zeros(1,1440);
Q_plot=zeros(1,1440);
```

```
V_plot=zeros(1,1440);
%-------------------------------------------------------------------------
for x=1:1440
    % Mostramos en pantalla el numero de medida 'x'
    disp(['Numero de medida: ',num2str(x)])
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%                   LECTURA DE POTENCIAS Y ESCALADO
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
    % Cargamos en MATLAB el archivo .mat que contiene las lecturas de P y Q
    % de 5 nodos de otra red diferente.
    %
    % Concretamente, el archivo contiene lecturas tomadas cada minuto a lo
    % largo de un dia. En total, 1440 medidas de P y Q de cada nodo.
    %
    % El objetivo es simular que estos 5 nodos se corresponden con los 5
    % nodos con carga de nuestra red. De esta forma, las potencias demandadas
    % en cada nodo ira cambiando a lo largo del dia, lo que se ajusta a la
    % situacion real.
    load ('ADRES_2sets_1min')
    % En el archivo, las lecturas de cada nodo "i" se llaman P_i_o_av y Q_i_o_av
    %
    % Unidades de las lecturas (archivo.mat) -> P [MW] y Q[Mvar]
    %
    % Renombramos las potencias para nuestra red:
    % De esta manera, los 5 nodos del archivo 'ADRES_2Sets_1min.mat' serían
    % en nuestra red los nodos 3,5,7,8,9.
    P3=P_1_o_av;
    P5=P_2_o_av;
    P7=P_3_o_av;
    P8=P_4_o_av;
    P9=P_5_o_av;
    Q3=Q_1_o_av;
    Q5=Q_2_o_av;
    Q7=Q_3_o_av;
    Q8=Q_4_o_av;
    Q9=Q_5_o_av;
    % Datos de la red -> P[MW] y Q[Mvar]
    P3_dato=84;
    P5_dato=34;
    P7_dato=4.9;
    P8_dato=52;
    P9_dato=2.7;
    Q3_dato=26;
    Q5_dato=12;
    Q7_dato=12.6;
    Q8_dato=39;
    Q9_dato=-3.4;
    % Para asignar las lecturas de potencia del archivo.mat a los nodos de
    % nuestra red es necesario realizar un escalado de los datos mediante la
    % siguiente operacion:
    %    (Lectura de potencia * k = Potencia "escalada" en nuestro nodo)
    %
    % k es la constante de escalado -> ki=Pi_dato/max(Pi) ; ki=Qi_dato/max(Qi)
    %
    % Ejemplo para nodo 3 (potencia activa) -> k3_p=84[MW]/max(P9)[MW]
    % Ejemplo para nodo 3 (potencia reactiva) -> k3_q=26[Mvar]/max(P9)[Mvar]
    %
    % Es decir, se calcula dividiendo el dato de potencia activa o reactiva
```

```
    % del nodo de nuestra red entre la maxima potencia activa o reactiva del
    % nodo equivalente de la red de la que estamos utilizando las lecturas.
    k3_p=P3_dato/max(P3);
    k5_p=P5_dato/max(P5);
    k7_p=P7_dato/max(P7);
    k8_p=P8_dato/max(P8);
    k9_p=P9_dato/max(P9);
    k3_q=Q3_dato/max(Q3);
    k5_q=Q5_dato/max(Q5);
    k7_q=Q7_dato/max(Q7);
    k8_q=Q8_dato/max(Q8);
    k9_q=Q9_dato/max(Q9);
%------------------------------------------------------------------------
% ------------------------------------------------------------------------
%                           BUS DATA (CDF)
% ------------------------------------------------------------------------
% ------------------------------------------------------------------------
    %  Bus Type
    %                   0 - Unregulated (load, PQ)
    %                   1 - Hold MVAR generation within voltage limits, (PQ)
    %                   2 - Hold voltage within VAR limits (gen, PV)
    %                   3 - Hold voltage and angle (swing, V-Theta) (must always
    %                       have one)
% ------------------------------------------------------------------------
    % Se introducen los datos de todas las barras de la red
    %
    % Numero de barra
    Bus_Number=[1,2,3,4,5,6,7,8,9];
    % Tipo de barra de acuerdo con el formado CDF
    Bus_Type=[3,0,0,0,0,0,0,0,0];
    % Tension nominal de las barras
    Nominal_kV=[220,220,132,132,30,30,13.8,13.8,13.8];
    % P_demandada [MW]
    Pd=[0,0,P3(x)*k3_p,0,P5(x)*k5_p,0,P7(x)*k7_p,P8(x)*k8_p,P9(x)*k9_p];
    % P_generada [MW]
    Pg=[0,0,0,0,0,0,0,0,0];
    % Q_demandada [Mvar
    Qd=[0,0,Q3(x)*k3_q,0,Q5(x)*k5_q,0,Q7(x)*k7_q,Q8(x)*k8_q,Q9(x)*k9_q];
    % Q_generada [Mvar]
    Qg=[0,0,0,0,0,0,0,0,0];
    % Tensión [pu]
    V=[1,1,1,1,1,1,1,1,1];
    % Ángulo de las tensiones [radianes]
    Theta=[0,0,0,0,0,0,0,0,0];
    % Se fija barra slack V=1 [pu] & Theta=0
    % Resto de variables desconocidas: V=1 & Theta=0 (Arranque plano) ; P=Q=0
% ------------------------------------------------------------------------
    % | Bus Number | Bus Type | Nominal kV | Load MW | Load Mvar | V[pu] | X[pu] |
    Bus_Data=[Bus_Number',Bus_Type',Nominal_kV',Pd',Pg',Qd',Qg',V',Theta'];
% ------------------------------------------------------------------------
    % Potencia Base S=100 [MVA]
    Sbase=100;
    % Tensión en las barras [pu]
    V=V';
    % Ángulos de las tensiones [radianes]
    Theta=Theta';
    % Pg: Potencia activa generada [pu]
    Pg=Pg'/Sbase;
    % Pd: Potencia activa demandada [pu]
    Pd=Pd'/Sbase;
```

```
    % Qg: Potencia reactiva generada [pu]
    Qg=Qg'/Sbase;
    % Qd: Potencia reactiva demandada [pu]
    Qd=Qd'/Sbase;
    % Pi=Pgi-Pdi: Potencia activa inyectada a la red desde la barra i
    P=Pg-Pd;
    % Qi=Qgi-Qdi: Potencia reactiva inyectada a la red desde la barra i
    Q=Qg-Qd;
    % P especificada
    Pesp=P;
    % Q especificada
    Qesp=Q;
%--------------------------------------------------------------------------
    % Cálculo del número de barras PV y PQ en el sistema
    % Solo hay barras PQ
%--------------------------------------------------------------------------
    % PV y PQ: Devuelve la posición de las barras tipo 3 (Slack), tipo 2(PV)
    % y tipo 0 (PQ) en el vector "Bus_Type". Es decir, nos dice que barras
    % son PV y PQ
    PV=find(Bus_Type==3 | Bus_Type==2);
    PQ=find(Bus_Type==0);
    % Número de barras PV (incluyendo la barra slack)
    nPV=length(PV);
    % Número de barras PQ
    nPQ=length(PQ);
%--------------------------------------------------------------------------
    % Cálculo del vector incógnitas X=[Theta | V] -> Ángulos y tensiones
    % desconocidas en todas las barras menos la barra slack: [Theta2...Theta9 |
V2...V9]
%--------------------------------------------------------------------------
    % Vx: Tensiones incógnitas (En todas las barras PQ) -> [V2...V9]
     k=1;
     Vx=zeros(nPQ,1);
     for i=1:nb
        if Bus_Type(i)==0
            Vx(k,1)=V(i);
            k=k+1;
        end
     end
    % Thetax: Ángulos incógnitas (Todas las barras menos slack) -> [Theta2...Theta9]
    Thetax=Theta(2:nb);
    % X: Vector Incógnitas -> [Theta2...Theta9 | V2...V9]
    X=[Thetax;Vx];
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
%       Ecuaciones de potencia y cálculo mediante Newton-Raphson
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
    % No se plantean las ecuaciones de potencia en la barra slack
    % Se plantean las potencias [P] en todas las barras PV y PQ -> [P]=[P2...P9]
    % Se plantean las potencias [Q] en las barras PQ -> [Q]=[Q2...Q9]
%--------------------------------------------------------------------------
    %      Ecuaciones del flujo de cargas en la barra "i" en forma polar:
    %P=V(i)*V(k)*[G(i,k)*cos(Theta(i)-Theta(k))+B(i,k)*sin(Theta(i)-Theta(k))]
    %Q=V(i)*V(k)*[G(i,k)*sin(Theta(i)-Theta(k))-B(i,k)*cos(Theta(i)-Theta(k))]
    %                          Para k=1:nb
    %                 P= Potencia activa estimada
    %                 Q= Potencia reactiva estimada
%--------------------------------------------------------------------------
    % Se va a realizar el método de Newton-Raphson para una tolerancia de 10-5
```

```
    % Es decir, hasta que todos los términos de la matriz R=[dP | dQ] sean
    % menores que 10-5 no se dentendrá el cálculo iterativo
%-------------------------------------------------------------------------
    % [dP]=[Pesp-Pe]: residuos de potencia activa -> [dP]=[dP2...dP9]
    % [dQ]=[Qesp-Qe]: residuos de potencia reactiva -> [dQ]=[dQ2...dQ9]
    % [R]=[dP2...dP9 | dQ2...dQ9]
%-------------------------------------------------------------------------
    % Se inicia la tolerancia para un valor cualquiera mayor que 10-5
    % Se inicia el nº de iteración en 1. Tras cada iteración este valor se irá
    % incrementando en 1 hasta que termine el cálculo iterativo. Al final,
    % se conocerán el nº de iteraciones que se realizaron.
    tol=1;
    iter=1;
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
%   Comienza el bucle para el cálculo iterativo mediante Newton-Rapson
%-------------------------------------------------------------------------
%-------------------------------------------------------------------------
    while tol>1e-5;
    %Potencias activas estimadas (P1...P9)
        P=zeros(nb,1);
        for i=1:nb
            for k=1:nb
                P(i)=P(i)+V(i)*V(k)*[G(i,k)*cos(Theta(i)-Theta(k))+B(i,k)*sin(Theta(i)-
Theta(k))];
            end
        end
    %Potencias reactivas estimada (Q1...Q9)
        Q=zeros(nb,1);
        for i=1:nb
            for k=1:nb
                Q(i)=Q(i)+V(i)*V(k)*[G(i,k)*sin(Theta(i)-Theta(k))-B(i,k)*cos(Theta(i)-
Theta(k))];
            end
        end
%-------------------------------------------------------------------------
    % Residuos de potencias de todas las barras [R]=[dP1..dP9 | dQ1..dQ9]
        dPbarras=Pesp-P;
        dQbarras=Qesp-Q;
    % Selección de los residuos de potencias que nos interesan. Se van a
    % utilizar las potencias P y Q conocidas -> P2...P9 y Q2...Q9
    % No se incluye la barra slack
%-------------------------------------------------------------------------
    % Residuos [dP]
        dP=dPbarras(2:nb);
    % Se utilizan todas las P conocidas, esto es, de las barras PQ
%-------------------------------------------------------------------------
    % Residuos [dQ]
        k=1;
        dQ=zeros(nPQ,1);
        for i=1:nb
            if Bus_Type(i)==0
                dQ(k,1)=dQbarras(i);
                k=k+1;
            end
        end
    % Se utilizan todas las Q conocidas, esto es, de las barras PQ
%-------------------------------------------------------------------------
    %Residuos de Potencia [R]
        R=[dP;dQ];
```

```
%-------------------------------------------------------------------------
    % Cálculo del Jacobiano [J]
%-------------------------------------------------------------------------
    % Terminos del Jacobiano
    % [J]=[H N;M L]

    % H(i,k)=dP(i)/dTheta(k)          N(i,k)=V(k)*dP(i)/dV(k)
    % M(i,k)=dQ(i)/dTheta(k)          L(i,k)=V(k)*dQ(i)/dV(k)

    % Para i=k
    % H(i,i)=-Q(i)-B(i,i)*V(i)^2       N(i,i)=P(i)+G(i,i)*V(i)^2
    % L(i,i)=Q(i)-B(i,i)*V(i)^2        M(i,i)=P(i)-G(i,i)*V(i)^2

    % Para i distinto k
    % H(i,k)=L(i,k)=V(i)*V(k)*[G(i,k)*sin(Theta(i)-Theta(k))-B(i,k)*cos(Theta(i)-
Theta(k))];
    % N(i,k)=-M(i,k)=V(i)*V(k)*[G(i,k)*cos(Theta(i)-Theta(k))+B(i,k)*sin(Theta(i)-
Theta(k))];
%-------------------------------------------------------------------------
    % Cálculo de [H]
        H=zeros(nb-1);
        for i=1:nb-1
            m=i+1;
            for k=1:nb-1
                n=k+1;
                if n==m
                    H(i,k)=-Q(m)-B(m,m)*(V(m))^2;
                else
                    H(i,k)=V(m)*V(n)*[G(m,n)*sin(Theta(m)-Theta(n))-
B(m,n)*cos(Theta(m)-Theta(n))];
                end
            end
        end
%-------------------------------------------------------------------------
    % Cálculo de [N]
        N=zeros(nb-1,nPQ);
        for i=1:nb-1
            m=i+1;
            for k=1:nPQ
                n=PQ(k);
                if n==m
                    N(i,k)=P(m)+G(m,m)*(V(m))^2;
                else
                    N(i,k)=V(m)*V(n)*[G(m,n)*cos(Theta(m)-
Theta(n))+B(m,n)*sin(Theta(m)-Theta(n))];
                end
            end
        end
%-------------------------------------------------------------------------
    % Cálculo de [M]
        M=zeros(nPQ,nb-1);
        for i=1:nPQ
            m=PQ(i);
            for k=1:nb-1
                n=k+1;
                if n==m
                    M(i,k)=P(m)-G(m,m)*(V(m))^2;
                else
```

```
                    M(i,k)=-V(m)*V(n)*[G(m,n)*cos(Theta(m)-
Theta(n))+B(m,n)*sin(Theta(m)-Theta(n))];
                end
            end
        end
%-------------------------------------------------------------------------
    % Cálculo de [L]
        L=zeros(nPQ,nPQ);
        for i=1:nPQ
            m=PQ(i);
            for k=1:nPQ
                n=PQ(k);
                if n==m
                    L(i,k)=Q(m)-B(m,m)*(V(m))^2;
                else
                    L(i,k)=V(m)*V(n)*[G(m,n)*sin(Theta(m)-Theta(n))-
B(m,n)*cos(Theta(m)-Theta(n))];
                end
            end
        end
%-------------------------------------------------------------------------
    % Jacobiano [J]
        J=[H N;M L];
    % Convierto en matriz dispersa el Jacobiano
        J=sparse(J);
%-------------------------------------------------------------------------
    % Cálculo de [dX]=[dTheta | dV/V] resolviendo el sistema: [R]=[J]*[dX]
    % [dX] es el Vector de Errores -> [dX]=[J]\[R]
        dX=J\R;
%-------------------------------------------------------------------------
    % Como [dX]=[dTheta | dV/V], vamos a obtener [dX]=[dTheta | dV]
    % multiplicando [dV/V]*[V], donde [V] son las tensiones incógnitas
        for i=nb:length(dX)
            dX(i)=dX(i)*X(i);
        end
%-------------------------------------------------------------------------
    % El vector [dX] se puede dividir en [dTheta] y [dV] ya que [dX]=[dTheta | dV]
        dTheta=dX(1:nb-1);
        dV=dX(nb:end);
%-------------------------------------------------------------------------
    % Actualizar las variables [Theta] y [V]
%-------------------------------------------------------------------------
    % [Theta]nueva=[Theta]antigua+[dTheta]
        Theta(2:nb)=Theta(2:nb)+dTheta;
    % No se modifica el ángulo de la barra slack que está fijado en 0°
%-------------------------------------------------------------------------
    % [V]nueva=[V]antigua+[dV]
        k=1;
        for i=2:nb
            if Bus_Type(i)==0
                V(i)=V(i)+dV(k);
                k=k+1;
            end
        end
    % No se modifica el modulo de la tension de la barra slack que está
    % fijado en 1 [pu]
%-------------------------------------------------------------------------
%               COMPROBAR TOLERANCIA Y SI NO REPETIR BUCLE
%-------------------------------------------------------------------------
        tol=max(abs(R));
```

```
        iter=iter+1;
    end
%--------------------------------------------------------------------------
%                VALORES POR UNIDAD -> VALORES REALES
%--------------------------------------------------------------------------
    % P[pu]*Sbase=P[MW]
    % Q[pu]*Sbase=Q[MW]
    % V[pu]*Vbase=V[V]
    P=P*Sbase;
    Q=Q*Sbase;
    V=V.*Nominal_kV';
%--------------------------------------------------------------------------
    % Modificamos el vector que recoge los resultados de cada iteracion del
    % nodo 9 para posteriormente visualizar el perfil de cargas diario de
    % este nodo.
    % Además, cambiamos el signo de las potencias obtenidas para graficar
    % como positivas las potencias demandadas, y como potencias negativas
    % las potencias generadas:
    % (+) -> Potencias demandadas
    % (-) -> Potencias generadas
    P_plot(x)=P(9)*(-1);
    Q_plot(x)=Q(9)*(-1);
%--------------------------------------------------------------------------
    % Modificamos el vector que recoge los resultados de cada iteracion del
    % nodo 9 para posteriormente visualizar el perfil de tensiones diario de
    % este nodo.
    V_plot(x)=V(9);
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
%         SI SE QUIEREN MOSTRAR LOS RESULTADOS DE CADA ITERACION
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
% Vamos a convertir los ángulos de radianes a grados: [radianes]->[grados]
% Theta_radianes=Theta;
% Theta_grados= Theta*180/pi;
% Phase_Shift=[0,0,-30,-30,-30,-30,-30+30,-30+30,-30+30]; % Incluimos el índice horario
de los trafos
% Theta_grados=Theta_grados-Phase_Shift';
%--------------------------------------------------------------------------
% disp(['Potencia activa [P]: ',num2str(P')])
% disp(['Potencia reactiva [Q]: ',num2str(Q')])
% disp(['Tensiones [V]: ',num2str(V')])
% disp(['Angulos de las tensiones en radianes [Theta_radianes]:
',num2str(Theta_radianes')])
% disp(['Angulos de las tensiones en grados [Theta_grados]: ',num2str(Theta_grados')])
% disp(['Numero de iteraciones: ',num2str(iter)])
% disp(' ')
%--------------------------------------------------------------------------
end
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
%                      PERFIL DE CARGAS DIARIO
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
% Graficamos el perfil de potencias (P y Q) del nodo 9
figure(1);          % Abrimos nueva grafica
plot(P_plot,'g')    % P se dibuja en color verde
grid on             % Utilizamos 'grid on' para añadir lineas de cuadriculas a la
grafica
title('Perfil de cargas diario: Nodo 9')
```

```
xlabel('Tiempo')
ylabel('Potencia [MW]')
legend('P')
% Graficamos el perfil de potencia reactiva [Q] del nodo 9
figure(2);          % Abrimos nueva grafica
plot(Q_plot,'b')    % Q se dibuja en color azul
grid on
% Añadimos el titulo de la grafica, el nombre de los ejes, y la leyenda
title('Perfil de cargas diario: Nodo 9')
xlabel('Tiempo')
ylabel('Potencia [Mvar]')
legend('Q')
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
%                      PERFIL DE TENSIONES DIARIO
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
figure(3);          % Abrimos una nueva grafica
plot(V_plot,'r')    % V se dibuja en color rojo
grid on
% Añadimos el titulo de la grafica, el nombre de los ejes, y la leyenda
title('Perfil de tensiones diario: Nodo 9')
xlabel('Tiempo')
ylabel('Tension [kV]')
legend('V')
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
toc
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
%                                   FIN
%--------------------------------------------------------------------------
%--------------------------------------------------------------------------
```

```
#-------------------------------------------------------------------------
#                           FLUJO DE CARGAS
#-------------------------------------------------------------------------
#                          Objetivo del estudio:
# Obtener las condiciones de funcionamiento (tensiones de barra, corrientes
# circulantes, caídas de tensión, pérdidas, etc.) de un Sistema Eléctrico
# de potencia (SEP) en régimen permanente (RP)
# En nuestro caso se van a determinar las tensiones complejas (modulo y angulo)
# y potencias (P,Q) en cada barra del sistema o red
#-------------------------------------------------------------------------
#           Pasos a seguir para el planteamiento del problema:
# 1- Construir el equivalente por fase del sistema con todas las
#    variables y parámetros representados en por unidad en unas bases
#    comunes (PowerWorld)
# 2- Construir la Matriz de Admitancias de barra [Ybarra]
# 3- Clasificar las barras
# 4- Plantear las ecuaciones de potencia
# 5- Resolución de los flujos de carga (Método de Newton-Raphson)
#-------------------------------------------------------------------------
#           Algoritmo de cálculo (Método de Newton-Raphson):
# 1- Inicializar las tensiones con un perfil plano (1<0º pu en todas
#    las barras)
# 2- Calcular R=[dP | dQ]. Si todos los elementos de este vector son
#    menores que la tolerancia detener el proceso. En caso contrario,
#    calcular el Jacobiano [J] y continuar
# 3- Obtener dX=[dTheta | dV/V] resolviendo el sistema: [R]=[J]*[dX]
# 4- Actualizar X=[Theta | V] y volver al paso 2. [X]nueva=[X]antigua+[dX]
#-------------------------------------------------------------------------
#-------------------------------------------------------------------------
#    Codigo para conocer el tiempo que Python tarda en ejecutar el script:
# import time
# start=time.time()
# --codigo--
# end=time.time()
# elapsed=end-start
# print(elapsed)
#-------------------------------------------------------------------------
import time
start=time.time()
#-------------------------------------------------------------------------
# Importamos el modulo "numpy" para trabajar con "arrays" de aqui en adelante.
import numpy as np
# np es una abreviatura de numpy -> Ahora, los sub-modulos y las funciones
# contenidas en el modulo "numpy" se pueden llamar escribiendo np.__() en lugar
# de numpy.__()
#-------------------------------------------------------------------------
#-------------------------------------------------------------------------
#                            BRANCH DATA (CDF [17])
#-------------------------------------------------------------------------
#-------------------------------------------------------------------------
# Branch Type       0 - Transmission line
#                   1 - Fixed tap
#                   2 - Variable tap for voltage control (TCUL, LTC)
#                   3 - Variable tap (turns ratio) for MVAR control
#                   4 - Variable phase angle for MW control (phase shifter)
#-------------------------------------------------------------------------
# Se introducen los datos de todas las lineas de la red
#
# Numero de linea o "branch"
Branch=np.array([1,2,3,4,5,6,7,8])
# Desde la barra "i"
```

```
From_Bus=np.array([1,2,3,4,5,6,3,8])
# Hasta la barra "k"
To_Bus=np.array([2,3,4,5,6,7,8,9])
# Tipo de linea de acuerdo con el formado CDF
Branch_Type=np.array([0,2,0,2,0,2,2,0])
# Resistencia [R] de la linea en por unidad
R_pu=np.array([0.000243,0.001667,0.001386,0.008000,0.018933,0.095000,0.006133,0.152174])
# Reactancia [X] de la linea en por unidad
X_pu=np.array([0.002331,0.023889,0.001300,0.080000,0.004433,0.480000,0.056667,0.105860])
# Tap de los transformadores
Tap=np.array([1,1.0125,1,0.9875,1,0.9250,0.9750,1])     # Tap=1 para lineas
#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
#                        MATRIZ DE ADMITANCIAS [Ybarra]
#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
# Se renombran los vectores por comodidad de escritura
R=R_pu
X=X_pu
a=Tap
# Impedancias de las líneas [z]
z=np.zeros((8),dtype=complex) # Creamos un vector fila 'complejo' de ceros de dimension 1x8 [0+0j,0+0j...]
z.real=np.array(R) # Parte real de las impedancias (Resistencias [R])
z.imag=np.array(X) # Parte imaginaria de las impedancias (Reactancias [X])
# Admitancias de las líneas [y]
y=1/z
# Número de líneas
nl=len(From_Bus)
# Número de barras
nb=max(max(From_Bus),max(To_Bus))
#-------------------------------------------------------------------------------
# Construcción Matriz Admitancias [Y]
Ybarra=np.zeros((nb,nb),dtype=complex) # Creamos matriz de ceros de dimension (9x9)
# Elementos fuera de la diagonal
for i in range (nl):
    Ybarra[From_Bus[i]-1,To_Bus[i]-1]= Ybarra[From_Bus[i]-1,To_Bus[i]-1]-y[i]/a[i]
    Ybarra[To_Bus[i]-1,From_Bus[i]-1]=Ybarra[From_Bus[i]-1,To_Bus[i]-1]
# Elementos de la diagonal
for i in range (nb):
    for k in range (nl):
        if From_Bus[k]==i+1:
            Ybarra[i,i]=Ybarra[i,i]+(y[k]/a[k])+[y[k]*(1-a[k])/(a[k]**2)]
        elif To_Bus[k]==i+1:
            Ybarra[i,i]=Ybarra[i,i]+(y[k]/a[k])+[y[k]*(a[k]-1)/a[k]]
# Parte real [G:conductancia] e imaginaria [B:susceptancia] de las admitancias
# de las lineas [Y=G+jB]
G=Ybarra.real  # Conductancia [G]
B=Ybarra.imag  # Susceptancia [B]
#-------------------------------------------------------------------------------
# Creamos vectores fila de ceros de dimension (1 x samples) para ir recogiendo
# los resultados de potencia (P y Q) y de tension (V) del nodo 9. Cada vez que
# se resuelva el flujo de cargas, los resultados de P,Q,V del nodo 9 se iran
# almacenando en estos vectores fila, sustituyendo asi los valores originiales
# de estos vectores, los cuales eran todos cero.
#
# La variable 'samples' indica el numero de medidas de potencias que queremos
# utilizar para el flujo de cargas -> Utilizamos 60 medidas, aunque este programa
# esta preparado para leer 1440 medidas. Es decir, se va a resolver el flujo
# de cargas 60 veces, utilizando cada vez nuevos datos de potencia proporcionados
# por el PowerMeter y por una base de datos de potencias con 1440 medidas.
#
# Estos vectores nos serviran para dibujar el perfil diario de cargas y tensiones
# al final del programa.
```

```
samples=60
P_plot=np.zeros((samples))
Q_plot=np.zeros((samples))
V_plot=np.zeros((samples))
#------------------------------------------------------------------------------
#------------------------------------------------------------------------------
#                 CODIGO PARA LECTURA DATOS POWER METER
#------------------------------------------------------------------------------
#------------------------------------------------------------------------------
# from pymodbus3.client.sync import ModbusTcpClient
# from pymodbus3.constants import Endian
# --codigo--
# f.close()
# client.close()
#------------------------------------------------------------------------------
from pymodbus3.client.sync import ModbusTcpClient
from pymodbus3.constants import Endian
from pymodbus3.payload import BinaryPayloadDecoder
import time
import threading
from datetime import datetime
from datetime import timedelta
#
def boolList2BinString(lst):
    return '0b' + ''.join(['1' if x else '0' for x in lst])
#
client = ModbusTcpClient('80.31.186.204') # IP para conectarse al PowerMeter
#
f = open('registro.txt', 'a')
# Abre un archivo.txt llamado 'registro', en el que se ira registrando cada
# medida del PowerMeter con el siguiente formato: fecha - hora - V[V] - I[A] - P[kW] - Q[kvar]
#
now_ref = datetime.now()
total_delay = 3
for x in range(samples):
    print ("Medida PowerMeter nº:",x+1)
    #
    # Reading date and time from smart meter
    # Reading registers
    year = client.read_holding_registers(1844,1)
    m_day = client.read_holding_registers(1845,1)
    h_min = client.read_holding_registers(1846,1)
    mili = client.read_holding_registers(1847,1)
    # Decoding
    year_decoder = BinaryPayloadDecoder.from_registers(year.registers, endian=Endian.Big)
    m_day_decoder = BinaryPayloadDecoder.from_registers(m_day.registers, endian=Endian.Big)
    h_min_decoder = BinaryPayloadDecoder.from_registers(h_min.registers, endian=Endian.Big)
    mili_decoder = BinaryPayloadDecoder.from_registers(mili.registers, endian=Endian.Big)
    # Decoding year
    yr = year_decoder.decode_bits()
    yr = year_decoder.decode_bits()
    yr_m = yr[6],yr[5],yr[4],yr[3],yr[2],yr[1],yr[0]
    # Decoding month and day
    mt = m_day_decoder.decode_bits()
    dy = m_day_decoder.decode_bits()
    mt_m = mt[5],mt[4],mt[3],mt[2],mt[1],mt[0]
    dy_m = dy[4],dy[3],dy[2],dy[1],dy[0]
    # Decoding hour and minute
    hr = h_min_decoder.decode_bits()
    hr_m = hr[4],hr[3],hr[2],hr[1],hr[0]
    mn = h_min_decoder.decode_bits()
    mn_m = mn[5],mn[4],mn[3],mn[2],mn[1],mn[0]
    # Decoding miliseconds
    ml = mili_decoder.decode_16bit_uint()
```

```
    # Variables are represented as integer values
    year = int(boolList2BinString(yr_m),2)
    month = int(boolList2BinString(mt_m),2)
    day = int(boolList2BinString(dy_m),2)
    hour = int(boolList2BinString(hr_m),2)
    minute = int(boolList2BinString(mn_m),2)
    miliseconds = ml
    # Shown date and time
    # print str(day)+'/'+str(month)+'/'+str(year)+' '+str(hour)+':'+str(minute)+':'+str(miliseconds)
    #
    # Read registers
    # Voltage phase 'a' to neutral (Van)
    Van = client.read_holding_registers(3027, 2)
    # Current phase 'a' (Ia)
    Ia = client.read_holding_registers(2999, 2)
    # Active power phase 'a' (Pa)
    Pa = client.read_holding_registers(3053, 2)
    # Reactive power phase 'a' (Qa)
    Qa = client.read_holding_registers(3061, 2)
    # Decoding (FLOAT32)
    Van_decoder = BinaryPayloadDecoder.from_registers(Van.registers, endian=Endian.Big)
    Ia_decoder = BinaryPayloadDecoder.from_registers(Ia.registers, endian=Endian.Big)
    Pa_decoder = BinaryPayloadDecoder.from_registers(Pa.registers, endian=Endian.Big)
    Qa_decoder = BinaryPayloadDecoder.from_registers(Qa.registers, endian=Endian.Big)
    Van_f = Van_decoder.decode_32bit_float()
    Ia_f = Ia_decoder.decode_32bit_float()
    Pa_f = Pa_decoder.decode_32bit_float()
    Qa_f = Qa_decoder.decode_32bit_float()
    # Showing values [V],[P],[Q]
    print ("V9[V]:"+ str(Van_f)+"   ","P9[kW]:"+str(Pa_f)+"   ","Q9[kvar]:"+str(Qa_f))
    # Mostramos los valores de V,P,Q. Estas medidas son las que asignamos al
    # nodo 9.
#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
#                   LECTURA DE POTENCIAS Y ESCALADO
#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
    # Importamos el archivo 'ADRES_2Sets_1min.mat' desde MATLAB a Python.
    #
    # Este archivo contiene lecturas de P y Q de 5 nodos de otra red, registradas
    # durante cada minuto a lo largo de un dia completo. Es decir, cada nodo tiene
    # registrado 1440 lecturas de P y 1440 lecturas de Q.
    #
    # El objetivo es simular que estos 5 nodos se corresponden con los 5 nodos
    # con carga de nuestra red. De esta forma, las potencias demandadas en cada nodo
    # ira cambiando a lo largo del dia, lo que se ajusta a la situacion real.
    #
    # En realidad, del archivo.mat solo vamos a utilizar las medidas de los
    # 4 primeros nodos. Las lecturas de potencia correspondientes al ultimo nodo
    # son las potencias medidas a traves del PowerMeter.
    #
    # En resumen, 4 nodos se simulan con las potencias del archivo.mat, y un nodo
    # se simula con las potencias medidas con el PowerMeter.
    #
    # Utilizamos el modulo "scipy" para leer en Python el archivo de MATLAB
    from scipy.io import loadmat
    potencias=loadmat('ADRES_2sets_1min.mat')
    # En el archivo, las lecturas de cada nodo "i" se llaman P_i_o_av y Q_i_o_av.
    #
    # Unidades de las lecturas (archivo.mat) -> P [MW] y Q[Mvar]
    # Unidades de las lecturas (PowerMeter) -> P [kW] y Q[kvar]
    #
    # Creamos un vector fila de dimension (1x1440) para cada potencia.
    P_1_o_av=potencias["P_1_o_av"]
```

```
    P_2_o_av=potencias["P_2_o_av"]
    P_3_o_av=potencias["P_3_o_av"]
    P_4_o_av=potencias["P_4_o_av"]
    # P_5_o_av=potencias["P_5_o_av"] ->  No lo utilizamos (PowerMeter)
    Q_1_o_av=potencias["Q_1_o_av"]
    Q_2_o_av=potencias["Q_2_o_av"]
    Q_3_o_av=potencias["Q_3_o_av"]
    Q_4_o_av=potencias["Q_4_o_av"]
    # Q_5_o_av=potencias["Q_5_o_av"] ->  No lo utilizamos (PowerMeter)
    #
    # Renombramos las potencias para nuestra red.
    # De esta manera los 4 nodos del archivo 'ADRES_2Sets_1min.mat' serían en
    # nuestra red los nodos 3,5,7,8. El nodo 9 se corresponde con las medidas
    # del PowerMeter.
    P3=P_1_o_av
    P5=P_2_o_av
    P7=P_3_o_av
    P8=P_4_o_av
    P9=Pa_f
    Q3=Q_1_o_av
    Q5=Q_2_o_av
    Q7=Q_3_o_av
    Q8=Q_4_o_av
    Q9=Qa_f
    # Datos de la red -> P[MW] y Q[Mvar]
    P3_dato=84
    P5_dato=34
    P7_dato=4.9
    P8_dato=52
    P9_dato=2.7
    Q3_dato=26
    Q5_dato=12
    Q7_dato=12.6
    Q8_dato=39
    Q9_dato=-3.4
    # Para asignar las lecturas de potencia del archivo.mat y del PowerMeter
    # a los nodos de nuestra red es necesario realizar un escalado de los datos
    #mediante la siguiente operacion:
    # (Lectura de potencia * k = Potencia "escalada" en nuestro nodo)
    #
    # k es la constante de escalado -> ki=Pi_dato/max(Pi) ; ki=Qi_dato/max(Qi)
    #
    # Ejemplo para nodo 3 (potencia activa) -> k3_p=84[MW]/max(P9)[MW]
    # Ejemplo para nodo 3 (potencia reactiva) -> k3_q=26[Mvar]/max(P9)[Mvar]
    #
    # Es decir, se calcula dividiendo el dato de potencia activa o reactiva del nodo
    # de nuestra red entre la maxima potencia activa o reactiva del nodo equivalente
    # de la red de la que estamos utilizando las lecturas.
    # Esta operacion es valida para los nodos 3,5,7,8.
    #
    # Como el nodo 9 toma las potencias del PowerMeter, el escalado se hace tomando
    # como la maxima potencia activa 18[W], por ser el maximo consumo medido
    # por el PowerMeter cuando tiene una pequeña carga conectada.
    # De esta forma -> k9=(1/1000)*c
    # (Se divide por (1/1000) porque el PowerMeter da la medida en kW, y queremos
    # que este en MW para nuestra red)
    # siendo c=(2.7[MW]*1000000)/18[W]
    # Asi conseguimos escalar las potencias medidas con el PowerMeter (en kW y kvar)
    # y adecuarlas a nuestra red (en MW y Mvar).
    k3_p=P3_dato/max(P3[0,])
    k5_p=P5_dato/max(P5[0,])
    k7_p=P7_dato/max(P7[0,])
    k8_p=P8_dato/max(P8[0,])
    c=(P9_dato*1e6)/18
```

```
    k9_p=(1/1e3)*c
    k3_q=Q3_dato/max(Q3[0,])
    k5_q=Q5_dato/max(Q5[0,])
    k7_q=Q7_dato/max(Q7[0,])
    k8_q=Q8_dato/max(Q8[0,])
    # Utilizamos la misma constante de escalado para Q que para P en el nodo 9.
    k9_q=k9_p
#------------------------------------------------------------------------------
#------------------------------------------------------------------------------
#                               BUS DATA (CDF)
#------------------------------------------------------------------------------
#------------------------------------------------------------------------------
    #  Bus Type
    #                   0 - Unregulated (load, PQ)
    #                   1 - Hold MVAR generation within voltage limits, (PQ)
    #                   2 - Hold voltage within VAR limits (gen, PV)
    #                   3 - Hold voltage and angle (swing, V-Theta) (must always
    #                       have one)
#------------------------------------------------------------------------------
    # Se introducen los datos de todas las barras de la red
    #
    # Numero de barra
    Bus_Number=np.array([1,2,3,4,5,6,7,8,9])
    # Tipo de barra de acuerdo con el formado CDF
    Bus_Type=np.array([3,0,0,0,0,0,0,0,0])
    # Tension nominal de las barras
    Nominal_kV=np.array([220,220,132,132,30,30,13.8,13.8,13.8])
    # P_demandada [MW]
    Pd=np.array([0,0,P3[0,x]*k3_p,0,P5[0,x]*k5_p,0,P7[0,x]*k7_p,P8[0,x]*k8_p,P9*k9_p])
    # P_generada [MW]
    Pg=np.array([0,0,0,0,0,0,0,0,0])
    # Q_demandada [Mvar]
    Qd=np.array([0,0,Q3[0,x]*k3_q,0,Q5[0,x]*k5_q,0,Q7[0,x]*k7_q,Q8[0,x]*k8_q,Q9*k9_q])
    # Q_generada [Mvar]
    Qg=np.array([0,0,0,0,0,0,0,0,0])
    # Tensión [pu]
    V=np.array([1,1,1,1,1,1,1,1,1],dtype=float)
    # Ángulo de las tensiones [radianes]
    Theta=np.array([0,0,0,0,0,0,0,0,0],dtype=float)
    # Se fija en la barra slack V=1 [pu] & Theta=0
    # Resto de variables desconocidas: V=1 & Theta=0 (Arranque plano) ; P=Q=0
#------------------------------------------------------------------------------
    # Potencia Base S=100 [MVA]
    Sbase=100
    # Pg: Potencia activa generada [pu]
    Pg=Pg/Sbase
    # Qg: Potencia reactiva generada [pu]
    Qg=Qg/Sbase
    # Pd: Potencia activa demandada [pu]
    Pd=Pd/Sbase
    # Qd: Potencia reactiva demandada [pu]
    Qd=Qd/Sbase
    # Pi=Pgi-Pdi: Potencia activa inyectada a la red desde la barra i
    P=Pg-Pd
    # Qi=Qgi-Qdi: Potencia reactiva inyectada a la red desde la barra i
    Q=Qg-Qd
    # P especificada
    Pesp=P
    # Q especificada
    Qesp=Q
#------------------------------------------------------------------------------
    # Cálculo del número de barras PV y PQ en el sistema
    # Solo hay barras PQ
#------------------------------------------------------------------------------
```

```
    # PV y PQ: Devuelve la posición de las barras tipo 3 (Slack) y tipo 0 (PQ)
    # en el vector "Bus_Type". Es decir, nos dice que barras son PV y PQ
    PV=np.where(Bus_Type==3)  # Como no hay barras PV, buscamos solo barras tipo 3
    PQ=np.where(Bus_Type==0)
    # np.where devuelve un array de dimension 1x2. El primer elemento [0] del array
    # es un array con los indices o posiciones de "3" o "0" en "Bus_Type".
    # El segundo elemento [1] del array devuelto por np.where indica el formato del
    # array del primer elemento [0].
    #
    # Número de barras PV (incluyendo la barra slack)
    nPV=len(PV[0])
    # Número de barras PQ
    nPQ=len(PQ[0])
#-------------------------------------------------------------------------------
    # Cálculo del vector incógnitas X=[Theta | V] -> Ángulos y tensiones desconocidas
    # en todas las barras menos la barra slack: [Theta2...Theta9 | V2...V9]
#-------------------------------------------------------------------------------
    # Vx: Tensiones incógnitas (En todas las barras PQ) -> [V2...V9]
    k=0
    Vx=np.zeros((nPQ))
    for i in range (nb):
        if Bus_Type[i]==0:
            Vx[k]=V[i]
            k=k+1
    # Thetax: Ángulos incógnitas (Todas las barras menos slack) -> [Theta2...Theta9]
    Thetax=Theta[1:]
    # X: Vector Incógnitas -> [Theta2...Theta9 | V2...V9]
    X=np.concatenate((Thetax,Vx))
#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
#         Ecuaciones de potencia y cálculo mediante Newton-Raphson
#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
    # No se plantean las ecuaciones de potencia en la barra slack
    # Se plantean las potencias [P] en todas las barras PV y PQ -> [P]=[P2...P9]
    # Se plantean las potencias [Q] en las barras PQ -> [Q]=[Q2...Q9]
#-------------------------------------------------------------------------------
    #      Ecuaciones del flujo de cargas en la barra "i" en forma polar:
    # P=V[i]*V[k]*(G[i,k]*cos(Theta[i]-Theta[k])+B[i,k]*sin(Theta[i]-Theta[k]))
    # Q=V[i]*V[k]*(G[i,k]*sin(Theta[i]-Theta[k])-B[i,k]*cos(Theta[i]-Theta[k]))
    #                       Para k in range (nb)
    #                   P= Potencia activa estimada
    #                   Q= Potencia reactiva estimada
#-------------------------------------------------------------------------------
    # Se va a realizar el método de Newton-Raphson para una tolerancia de 10-5.
    # Es decir, hasta que todos los términos de la matriz R=[dP | dQ] sean
    # menores que 10-5 no se dentendrá el cálculo iterativo.
#-------------------------------------------------------------------------------
    # [dP]=[Pesp-Pe]: residuos de potencia activa -> [dP]=[dP2...dP9]
    # [dQ]=[Qesp-Qe]: residuos de potencia reactiva -> [dQ]=[dQ2...dQ9]
    # [R]=[dP2...dP9 | dQ2...dQ9]
#-------------------------------------------------------------------------------
    # Se inicia la tolerancia para un valor cualquiera mayor que 10-5.
    # Se inicia el nº de iteración en 1. Tras cada iteración este valor se irá
    # incrementando en 1 hasta que termine el cálculo iterativo. Al final,
    # se conocerán el nº de iteraciones que se realizaron.
    tol=1
    iteracion=0
#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
#    Comienza el bucle para el cálculo iterativo mediante Newton-Rapson
#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
    while tol>1e-5:
```

```
    # Potencias activas estimadas (P1...P9)
        P=np.zeros((nb))
        for i in range (nb):
            for k in range (nb):
                P[i]=P[i]+V[i]*V[k]*(G[i,k]*np.cos(Theta[i]-Theta[k])+B[i,k]*np.sin(Theta[i]-Theta[k]))
    # Potencias reactivas estimada (Q1...Q9)
        Q=np.zeros(nb)
        for i in range (nb):
            for k in range (nb):
                Q[i]=Q[i]+V[i]*V[k]*(G[i,k]*np.sin(Theta[i]-Theta[k])-B[i,k]*np.cos(Theta[i]-Theta[k]))
#-------------------------------------------------------------------------
    # Residuos de potencias de todas las barras [R]=[dP1..dP9 | dQ1..dQ9]
        dPbarras=Pesp-P
        dQbarras=Qesp-Q
    # Selección de los residuos de potencias que nos interesan. Se van a
    # utilizar las potencias P y Q conocidas -> P2...P9 y Q2...Q9
    # No se incluye la barra slack
#-------------------------------------------------------------------------
    # Residuos [dP]
        dP=dPbarras[1:]
    # Se utilizan todas las P conocidas, esto es, de las barras PQ
#-------------------------------------------------------------------------
    # Residuos [dQ]
        k=0
        dQ=np.zeros((nPQ))
        for i in range (nb):
            if Bus_Type[i]==0:
                dQ[k]=dQbarras[i]
                k=k+1
    # Se utilizan todas las Q conocidas, esto es, de las barras PQ
#-------------------------------------------------------------------------
    # Residuos de Potencia [R]
        R=np.concatenate((dP,dQ))
#-------------------------------------------------------------------------
    # Cálculo del Jacobiano [J]
#-------------------------------------------------------------------------
    # Terminos del Jacobiano
    # [J]=[H N;M L]
    #
    # H[i,k]=dP[i]/dTheta[k]          N[i,k]=V[k]*dP[i]/dV[k]
    # M[i,k]=dQ[i]/dTheta[k]          L[i,k]=V[k]*dQ[i]/dV[k]
    #
    # Para i=k
    # H[i,i]=-Q[i]-B[i,i]*V[i]**2     N[i,i]=P[i]+G[i,i]*V[i]**2
    # L[i,i]=Q[i]-B[i,i]*V[i]**2      M[i,i]=P[i]-G[i,i]*V[i]**2
    #
    # Para i distinto k
    # H[i,k]=L[i,k]=V[i]*V[k]*(G[i,k]*sin(Theta[i]-Theta[k])-B[i,k]*cos(Theta[i]-Theta[k]))
    # N[i,k]=-M[i,k]=V[i]*V[k]*(G[i,k]*cos(Theta[i]-Theta[k])+B[i,k]*sin(Theta[i]-Theta[k]))
#-------------------------------------------------------------------------
    # Cálculo de [H]
        H=np.zeros((nb-1,nb-1))
        for i in range (nb-1):
            m=i+1
            for k in range (nb-1):
                n=k+1
                if n==m:
                    H[i,k]=-Q[m]-B[m,m]*(V[m])**2
                else:
                    H[i,k]=V[m]*V[n]*(G[m,n]*np.sin(Theta[m]-Theta[n])-B[m,n]*np.cos(Theta[m]-Theta[n]))
#-------------------------------------------------------------------------
```

```
    # Cálculo de [N]
        N=np.zeros((nb-1,nPQ))
        for i in range (nb-1):
            m=i+1
            for k in range (nPQ):
                n=PQ[0][k]
                if n==m:
                    N[i,k]=P[m]+G[m,m]*(V[m])**2
                else:
                    N[i,k]=V[m]*V[n]*(G[m,n]*np.cos(Theta[m]-Theta[n])+B[m,n]*np.sin(Theta[m]-
    Theta[n]))
#-----------------------------------------------------------------------------
    # Cálculo de [M]
        M=np.zeros((nPQ,nb-1))
        for i in range (nPQ):
            m=PQ[0][i]
            for k in range (nb-1):
                n=k+1
                if n==m:
                    M[i,k]=P[m]-G[m,m]*(V[m])**2
                else:
                    M[i,k]=-V[m]*V[n]*(G[m,n]*np.cos(Theta[m]-Theta[n])+B[m,n]*np.sin(Theta[m]-
    Theta[n]))
#-----------------------------------------------------------------------------
    # Cálculo de [L]
        L=np.zeros((nPQ,nPQ))
        for i in range (nPQ):
            m=PQ[0][i]
            for k in range (nPQ):
                n=PQ[0][k]
                if n==m:
                    L[i,k]=Q[m]-B[m,m]*(V[m])**2
                else:
                    L[i,k]=V[m]*V[n]*(G[m,n]*np.sin(Theta[m]-Theta[n])-B[m,n]*np.cos(Theta[m]-
    Theta[n]))
#-----------------------------------------------------------------------------
   # Jacobiano [J]
        # J=[H,N; M,L]
        J1=np.hstack([H,N]) # Junta horizontalmente los 'arrays' [H] y [N]. Dimension [J1]: 8x16
        J2=np.hstack([M,L]) # Junta horizontalmente los 'arrays' [M] y [L]. Dimension [J2]: 8x16
        J=np.vstack([J1,J2]) # Junta verticalmente los 'arrays' [J1] y [J2]. Dimeension [J]: 16x16
#-----------------------------------------------------------------------------
    # Cálculo de [dX]=[dTheta | dV/V] resolviendo el sistema: [R]=[J]*[dX]
    # [dX] es el Vector de Errores -> [dX]=inv(J)*[R]
        J_inverse=np.linalg.inv(J)
        dX=np.dot(J_inverse,R)
#-----------------------------------------------------------------------------
    # Como [dX]=[dTheta | dV/V], vamos a obtener [dX]=[dTheta | dV]
    # multiplicando [dV/V]*[V], donde [V] son las tensiones incógnitas
        for i in range (nb-1,len(dX)):
            dX[i]=dX[i]*X[i]
#-----------------------------------------------------------------------------
    # El vector [dX] se puede dividir en [dTheta] y [dV] ya que [dX]=[dTheta | dV]
        dTheta=dX[0:nb-1]
        dV=dX[nb-1:]
#-----------------------------------------------------------------------------
    # Actualizar las variables [Theta] y [V]
#-----------------------------------------------------------------------------
    # [Theta]nueva=[Theta]antigua+[dTheta]
        Theta[1:nb]=Theta[1:nb]+dTheta
    # No se modifica el ángulo de la barra slack que está fijado en 0º
#-----------------------------------------------------------------------------
    # [V]nueva=[V]antigua+[dV]
        k=1
```

```
        for i in range (nb-1):
            if Bus_Type[k]==0:
                V[k]=V[k]+dV[i]
                k=k+1
    # No se modifica el modulo de la tension de la barra slack que está fijado en 1 [pu]
#-------------------------------------------------------------------------------
#               COMPROBAR TOLERANCIA Y SI NO REPETIR BUCLE
#-------------------------------------------------------------------------------
        tol=max(abs(R))
        iteracion=iteracion+1
#-------------------------------------------------------------------------------
#                   VALORES POR UNIDAD -> VALORES REALES
#-------------------------------------------------------------------------------
    # P[pu]*Sbase=P[MW]
    # Q[pu]*Sbase=Q[MW]
    # V[pu]*Vbase=V[V]
    P=P*Sbase
    Q=Q*Sbase
    V=V*Nominal_kV
#-------------------------------------------------------------------------------
    # Modificamos el vector que recoge los resultados de cada iteracion del
    # nodo 9 para posteriormente visualizar el perfil de cargas diario de
    # este nodo.
    # Además, cambiamos el signo de las potencias obtenidas para graficar
    # como positivas las potencias demandadas, y como potencias negativas las
    # potencias generadas:
    # (+) -> Potencias demandadas
    # (-) -> Potencias generadas
    P_plot[x]=P[8]*(-1)
    Q_plot[x]=Q[8]*(-1)
#-------------------------------------------------------------------------------
    # Modificamos el vector que recoge los resultados de cada iteracion del
    # nodo 9 para posteriormente visualizar el perfil de tensiones diario de
    # este nodo.
    V_plot[x]=V[8]
#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
#                   CONTINUACION CODIGO POWERMETER
#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
    # En estas lineas se convierte la fecha recogida por el PowerMeter a formato
    # 'string' para poder escribirla en el archivo 'registro.txt'.
    #
    # Convert to string
    yr_st=str(year)
    mt_st=str(month)
    dy_st=str(day)
    hr_st=str(hour)
    mn_st=str(minute)
    ml_st=str(miliseconds)
    # Fill with '0s' for uniformity
    while len(yr_st) < 2:
        yr_st='0'+yr_st
    while len(mt_st) < 2:
        mt_st='0'+mt_st
    while len(dy_st) < 2:
        dy_st='0'+dy_st
    while len(hr_st) < 2:
        hr_st='0'+hr_st
    while len(mn_st) < 2:
        mn_st='0'+mn_st
    while len(ml_st) < 5:
        ml_st='0'+ml_st
```

```
630.     f.write(dy_st+'/'+mt_st+'/'+yr_st+' '+hr_st+':'+mn_st+':'+ml_st+'    '+str("%.5f" % Van_f)+'    '+str
    ("%.5f" % Ia_f)+'    '+str("%.5f" % Pa_f)+'    '+str("%.5f" % Qa_f)+'\n')
631.     # f.write escribe en el archivo 'registro.txt' la fecha (hasta los milisegundos),
632.     # V,I,P,Q, de cada lectura del PowerMeter.
633. #-------------------------------------------------------------------------
634. #-------------------------------------------------------------------------
635. #           SI SE QUIEREN MOSTRAR LOS RESULTADOS DE CADA ITERACION
636. #-------------------------------------------------------------------------
637. #-------------------------------------------------------------------------
638.     # Convertimos los ángulos de radianes a grados: [radianes]->[grados]
639.   # Theta_radianes=Theta
640.   # Theta_grados=Theta*180/np.pi
641.   # Phase_Shift=[0,0,-30,-30,-30,-30,-30+30,-30+30,-
    30+30] # Incluimos el índice horario de los trafos
642.   # Theta_grados=Theta_grados-Phase_Shift
643. #-------------------------------------------------------------------------
644.     # Mostramos los resultados en vectores columna: P,Q,V,Theta_radianes,Theta_grados,iteracion
645.   # print ("Potencia activa [P]:",np.reshape(P,(9,1)))
646.   # print ("Potencia reactiva [Q]:",np.reshape(Q,(9,1)))
647.   # print ("Tensiones [V]:",np.reshape(V,(9,1)))
648.   # print ("Angulos de las tensiones en radianes [Theta_radianes]:",np.reshape(Theta_radianes,(9,1)))
649.   # print ("Angulos de las tensiones en grados [Theta_grados]:",np.reshape(Theta_grados,(9,1)))
650.   # print ("Numero de iteraciones:",iteracion)
651.   # print ("")
652. #-------------------------------------------------------------------------
653.     now2 = datetime.now()
654.     delay = total_delay*x+total_delay-(now2-now_ref).total_seconds()
655.     #print ("%.6f" % delay)
656.     time.sleep(delay)
657.     # time.sleep (delay) detiene la ejecucion del codigo durante un tiempo
658.     # equivalente a la variable 'delay'. Esto se hace debido al retraso que
659.     # existe durante el recibimiento de las medidas del PowerMeter.
660. #-------------------------------------------------------------------------
661. f.close()       # Cierra el archivo 'registro.txt'
662. client.close()  # Cierra la conexion con el PowerMeter
663. #-------------------------------------------------------------------------
664. # Cada línea del texto 'registro.txt' está formada por 6 columnas: Fecha-Hora-
665. # V-I-P-Q
666. #
667. # Si no existía 'registro.txt' antes de ejecutar este programa, el archivo tiene
668. # tantas líneas de texto como medidas haya realizado el PowerMeter.
669. #
670. # Si existía previamente 'registro.txt' porque se ejecuto el programa mas veces
671. # con anterioridad, los nuevos datos se almacenan despues de los que ya
672. # existian.
673. #
674. # Con la siguiente instruccion guardamos cada columna de datos en un vector.
675. # Por ejemplo, el vector 'Date' estara formado por todas las fechas de las medidas
676. # del PowerMeter, es decir, este vector incluye la primera columna de todas
677. # las lineas de texto.
678. Date,Time,V,I,P,Q= np.loadtxt('registro.txt',dtype=np.str,unpack=True)
679. # Para asegurarnos de coger los tiempos correctos en caso de que previamente
680. # existiera 'registro.txt':
681. Time=Time[-samples:]
682. #-------------------------------------------------------------------------
683. #-------------------------------------------------------------------------
684. #                       PERFIL DE CARGAS DIARIO
685. #-------------------------------------------------------------------------
686. #-------------------------------------------------------------------------
687. import matplotlib.pyplot as plt
688. from matplotlib import style
689. style.use('ggplot')
690. # Abrimos una grafica que vamos a dividir en tres sub-graficas, una encima
691. # de otra.
```

```
692. # La grafica superior va a representar el perfil de P
693. # La grafica del medio va a representar el perfil de V
694. # La grafica inferior va a representar el perfil de Q
695. # Estas graficas van a compartir el eje X
696. fig=plt.figure()
697. #-------------------------------------------------------------------------
698. # Graficamos el perfil de potencia activa (P y Q) del nodo 9
699. #-------------------------------------------------------------------------
700. # Abrimos la sub-grafica superior para representar el perfil diario de P
701. ax1=plt.subplot2grid((7,1),(0,0),rowspan=2,colspan=1)
702. # Dibujamos P en color verde
703. ax1.plot(Time,P_plot,color='g',linewidth=2)
704. # Añadimos el titulo de la grafica, el nombre del eje Y, y la leyenda.
705. # Ademas, se va a colorear o rellenar el area que forma la grafica con el
706. # eje X
707. ax1.fill_between(Time,P_plot,0,facecolor='seagreen',alpha=0.7)
708. plt.title('Perfil de cargas y tensiones: Nodo 9',color='navy')
709. plt.ylabel('P [MW]',color='darkred')
710. ax1.set_yticks([0,1,2,3])
711. #-------------------------------------------------------------------------
712. # Abrimos la sub-grafica inferior para representar el perfil diario de Q
713. ax3=plt.subplot2grid((7,1),(5,0),rowspan=2,colspan=1,sharex=ax1)
714. # Dibujamos Q en color azul
715. ax3.plot(Time,Q_plot,color='navy',linewidth=2)
716. # Añadimos el titulo de la grafica, el nombre del eje Y, y la leyenda.
717. # Ademas, se va a colorear o rellenar el area que forma la grafica con el
718. # eje X
719. ax3.fill_between(Time,Q_plot,0, facecolor='b', alpha=0.7)
720. plt.ylabel('Q [Mvar]',color='darkred')
721. plt.xlabel('Tiempo',color='darkred')
722. ax3.set_yticks([0,-1.5,-3,-4.5,-6,-7.5])
723. #-------------------------------------------------------------------------
724. # Por temas de espacio al representar las graficas, si samples > 50 los valores
725. # del eje X se giran 90 grados. Si samples <= 50 los valores del eje X se giran
726. # 45 grados en sentido horario.
727. if samples>50:
728.     for label in ax3.xaxis.get_ticklabels():
729.         label.set_rotation(90)
730. else:
731.     for label in ax3.xaxis.get_ticklabels():
732.         label.set_rotation(45)
733. #-------------------------------------------------------------------------
734. #-------------------------------------------------------------------------
735. #                      PERFIL DE TENSIONES DIARIO
736. #-------------------------------------------------------------------------
737. #-------------------------------------------------------------------------
738. # Abrimos la sub-grafica inferior para representar el perfil diario de V
739. ax2=plt.subplot2grid((7,1),(2,0),rowspan=3,colspan=1,sharex=ax1)
740. # Dibujamos V en color rojo
741. ax2.plot(Time,V_plot,color='r',linewidth=2)
742. # Añadimos el titulo de la grafica, el nombre del eje Y, y la leyenda.
743. # Ademas, se va a colorear o rellenar el area que forma la grafica con el
744. # eje X (en realidad va a rellenar el area entre la grafica y el valor 13.7)
745. ax2.fill_between(Time,V_plot,13.7, facecolor='r', alpha=0.7)
746. plt.ylabel('V [kV]',color='darkred')
747. ax2.set_yticks([13.7,13.75,13.8,13.85,13.9])
748. # Impedimos la notacion cientifica en los valores del eje y de las grafica
749. # con la siguiente instruccion:
750. ax2.ticklabel_format(axis='y',useOffset=False, style='plain')
751. #-------------------------------------------------------------------------
752. # Ajustamos las sub-graficas para su correcta visualizacion
753. plt.subplots_adjust(left=0.1, bottom=0.2, right=0.9, top=0.9, wspace=0, hspace=1)
754. # Se hacen NO visibles los ejes X de ax1 y ax2
755. plt.setp(ax1.get_xticklabels(), visible=False)
```

```
plt.setp(ax2.get_xticklabels(), visible=False)
#-------------------------------------------------------------------------------
# Finalmente, mostramos ambas graficas
plt.show()
#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
end=time.time()
elapsed=end-start  # Tiempo transcurrido (en segundos) una vez ejecutado el programa
if elapsed<=60:
    print ("Elapsed time:",'%.4f' % elapsed,"seconds")
elif elapsed>60 and elapsed<=3600:
    elapsed=elapsed/60
    print ("Elapsed time:",'%.4f' % elapsed,"minutes")
elif elapsed>3600 and elapsed <=86400:
    elapsed=elapsed/3600
    print ("Elapsed time:",'%.4f' % elapsed,"hours")
elif elapsed>86400:
    elapsed=elapsed/86400
    print ("Elapsed time:",'%.4f' % elapsed,"days")
#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
#                                   FIN
#-------------------------------------------------------------------------------
#-------------------------------------------------------------------------------
```

```
#-------------------------------------------------------------------------
#                       REPRESENTACION EN TIEMPO REAL
#-------------------------------------------------------------------------
# Modificando el codigo utilizado para el calculo del flujo de cargas de la red
# de 9 nodos se puede llegar a representar graficamente en tiempo real cualquier
# variable de estado que se desee.
# En conreto, se van a representar en tiempo real los valores de P,Q y V del
# nodo 9. Estos valores son los resultados de resolver el flujo de cargas en
# cada instante de tiempo.
# La variable 'interval' es la que indica cada cuandos milisegundos
# se actualiza la informacion, es decir, cada cuanto tiempo se toman medidas
# del PowerMeter para resolver el flujo de cargas.
#-------------------------------------------------------------------------
import numpy as np
#-------------------------------------------------------------------------
#-------------------------------------------------------------------------
#                           BRANCH DATA (CDF [17])
#-------------------------------------------------------------------------
#-------------------------------------------------------------------------
# Branch Type       0 - Transmission line
#                   1 - Fixed tap
#                   2 - Variable tap for voltage control (TCUL, LTC)
#                   3 - Variable tap (turns ratio) for MVAR control
#                   4 - Variable phase angle for MW control (phase shifter)
#-------------------------------------------------------------------------
# Se introducen los datos de todas las lineas de la red
#
# Numero de linea o "branch"
Branch=np.array([1,2,3,4,5,6,7,8])
# Desde la barra "i"
From_Bus=np.array([1,2,3,4,5,6,3,8])
# Hasta la barra "k"
To_Bus=np.array([2,3,4,5,6,7,8,9])
# Tipo de linea de acuerdo con el formado CDF
Branch_Type=np.array([0,2,0,2,0,2,2,0])
# Resistencia [R] de la linea en por unidad
R_pu=np.array([0.000243,0.001667,0.001386,0.008000,0.018933,0.095000,0.006133,0.152174])
# Reactancia [X] de la linea en por unidad
X_pu=np.array([0.002331,0.023889,0.001300,0.080000,0.004433,0.480000,0.056667,0.105860])
# Tap de los transformadores
Tap=np.array([1,1.0125,1,0.9875,1,0.9250,0.9750,1])     # Tap=1 para lineas
#-------------------------------------------------------------------------
#-------------------------------------------------------------------------
#                       MATRIZ DE ADMITANCIAS [Ybarra]
#-------------------------------------------------------------------------
#-------------------------------------------------------------------------
# Se renombran los vectores por comodidad de escritura
R=R_pu
X=X_pu
a=Tap
# Impedancias de las líneas [z]
z=np.zeros((8),dtype=complex) # Creamos un vector fila 'complejo' de ceros de dimension 1x8 [0+0j,0+0j.
..]
z.real=np.array(R) # Parte real de las impedancias (Resistencias [R])
z.imag=np.array(X) # Parte imaginaria de las impedancias (Reactancias [X])
# Admitancias de las líneas [y]
y=1/z
# Número de líneas
nl=len(From_Bus)
# Número de barras
nb=max(max(From_Bus),max(To_Bus))
```

```
#-------------------------------------------------------------------------
# Construcción Matriz Admitancias [Y]
Ybarra=np.zeros((nb,nb),dtype=complex) # Creamos matriz de ceros de dimension (9x9)
# Elementos fuera de la diagonal
for i in range (nl):
    Ybarra[From_Bus[i]-1,To_Bus[i]-1]= Ybarra[From_Bus[i]-1,To_Bus[i]-1]-y[i]/a[i]
    Ybarra[To_Bus[i]-1,From_Bus[i]-1]=Ybarra[From_Bus[i]-1,To_Bus[i]-1]
# Elementos de la diagonal
for i in range (nb):
    for k in range (nl):
        if From_Bus[k]==i+1:
            Ybarra[i,i]=Ybarra[i,i]+(y[k]/a[k])+[y[k]*(1-a[k])/(a[k]**2)]
        elif To_Bus[k]==i+1:
            Ybarra[i,i]=Ybarra[i,i]+(y[k]/a[k])+[y[k]*(a[k]-1)/a[k]]
# Parte real [G:conductancia] e imaginaria [B:susceptancia] de las admitancias
# de las lineas [Y=G+jB]
G=Ybarra.real  # Conductancia [G]
B=Ybarra.imag  # Susceptancia [B]
#-------------------------------------------------------------------------
#-------------------------------------------------------------------------
# Modulos utilizados para la representacion en tiempo real:
import matplotlib.pyplot as plt
import matplotlib.animation as animation
#-------------------------------------------------------------------------
# Abrimos una grafica que vamos a dividir en tres sub-graficas, una encima
# de otra.
# La grafica superior va a representar el perfil de P
# La grafica del medio va a representar el perfil de V
# La grafica inferior va a representar el perfil de Q
# Estas graficas van a compartir el eje X
fig=plt.figure()
from matplotlib import style
style.use('ggplot')
#-------------------------------------------------------------------------
# Abrimos la primera sub-grafica para representar el perfil diario de P
ax1=plt.subplot2grid((7,1),(0,0),rowspan=2,colspan=1)
# Añadimos el titulo de la grafica, el nombre del eje Y, y la leyenda.
plt.title('Perfil de cargas y tensiones: Nodo 9',color='navy')
plt.ylabel('P [MW]',color='darkred')
    # Abrimos la segunda sub-grafica para representar el perfil diario de V
    ax2=plt.subplot2grid((7,1),(2,0),rowspan=3,colspan=1,sharex=ax1)
    plt.ylabel('V [kV]',color='darkred')
    # Abrimos la tercera sub-grafica para representar el perfil diario de Q
    ax3=plt.subplot2grid((7,1),(5,0),rowspan=2,colspan=1,sharex=ax1)
    plt.ylabel('Q [Mvar]',color='darkred')
    plt.xlabel('Tiempo',color='darkred')
    #-------------------------------------------------------------------------
    # Ajustamos las sub-graficas para su correcta visualizacion
    plt.subplots_adjust(left=0.1, bottom=0.2, right=0.9, top=0.9, wspace=0, hspace=2)
    # Se hacen NO visibles los ejes X de ax1 y ax2
    plt.setp(ax1.get_xticklabels(), visible=False)
    plt.setp(ax2.get_xticklabels(), visible=False)
    # Se rotan los valores del eje X 45º en sentido horario
    for label in ax3.xaxis.get_ticklabels():
        label.set_rotation(45)
    #-------------------------------------------------------------------------
    # Creamos listas vacias para ir almacenando en ellas los valores de P,Q,V cada
    # vez que se resuelva el flujo de cargas de la red.
    # Inicialmente estas listas estan formadas por 20 valores, todos cero.
    b=20
    V_plot=[0 for x in range(b)]
    P_plot=[0 for x in range(b)]
    Q_plot=[0 for x in range(b)]
    # [Time_plot] va a almacenar la hora de cada medida realizada por el PowerMeter.
```

```
125.      # Esta lista ira leyendo 'registro.txt' cada vez que se actualice para tomar
126.      # de ese archivo la hora exacta en que fue tomada cada medida.
127.      Time_plot=[0 for x in range(b)]
128.      # Inicializamos x=0 para que una vez ejecutada la funcion 'animate' se
129.      # incremente el valor en una unidad
130.      x=0
131.      #--------------------------------------------------------------------------------
132.      #--------------------------------------------------------------------------------
133.      def animate(i):
134.          # Declaramos las variables 'x','V_plot','P_plot','Q_plot','Time_plot'
135.          # como variables globales, ya que Python por defecto considera las
136.          # variables dentro de las funciones como locales.
137.          # De esta manera conseguimos modificar al final de esta funcion ('animate')
138.          # el valor de estas variables, definidas fuera de esta funcion.
139.          global x,b
140.          global V_plot, P_plot, Q_plot, Time_plot
141.      #--------------------------------------------------------------------------------
142.      #                 CODIGO PARA LECTURA DATOS POWER METER
143.      #--------------------------------------------------------------------------------
144.          # from pymodbus3.client.sync import ModbusTcpClient
145.          # from pymodbus3.constants import Endian
146.          # --codigo--
147.          # f.close()
148.          # client.close()
149.      #--------------------------------------------------------------------------------
150.          from pymodbus3.client.sync import ModbusTcpClient
151.          from pymodbus3.constants import Endian
152.          from pymodbus3.payload import BinaryPayloadDecoder
153.          import time
154.          import threading
155.          from datetime import datetime
156.          from datetime import timedelta
157.          #
158.          def boolList2BinString(lst):
159.              return '0b' + ''.join(['1' if x else '0' for x in lst])
160.          #
161.          client = ModbusTcpClient('80.31.186.204') # IP para conectarse al PowerMeter
162.          #
163.          f = open('registro.txt', 'a')
164.          # Abre un archivo.txt llamado 'registro', en el que se ira registrando cada
165.          # medida del PowerMeter con el siguiente formato: fecha - hora - V[V] - I[A] - P[kW] - Q[kvar
      ]
166.          #
167.          # Reading date and time from smart meter
168.          # Reading registers
169.          year = client.read_holding_registers(1844,1)
170.          m_day = client.read_holding_registers(1845,1)
171.          h_min = client.read_holding_registers(1846,1)
172.          mili = client.read_holding_registers(1847,1)
173.          # Decoding
174.          year_decoder = BinaryPayloadDecoder.from_registers(year.registers, endian=Endian.Big)
175.          m_day_decoder = BinaryPayloadDecoder.from_registers(m_day.registers, endian=Endian.Big)
176.          h_min_decoder = BinaryPayloadDecoder.from_registers(h_min.registers, endian=Endian.Big)
177.          mili_decoder = BinaryPayloadDecoder.from_registers(mili.registers, endian=Endian.Big)
178.          # Decoding year
179.          yr = year_decoder.decode_bits()
180.          yr = year_decoder.decode_bits()
181.          yr_m = yr[6],yr[5],yr[4],yr[3],yr[2],yr[1],yr[0]
182.          # Decoding month and day
183.          mt = m_day_decoder.decode_bits()
184.          dy = m_day_decoder.decode_bits()
185.          mt_m = mt[5],mt[4],mt[3],mt[2],mt[1],mt[0]
186.          dy_m = dy[4],dy[3],dy[2],dy[1],dy[0]
187.          # Decoding hour and minute
```

```
        hr = h_min_decoder.decode_bits()
        hr_m = hr[4],hr[3],hr[2],hr[1],hr[0]
        mn = h_min_decoder.decode_bits()
        mn_m = mn[5],mn[4],mn[3],mn[2],mn[1],mn[0]
        # Decoding miliseconds
        ml = mili_decoder.decode_16bit_uint()
        # Variables are represented as integer values
        year = int(boolList2BinString(yr_m),2)
        month = int(boolList2BinString(mt_m),2)
        day = int(boolList2BinString(dy_m),2)
        hour = int(boolList2BinString(hr_m),2)
        minute = int(boolList2BinString(mn_m),2)
        miliseconds = ml
        # Shown date and time
        # print str(day)+'/'+str(month)+'/'+str(year)+' '+str(hour)+':'+str(minute)+':'+str(miliseconds)
        #
        # Read registers
        # Voltage phase 'a' to neutral (Van)
        Van = client.read_holding_registers(3027, 2)
        # Current phase 'a' (Ia)
        Ia = client.read_holding_registers(2999, 2)
        # Active power phase 'a' (Pa)
        Pa = client.read_holding_registers(3053, 2)
        # Reactive power phase 'a' (Qa)
        Qa = client.read_holding_registers(3061, 2)
        # Decoding (FLOAT32)
        Van_decoder = BinaryPayloadDecoder.from_registers(Van.registers, endian=Endian.Big)
        Ia_decoder = BinaryPayloadDecoder.from_registers(Ia.registers, endian=Endian.Big)
        Pa_decoder = BinaryPayloadDecoder.from_registers(Pa.registers, endian=Endian.Big)
        Qa_decoder = BinaryPayloadDecoder.from_registers(Qa.registers, endian=Endian.Big)
        Van_f = Van_decoder.decode_32bit_float()
        Ia_f = Ia_decoder.decode_32bit_float()
        Pa_f = Pa_decoder.decode_32bit_float()
        Qa_f = Qa_decoder.decode_32bit_float()
        # Showing values [V],[P],[Q]
        print ("Lectura del PowerMeter nº:",x+1)
        print ("V9[V]:"+ str(Van_f)+"   ","P9[kW]:"+str(Pa_f)+"   ","Q9[kvar]:"+str(Qa_f))
        # Mostramos los valores de V,P,Q. Estas medidas son las que asignamos al
        # nodo 9.
    #-------------------------------------------------------------------------------
    #-------------------------------------------------------------------------------
    #                   LECTURA DE POTENCIAS Y ESCALADO
    #-------------------------------------------------------------------------------
    #-------------------------------------------------------------------------------
        # Importamos el archivo 'ADRES_2Sets_1min.mat' desde MATLAB a Python.
        #
        # Este archivo contiene lecturas de P y Q de 5 nodos de otra red, registradas
        # durante cada minuto a lo largo de un dia completo. Es decir, cada nodo tiene
        # registrado 1440 lecturas de P y 1440 lecturas de Q.
        #
        # El objetivo es simular que estos 5 nodos se corresponden con los 5 nodos
        # con carga de nuestra red. De esta forma, las potencias demandadas en cada nodo
        # ira cambiando a lo largo del dia, lo que se ajusta a la situacion real.
        #
        # En realidad, del archivo.mat solo vamos a utilizar las medidas de los
        # 4 primeros nodos. Las lecturas de potencia correspondientes al ultimo nodo
        # son las potencias medidas a traves del PowerMeter.
        #
        # En resumen, 4 nodos se simulan con las potencias del archivo.mat, y un nodo
        # se simula con las potencias medidas con el PowerMeter.
        from scipy.io import loadmat
        potencias=loadmat('ADRES_2sets_1min.mat')
        # En el archivo, las lecturas de cada nodo "i" se llaman P_i_o_av y Q_i_o_av.
```

```
        #
        # Unidades de las lecturas (archivo.mat) -> P [MW] y Q[Mvar]
        # Unidades de las lecturas (PowerMeter) -> P [kW] y Q[kvar]
        #
        # Creamos un vector fila de dimension (1x1440) para cada potencia.
        P_1_o_av=potencias["P_1_o_av"]
        P_2_o_av=potencias["P_2_o_av"]
        P_3_o_av=potencias["P_3_o_av"]
        P_4_o_av=potencias["P_4_o_av"]
        # P_5_o_av=potencias["P_5_o_av"] ->  No lo utilizamos (PowerMeter)
        Q_1_o_av=potencias["Q_1_o_av"]
        Q_2_o_av=potencias["Q_2_o_av"]
        Q_3_o_av=potencias["Q_3_o_av"]
        Q_4_o_av=potencias["Q_4_o_av"]
        # Q_5_o_av=potencias["Q_5_o_av"] ->  No lo utilizamos (PowerMeter)
        #
        # Renombramos las potencias para nuestra red.
        # De esta manera los 4 nodos del archivo 'ADRES_2Sets_1min.mat' serían en
        # nuestra red los nodos 3,5,7,8. El nodo 9 se corresponde con las medidas
        # del PowerMeter.
        P3=P_1_o_av
        P5=P_2_o_av
        P7=P_3_o_av
        P8=P_4_o_av
        P9=Pa_f
        Q3=Q_1_o_av
        Q5=Q_2_o_av
        Q7=Q_3_o_av
        Q8=Q_4_o_av
        Q9=Qa_f
        # Datos de la red -> P[MW] y Q[Mvar]
        P3_dato=84
        P5_dato=34
        P7_dato=4.9
        P8_dato=52
        P9_dato=2.7
        Q3_dato=26
        Q5_dato=12
        Q7_dato=12.6
        Q8_dato=39
        Q9_dato=-3.4
        # Para asignar las lecturas de potencia del archivo.mat y del PowerMeter
        # a los nodos de nuestra red es necesario realizar un escalado de los datos
        #mediante la siguiente operacion:
        # (Lectura de potencia * k = Potencia "escalada" en nuestro nodo)
        #
        # k es la constante de escalado -> ki=Pi_dato/max(Pi) ; ki=Qi_dato/max(Qi)
        #
        # Ejemplo para nodo 3 (potencia activa) -> k3_p=84[MW]/max(P9)[MW]
        # Ejemplo para nodo 3 (potencia reactiva) -> k3_q=26[Mvar]/max(P9)[Mvar]
        #
        # Es decir, se calcula dividiendo el dato de potencia activa o reactiva del nodo
        # de nuestra red entre la maxima potencia activa o reactiva del nodo equivalente
        # de la red de la que estamos utilizando las lecturas.
        # Esta operacion es valida para los nodos 3,5,7,8.
        #
        # Como el nodo 9 toma las potencias del PowerMeter, el escalado se hace tomando
        # como la maxima potencia activa 18[W], por ser el maximo consumo medido
        # por el PowerMeter cuando tiene una pequeña carga conectada.
        # De esta forma -> k9=(1/1000)*c
        # (Se divide por (1/1000) porque el PowerMeter da la medida en kW, y queremos
        # que este en MW para nuestra red)
        # siendo c=(2.7[MW]*1000000)/18[W]
        # Asi conseguimos escalar las potencias medidas con el PowerMeter (en kW y kvar)
```

```
        # y adecuarlas a nuestra red (en MW y Mvar).
        k3_p=P3_dato/max(P3[0,])
        k5_p=P5_dato/max(P5[0,])
        k7_p=P7_dato/max(P7[0,])
        k8_p=P8_dato/max(P8[0,])
        c=(P9_dato*1e6)/18
        k9_p=(1/1e3)*c
        k3_q=Q3_dato/max(Q3[0,])
        k5_q=Q5_dato/max(Q5[0,])
        k7_q=Q7_dato/max(Q7[0,])
        k8_q=Q8_dato/max(Q8[0,])
        # Utilizamos la misma constante de escalado para Q que para P en el nodo 9.
        k9_q=k9_p
    #---------------------------------------------------------------------------
    #---------------------------------------------------------------------------
    #                               BUS DATA (CDF)
    #---------------------------------------------------------------------------
    #---------------------------------------------------------------------------
        #  Bus Type
        #                   0 - Unregulated (load, PQ)
        #                   1 - Hold MVAR generation within voltage limits, (PQ)
        #                   2 - Hold voltage within VAR limits (gen, PV)
        #                   3 - Hold voltage and angle (swing, V-Theta) (must always
        #                       have one)
    #---------------------------------------------------------------------------
        # Se introducen los datos de todas las barras de la red
        #
        # Numero de barra
        Bus_Number=np.array([1,2,3,4,5,6,7,8,9])
        # Tipo de barra de acuerdo con el formado CDF
        Bus_Type=np.array([3,0,0,0,0,0,0,0,0])
        # Tension nominal de las barras
        Nominal_kV=np.array([220,220,132,132,30,30,13.8,13.8,13.8])
        # P_demandada [MW]
        Pd=np.array([0,0,P3[0,x]*k3_p,0,P5[0,x]*k5_p,0,P7[0,x]*k7_p,P8[0,x]*k8_p,P9*k9_p])
        # P_generada [MW]
        Pg=np.array([0,0,0,0,0,0,0,0,0])
        # Q_demandada [Mvar]
        Qd=np.array([0,0,Q3[0,x]*k3_q,0,Q5[0,x]*k5_q,0,Q7[0,x]*k7_q,Q8[0,x]*k8_q,Q9*k9_q])
        # Q_generada [Mvar]
        Qg=np.array([0,0,0,0,0,0,0,0,0])
        # Tensión [pu]
        V=np.array([1,1,1,1,1,1,1,1,1],dtype=float)
        # Ángulo de las tensiones [radianes]
        Theta=np.array([0,0,0,0,0,0,0,0,0],dtype=float)
        # Se fija en la barra slack V=1 [pu] & Theta=0
        # Resto de variables desconocidas: V=1 & Theta=0 (Arranque plano) ; P=Q=0
    #---------------------------------------------------------------------------
        # Potencia Base S=100 [MVA]
        Sbase=100
        # Pg: Potencia activa generada [pu]
        Pg=Pg/Sbase
        # Qg: Potencia reactiva generada [pu]
        Qg=Qg/Sbase
        # Pd: Potencia activa demandada [pu]
        Pd=Pd/Sbase
        # Qd: Potencia reactiva demandada [pu]
        Qd=Qd/Sbase
        # Pi=Pgi-Pdi: Potencia activa inyectada a la red desde la barra i
        P=Pg-Pd
        # Qi=Qgi-Qdi: Potencia reactiva inyectada a la red desde la barra i
        Q=Qg-Qd
        # P especificada
        Pesp=P
```

```
        # Q especificada
        Qesp=Q
    #-------------------------------------------------------------------------
        # Cálculo del número de barras PV y PQ en el sistema
        # Solo hay barras PQ
    #-------------------------------------------------------------------------
        # PV y PQ: Devuelve la posición de las barras tipo 3 (Slack) y tipo 0 (PQ)
        # en el vector "Bus_Type". Es decir, nos dice que barras son PV y PQ
        PV=np.where(Bus_Type==3)  # Como no hay barras PV, buscamos solo barras tipo 3
        PQ=np.where(Bus_Type==0)
        # np.where devuelve un array de dimension 1x2. El primer elemento [0] del array
        # es un array con los indices o posiciones de "3" o "0" en "Bus_Type".
        # El segundo elemento [1] del array devuelto por np.where indica el formato del
        # array del primer elemento [0].
        #
        # Número de barras PV (incluyendo la barra slack)
        nPV=len(PV[0])
        # Número de barras PQ
        nPQ=len(PQ[0])
    #-------------------------------------------------------------------------
        # Cálculo del vector incógnitas X=[Theta | V] -> Ángulos y tensiones desconocidas
        # en todas las barras menos la barra slack: [Theta2...Theta9 | V2...V9]
    #-------------------------------------------------------------------------
        # Vx: Tensiones incógnitas (En todas las barras PQ) -> [V2...V9]
        k=0
        Vx=np.zeros((nPQ))
        for i in range (nb):
            if Bus_Type[i]==0:
                Vx[k]=V[i]
                k=k+1
        # Thetax: Ángulos incógnitas (Todas las barras menos slack) -> [Theta2...Theta9]
        Thetax=Theta[1:]
        # X: Vector Incógnitas -> [Theta2...Theta9 | V2...V9]
        X=np.concatenate((Thetax,Vx))
    #-------------------------------------------------------------------------
    #-------------------------------------------------------------------------
    #         Ecuaciones de potencia y cálculo mediante Newton-Raphson
    #-------------------------------------------------------------------------
    #-------------------------------------------------------------------------
        # No se plantean las ecuaciones de potencia en la barra slack
        # Se plantean las potencias [P] en todas las barras PV y PQ -> [P]=[P2...P9]
        # Se plantean las potencias [Q] en las barras PQ -> [Q]=[Q2...Q9]
    #-------------------------------------------------------------------------
        #      Ecuaciones del flujo de cargas en la barra "i" en forma polar:
        # P=V[i]*V[k]*(G[i,k]*cos(Theta[i]-Theta[k])+B[i,k]*sin(Theta[i]-Theta[k]))
        # Q=V[i]*V[k]*(G[i,k]*sin(Theta[i]-Theta[k])-B[i,k]*cos(Theta[i]-Theta[k]))
        #                       Para k in range (nb)
        #                   P= Potencia activa estimada
        #                   Q= Potencia reactiva estimada
    #-------------------------------------------------------------------------
        # Se va a realizar el método de Newton-Raphson para una tolerancia de 10-5.
        # Es decir, hasta que todos los términos de la matriz R=[dP | dQ] sean
        # menores que 10-5 no se dentendrá el cálculo iterativo.
    #-------------------------------------------------------------------------
        # [dP]=[Pesp-Pe]: residuos de potencia activa -> [dP]=[dP2...dP9]
        # [dQ]=[Qesp-Qe]: residuos de potencia reactiva -> [dQ]=[dQ2...dQ9]
        # [R]=[dP2...dP9 | dQ2...dQ9]
    #-------------------------------------------------------------------------
        # Se inicia la tolerancia para un valor cualquiera mayor que 10-5.
        # Se inicia el nº de iteración en 1. Tras cada iteración este valor se irá
        # incrementando en 1 hasta que termine el cálculo iterativo. Al final,
        # se conocerán el nº de iteraciones que se realizaron.
        tol=1
        iteracion=0
```

```
    #-------------------------------------------------------------------------------
    #-------------------------------------------------------------------------------
    #    Comienza el bucle para el cálculo iterativo mediante Newton-Rapson
    #-------------------------------------------------------------------------------
    #-------------------------------------------------------------------------------
        while tol>1e-5:
        # Potencias activas estimadas (P1...P9)
            P=np.zeros((nb))
            for i in range (nb):
                for k in range (nb):
                    P[i]=P[i]+V[i]*V[k]*(G[i,k]*np.cos(Theta[i]-Theta[k])+B[i,k]*np.sin(Theta[i]-
Theta[k]))
        # Potencias reactivas estimada (Q1...Q9)
            Q=np.zeros(nb)
            for i in range (nb):
                for k in range (nb):
                    Q[i]=Q[i]+V[i]*V[k]*(G[i,k]*np.sin(Theta[i]-Theta[k])-B[i,k]*np.cos(Theta[i]-
Theta[k]))
    #-------------------------------------------------------------------------------
        # Residuos de potencias de todas las barras [R]=[dP1..dP9 | dQ1..dQ9]
            dPbarras=Pesp-P
            dQbarras=Qesp-Q
        # Selección de los residuos de potencias que nos interesan. Se van a
        # utilizar las potencias P y Q conocidas -> P2...P9 y Q2...Q9
        # No se incluye la barra slack
    #-------------------------------------------------------------------------------
        # Residuos [dP]
            dP=dPbarras[1:]
        # Se utilizan todas las P conocidas, esto es, de las barras PQ
    #-------------------------------------------------------------------------------
        # Residuos [dQ]
            k=0
            dQ=np.zeros((nPQ))
            for i in range (nb):
                if Bus_Type[i]==0:
                    dQ[k]=dQbarras[i]
                    k=k+1
        # Se utilizan todas las Q conocidas, esto es, de las barras PQ
    #-------------------------------------------------------------------------------
        # Residuos de Potencia [R]
            R=np.concatenate((dP,dQ))
    #-------------------------------------------------------------------------------
        # Cálculo del Jacobiano [J]
    #-------------------------------------------------------------------------------
        # Terminos del Jacobiano
        # [J]=[H N;M L]
        #
        # H[i,k]=dP[i]/dTheta[k]          N[i,k]=V[k]*dP[i]/dV[k]
        # M[i,k]=dQ[i]/dTheta[k]          L[i,k]=V[k]*dQ[i]/dV[k]
        #
        # Para i=k
        # H[i,i]=-Q[i]-B[i,i]*V[i]**2     N[i,i]=P[i]+G[i,i]*V[i]**2
        # L[i,i]=Q[i]-B[i,i]*V[i]**2      M[i,i]=P[i]-G[i,i]*V[i]**2
        #
        # Para i distinto k
        # H[i,k]=L[i,k]=V[i]*V[k]*(G[i,k]*sin(Theta[i]-Theta[k])-B[i,k]*cos(Theta[i]-Theta[k]))
        # N[i,k]=-M[i,k]=V[i]*V[k]*(G[i,k]*cos(Theta[i]-Theta[k])+B[i,k]*sin(Theta[i]-Theta[k]))
    #-------------------------------------------------------------------------------
        # Cálculo de [H]
            H=np.zeros((nb-1,nb-1))
            for i in range (nb-1):
                m=i+1
                for k in range (nb-1):
                    n=k+1
```

```
                    if n==m:
                        H[i,k]=-Q[m]-B[m,m]*(V[m])**2
                    else:
                        H[i,k]=V[m]*V[n]*(G[m,n]*np.sin(Theta[m]-Theta[n])-B[m,n]*np.cos(Theta[m]-
Theta[n]))
    #-------------------------------------------------------------------------------
        # Cálculo de [N]
            N=np.zeros((nb-1,nPQ))
            for i in range (nb-1):
                m=i+1
                for k in range (nPQ):
                    n=PQ[0][k]
                    if n==m:
                        N[i,k]=P[m]+G[m,m]*(V[m])**2
                    else:
                        N[i,k]=V[m]*V[n]*(G[m,n]*np.cos(Theta[m]-Theta[n])+B[m,n]*np.sin(Theta[m]-
Theta[n]))
    #-------------------------------------------------------------------------------
        # Cálculo de [M]
            M=np.zeros((nPQ,nb-1))
            for i in range (nPQ):
                m=PQ[0][i]
                for k in range (nb-1):
                    n=k+1
                    if n==m:
                        M[i,k]=P[m]-G[m,m]*(V[m])**2
                    else:
                        M[i,k]=-V[m]*V[n]*(G[m,n]*np.cos(Theta[m]-Theta[n])+B[m,n]*np.sin(Theta[m]-
Theta[n]))
    #-------------------------------------------------------------------------------
        # Cálculo de [L]
            L=np.zeros((nPQ,nPQ))
            for i in range (nPQ):
                m=PQ[0][i]
                for k in range (nPQ):
                    n=PQ[0][k]
                    if n==m:
                        L[i,k]=Q[m]-B[m,m]*(V[m])**2
                    else:
                        L[i,k]=V[m]*V[n]*(G[m,n]*np.sin(Theta[m]-Theta[n])-B[m,n]*np.cos(Theta[m]-
Theta[n]))
    #-------------------------------------------------------------------------------
       # Jacobiano [J]
            # J=[H,N; M,L]
            J1=np.hstack([H,N]) # Junta horizontalmente los 'arrays' [H] y [N]. Dimension [J1]: 8x16

            J2=np.hstack([M,L]) # Junta horizontalmente los 'arrays' [M] y [L]. Dimension [J2]: 8x16

            J=np.vstack([J1,J2]) # Junta verticalmente los 'arrays' [J1] y [J2]. Dimeension [J]: 16x1
6
    #-------------------------------------------------------------------------------
        # Cálculo de [dX]=[dTheta | dV/V] resolviendo el sistema: [R]=[J]*[dX]
        # [dX] es el Vector de Errores -> [dX]=inv(J)*[R]
            J_inverse=np.linalg.inv(J)
            dX=np.dot(J_inverse,R)
    #-------------------------------------------------------------------------------
        # Como [dX]=[dTheta | dV/V], vamos a obtener [dX]=[dTheta | dV]
        # multiplicando [dV/V]*[V], donde [V] son las tensiones incógnitas
            for i in range (nb-1,len(dX)):
                dX[i]=dX[i]*X[i]
    #-------------------------------------------------------------------------------
        # El vector [dX] se puede dividir en [dTheta] y [dV] ya que [dX]=[dTheta | dV]
            dTheta=dX[0:nb-1]
            dV=dX[nb-1:]
```

```
    #---------------------------------------------------------------------------
        # Actualizar las variables [Theta] y [V]
    #---------------------------------------------------------------------------
        # [Theta]nueva=[Theta]antigua+[dTheta]
            Theta[1:nb]=Theta[1:nb]+dTheta
        # No se modifica el ángulo de la barra slack que está fijado en 0º
    #---------------------------------------------------------------------------
        # [V]nueva=[V]antigua+[dV]
            k=1
            for i in range (nb-1):
                if Bus_Type[k]==0:
                    V[k]=V[k]+dV[i]
                    k=k+1
        # No se modifica el modulo de la tension de la barra slack que está fijado en 1 [pu]
    #---------------------------------------------------------------------------
    #               COMPROBAR TOLERANCIA Y SI NO REPETIR BUCLE
    #---------------------------------------------------------------------------
            tol=max(abs(R))
            iteracion=iteracion+1
    #---------------------------------------------------------------------------
    #                  VALORES POR UNIDAD -> VALORES REALES
    #---------------------------------------------------------------------------
        # P[pu]*Sbase=P[MW]
        # Q[pu]*Sbase=Q[MW]
        # V[pu]*Vbase=V[V]
        P=P*Sbase
        Q=Q*Sbase
        V=V*Nominal_kV
    #---------------------------------------------------------------------------
        # Modificamos el vector que recoge los resultados del flujo de cargas en el
        # nodo 9 para posteriormente visualizar el perfil diario de cargas y tensiones
        # de este nodo.
        # Además, cambiamos el signo de las potencias obtenidas para graficar
        # como positivas las potencias demandadas, y como potencias negativas las
        # potencias generadas:
        # (+) -> Potencias demandadas
        # (-) -> Potencias generadas
        V_plot.append(float(V[8]))
        P_plot.append(float(P[8]*-1))
        Q_plot.append(float(Q[8]*-1))
    #---------------------------------------------------------------------------
    #---------------------------------------------------------------------------
    #                   CONTINUACION CODIGO POWERMETER
    #---------------------------------------------------------------------------
    #---------------------------------------------------------------------------
        # En estas lineas se convierte la fecha recogida por el PowerMeter a formato
        # 'string' para poder escribirla en el archivo 'registro.txt'.
        #
        # Writing to the database
        # Convert to string
        yr_st=str(year)
        mt_st=str(month)
        dy_st=str(day)
        hr_st=str(hour)
        mn_st=str(minute)
        ml_st=str(miliseconds)
    #     Fill with '0s' for uniformity
        while len(yr_st) < 2:
            yr_st='0'+yr_st
        while len(mt_st) < 2:
            mt_st='0'+mt_st
        while len(dy_st) < 2:
            dy_st='0'+dy_st
        while len(hr_st) < 2:
```

```
626.             hr_st='0'+hr_st
627.         while len(mn_st) < 2:
628.             mn_st='0'+mn_st
629.         while len(ml_st) < 5:
630.             ml_st='0'+ml_st
631.         f.write(dy_st+'/'+mt_st+'/'+yr_st+' '+hr_st+':'+mn_st+':'+ml_st+'    '+str("%.5f" % Van_f)+'
    '+str("%.5f" % Ia_f)+'    '+str("%.5f" % Pa_f)+'    '+str("%.5f" % Qa_f)+'\n')
632.         # f.write escribe en el archivo 'registro.txt' la fecha (hasta los milisegundos),
633.         # V,I,P,Q, de cada lectura del PowerMeter.
634.     #-------------------------------------------------------------------------------
635.         f.close()       # Cierra el archivo 'registro.txt'
636.         client.close()  # Cierra la conexion con el PowerMeter
637.     #-------------------------------------------------------------------------------
638.         Real_time=hr_st+':'+mn_st+':'+ml_st
639.         Time_plot.append(Real_time)
640.     #-------------------------------------------------------------------------------
641.         # Representamos las grafias de P,Q,V. Cada vez que se resuelva el flujo
642.         # de cargas (esto sera cada cierto tiempo impuesto por la variable 'interval')
643.         # se iran añadiendo los resultados a la graficas.
644.         ax1.plot(Time_plot[len(Time_plot)-b:len(Time_plot)+1],P_plot[-b:],'g',linewidth=2)
645.         ax2.plot(Time_plot[len(Time_plot)-b:len(Time_plot)+1],V_plot[-b:],'darkred',linewidth=2)
646.         ax3.plot(Time_plot[len(Time_plot)-b:len(Time_plot)+1],Q_plot[-b:],'navy',linewidth=2)
647.         # Se colorea el area comprendida entre las graficas y el eje X
648.         ax1.fill_between(Time_plot,P_plot,0,facecolor='seagreen',alpha=0.5)
649.         ax2.fill_between(Time_plot,V_plot,0,facecolor='r',alpha=0.5)
650.         ax3.fill_between(Time_plot,Q_plot,0,facecolor='b',alpha=0.5)
651.         # Actualizamos los limites de los ejes X e Y
652.         ax1.axis([Time_plot[len(Time_plot)-b],Time_plot[len(Time_plot)-1],0,max(P_plot[-b:])+1])
653.         ax2.axis([Time_plot[len(Time_plot)-b],Time_plot[len(Time_plot)-1],13.7,max(V_plot[-
    b:])+0.1])
654.         # Impedimos la notacion cientifica en los valores del eje y de las grafica
655.         # con la siguiente instruccion:
656.         ax2.ticklabel_format(axis='y',useOffset=False, style='plain')
657.         ax3.axis([Time_plot[len(Time_plot)-b],Time_plot[len(Time_plot)-1],min(Q_plot[-b:])-1,0])
658.     #-------------------------------------------------------------------------------
659.         # Incrementamos el valor de la variable 'x' para la proxima vez
660.         # que se ejecute la funcion 'animate'.
661.         x=x+1
662.     #-------------------------------------------------------------------------------
663.     #-------------------------------------------------------------------------------
664.     # Con la siguiente instruccion conseguimos que cada intervalo de tiempo
665.     # 'interval', medido en milisegundos, se ejecute la funcion 'animate' y añada
666.     # los resultados obtenidos en esta funcion (P,Q,V del nodo 9) en la grafica
667.     # 'fig', compuesta por las tres subgraficas para P,Q y V.
668.     # Es decir, cada 3 segundos se ejecutara la funcion 'animate' y, por tanto, se
669.     # resolvera el flujo de cargas. Los resultados se añadiran a la grafica 'fig' y
670.     # asi continuamente hasta que se cierre la ventana de las graficas.
671.     ani=animation.FuncAnimation(fig,animate,interval=3000)
672.     plt.show()
673.     #-------------------------------------------------------------------------------
674.     #-------------------------------------------------------------------------------
675.     #                                 FIN
676.     #-------------------------------------------------------------------------------
677.     #-------------------------------------------------------------------------------
```