Universidad de la República
Facultad de Ingeniería

ILFUS

Identificación y Localización de Faltas Usando Sincrofasores

Memoria de proyecto presentada a la Facultad de Ingeniería de la Universidad de la República por

Sandino Silva, Agustin Olivera, Felipe Ohaco

en cumplimiento parcial de los requerimientos para la obtención del título de Ingeniero Electricista.

Tutor

Celia Sena ..... Universidad de la República

Tribunal

José Munsch ..... Universidad de la República
Agustín Fraschini ..... Universidad de la República
Ignacio Afonso ..... Universidad de la República

Montevideo
martes 23 enero, 2018

ILFUS
Identificación y Localización de Faltas Usando Sincrofasores, Sandino Silva, Agustin Olivera, Felipe Ohaco.

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 133 páginas.
Compilada el martes 23 enero, 2018.
http://iie.fing.edu.uy/

Prefacio

La ubicación rápida y exacta de la falla en una línea de transmisión es la clave para una mejora en el rendimiento y reducción de los costos de operación y mantenimiento de la misma. Por estas razones se hace necesaria la implementación de una aplicación rápida y eficiente para la localización de las fallas, la cual va a depender de la información obtenida del sistema de protección instalado en la línea. Actualmente, estos sistemas de protección permiten medir fasores de tensión y corriente en forma sincronizada teniendo como referencia la hora UTC (Universal Time Coordinated), los cuales son llamados sincrofasores. Los datos de los sincrofasores son recogidos en una base de datos centralizada o concentrador de sincrofasores, llamados PDC (Phasor Data Concentrators).

El proyecto propuesto trata del desarrollo de una aplicación para identificar y localizar fallas en líneas de transmisión utilizando información de los sincrofasores recogidas en la base de datos. Se deberán identificar los sincrofasores que detectan la falla, y con esa información identificar el tipo de falla y la distancia a la misma desde uno de los extremos.

Puede ser que en algunos casos no se obtenga la información de ambos extremos de la línea, por lo cual se deberán estudiar dos casos particulares en cuanto a la obtención de datos, midiendo en un solo extremo de la línea o en ambos. Esto se logra mediante la obtención de un grupo de sincrofasores de uno o ambos extremos de la línea en el momento de la falla. Dependiendo de cada caso se deberá procesar la información utilizando distintos algoritmos.

Esta página ha sido intencionalmente dejada en blanco.

Tabla de contenidos

Tabla de contenidos

Capítulo 1

Introducción

1.1. Motivación

En todo sistema eléctrico de potencia, los elementos que lo componen están expuestos a fallas. Especialmente las líneas de transmisión están propensas a cortocircuitos debido a la exposición de las mismas a condiciones climáticas así como también debido a sus ubicaciones geográficas.

Cuando ocurre una falla en una línea de transmisión, los sistemas de protección deben actuar de forma rápida para despejar la misma y evitar daños severos a los equipos.

Por otra parte, los algoritmos de localización deben determinar de forma precisa el punto donde ocurrió la falla. Ésto es de suma importancia ya que de ello depende el tiempo de reparación por parte del personal de mantenimiento.

Contar con un dispositivo que permita localizar el lugar de la falla en una línea de transmisión de manera confiable, permitirá restablecer la línea en el menor tiempo posible, reduciendo el tiempo en el que la misma se encuentre fuera de servicio.

Actualmente, UTE cuenta con equipos de protección y medida sincronizados instalados en varios puntos de la red. Estos equipos están relevando información en tiempo real y almacenando la misma, aunque no se utiliza esa información con el fin de identificar y localizar fallas.

1.2. Definición del problema

La precisión en la localización de fallas en líneas de transmisión por parte de los sistemas de protección actuales no es la deseada. Por lo tanto no es óptimo el tiempo de llegada al punto de la falla del personal de mantenimiento.

1.3. Objetivos del proyecto

1.3.1. Objetivos generales

Desarrollar una aplicación que identifique y localice fallas en líneas de transmisión utilizando información de sincrofasores recogidas de una base de datos. Esta herramienta será capaz de diferenciar los distintos tipos de fallas que ocurran en las líneas y determinar la distancia a la que ocurren las fallas medidas desde un extremo de la línea. Deberá además, cuando se cuente con medidas desde ambos extremos de la línea, determinar los parámetros de la misma.

1.3.2. Objetivos específicos

1. Estudiar el modelo de una línea de transmisión como cuadripolo pasivo e implementar un algoritmo para determinar sus parámetros.

2. Estudiar diferentes algoritmos de identificación y localización de fallas. En particular, algoritmos para identificar y localizar fallas utilizando datos de un extremo o dos extremos de la línea.

3. La identificación y localización de las fallas se deberá lograr de forma precisa, con un error relativo a la medida menor a 10 %.

4. Implementar los algoritmos estudiados mediante una aplicación y desarrollar un plan de prueba para la misma, utilizando archivos extraídos de la base de datos de UTE, de fallas reales ocurridas en las líneas de transmisión.

5. Crear el manual de usuario de la aplicación. Será la guía que ayudará al usuario a entender el funcionamiento de la aplicación.

Capítulo 2

Introducción a los sincrofasores

En este capítulo se presentarán las definiciones de sincrofasor y de los diferentes parámetros que permiten realizar una evaluación sobre las medidas obtenidas. Se definirán los equipos de medición y almacenamiento de los mismos y se explicará cual es la función que cumple cada uno y la forma en que se vinculan. Al final, se mencionará el conjunto de normas que aplican para los sincrofasores.

2.1. Definición de Fasor [2]

Es la representación fasorial de una señal sinusoidal, usada para el análisis de los sistemas eléctricos. La forma de la onda sinusoidal se define como:

$$x(t) = X_m.cos(wt + \phi) \tag{2.1}$$

Generalmente se representa como se muestra a continuación:

$$X = (X_m/\sqrt{2})e^{j\phi} = (X_m/\sqrt{2})(cos\phi + jsin\phi) = X_r + jX_i \tag{2.2}$$

donde la magnitud $X_m/\sqrt{2}$ es el valor eficaz o rms de la forma de onda, y los subíndices r e i significan la parte real e imaginaria del complejo en coordenadas rectangulares. El valor de ϕ depende de la escala de tiempos, particularmente desde donde se empezó a medir ($t = 0$). Es importante notar que el fasor es definido por la frecuencia angular w y para compararlo con otros fasores debe estar con la misma frecuencia y escala de tiempo.

2.2. Definición de sincrofasor [2]

El sincrofasor que representa la señal $x(t)$ en (2.1) es el valor X en la ecuación (2.2) donde ϕ es el ángulo de fase instantáneo relativo de la función coseno a la frecuencia nominal del sistema sincronizada con UTC (Universal Time Code). Bajo esta definición, ϕ es el offset de la función coseno a la frecuencia nominal del sistema sincronizado con UTC. La figura 2.1 muestra la relación entre la fase del ángulo y el tiempo UTC.

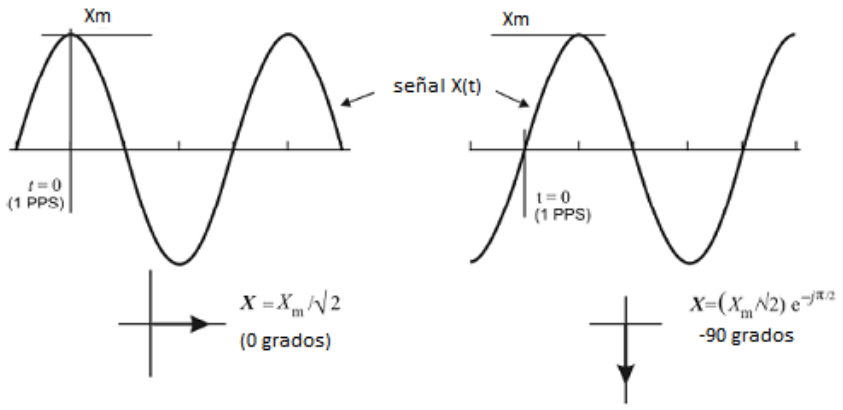


Figura 2.1: Relación entre la fase del angulo y el tiempo UTC

La sinusoide es

$$x(t) = X_m.cos(w_0 t + \phi) = X_m.cos(2\pi f_0 t + \phi) \tag{2.3}$$

donde f_0 es la frecuencia angular nominal del sistema (50 Hz o 60 Hz). En el caso general donde la amplitud y la frecuencia de la sinusoide son una función del tiempo $X_m(t)$ y $f(t)$, podemos definir la función $g = f - f_0$ donde f_0 es la frecuencia angular nominal y g es la diferencia entre la frecuencia actual y la nominal (tal que g será una función del tiempo), entonces:

$$\begin{aligned} x(t) &= X_m(t)cos(2\pi \int f dt + \phi) \\ &= X_m(t)cos(2\pi \int (f_0 + g)dt + \phi) \\ &= X_m(t)cos(2\pi f_0 t + (2\pi \int g dt + \phi)) \end{aligned} \tag{2.4}$$

entonces la representación del sincrofasor resulta:

$$X(t) = (X_m(t)/\sqrt{2})e^{j(2\pi \int g dt + \phi)} \tag{2.5}$$

En el caso especial en que $X_m(t) = X_m$ y $g = f - fo$ son constantes, entonces $\int g(t)dt = \int \Delta f dt = \Delta f t$ y por lo tanto la ecuación para el sincrofasor resulta:

$$X(t) = (X_m/\sqrt{2})e^{j(2\pi \Delta f t + \phi)} \tag{2.6}$$

donde el módulo del sincrofasor permanece constante y rota uniformemente a una velocidad angular $2\pi\Delta f$. Este concepto se muestra en la figura 2.2. Consideremos una señal sinusoidal con frecuencia no nominal del sistema, muestreada en intervalos $0, T_0, 2T_0, 3T_0 ... nT_0$ donde $T_0 = 1/f_0$ (el período del sistema eléctrico) y las correspondientes representaciones del fasor $X_0, X_1, X_2 ... X_n ....$ Si la frecuencia de la señal sinusoidal cumple que $f \neq f_0$ y $f < 2f_0$, el fasor en estudio tendrá una magnitud constante, pero la fase de éste cambiará uniformemente en un rango de

$2\pi(f - f_0)T_0$, como se ve en la figura 2.2. Si estos valores son reportados en el tiempo, la fase aumentará hasta alcanzar los 180 grados, ahí cambiará a -180 y volverá a incrementarse hasta los 180 grados, como se muestra en la figura 2.3 (comúnmente los sincrofasores se registran en ángulos desde -180 a 180).

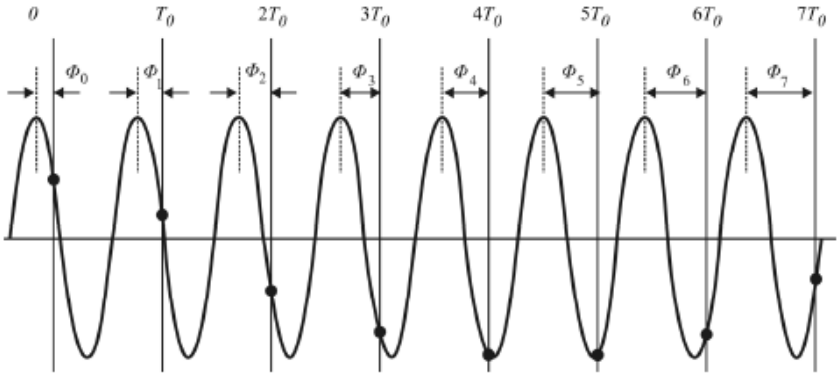


Figura 2.2: Sinusoide con $f > f_0$ observada cada T_0 segundos,que se aparta de la fase un ϕ uniforme, dependiendo de la diferencia entre $f - f_0$

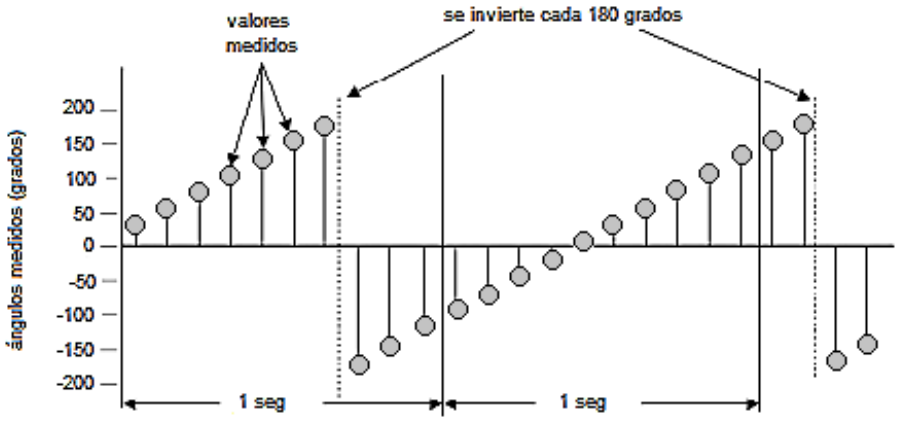


Figura 2.3: La fase varia entre 180 y -180 grados

Todas las mediciones son efectuadas en una base de tiempo común y relativas a la misma frecuencia, por lo tanto las mediciones de las fases son directamente comparables. Las diferencias en la frecuencia actual son incluidas en la estimación de la fase. El sincrofasor estimado también contiene los efectos y contribuciones de las otras señales, tales como las oscilaciones y cambios en la frecuencia local. Los sincrofasores son funciones del tiempo y cambiaran de un valor al siguiente a menos que la señal sea una sinusoide pura a la frecuencia nominal de la red.

2.2.1. Sincronización del tiempo de medición

Los PMU (Phasor Measurement Unit) deberán ser capaces de recibir el tiempo desde una fuente precisa y confiable, como el GPS (Global Positioning System). Este último, puede proveer un tiempo que se puede referenciar al UTC con suficiente precisión como para mantener el TVE (Total Vector Error), el FE (Frequency Error) y el ROCOF (Rate Of Change Of Frequency) en límites aceptables.

Todas las mediciones serán sincronizadas con el tiempo UTC con la precisión necesaria para cumplir con los requerimientos del estándar IEEE Std C37.118.1-2011.

Para cada medida de PMU se deberá asignar una etiqueta de tiempo que incluya el tiempo y la calidad de medida del mismo.

2.2.2. Estimación del sincrofasor

La PMU deberá calcular y ser capaz de reportar la estimación del sincrofasor como se define y describe en la clausula 4 de IEEE Std C37.118.1-2011. La estimación incluirá una sola fase o secuencia positiva o ambas. Deberá preverse la selección por el usuario de los valores medidos. En dicha clausula están previstos los tiempos reportados, la precisión de la medida, y los criterios de evaluación. Notar que las medidas son en realidad estimaciones de un cierto valor.

2.2.3. Estimación de la Frecuencia y ROCOF

El PMU deberá calcular y ser capaz de reportar la frecuencia y el ROCOF. Para esta medición el estándar define como se implementará. Dada la señal sinusoidal (2.7).

$$x(t) = X_m cos[\psi(t)] \tag{2.7}$$

la frecuencia es definida como:

$$f(t) = \frac{1}{2\pi}\frac{d\psi(t)}{dt} \tag{2.8}$$

el ROCOF se define como:

$$ROCOF(t) = \frac{df(t)}{dt} \tag{2.9}$$

Los sincrofasores se calculan siempre en relación con la frecuencia nominal del sistema f_0. Si el argumento del coseno es representado como:

$$\psi(t) = w_0 t + \phi(t) = 2\pi f_0 t + \phi(t) = 2\pi[f_0 t + \phi(t)/2\pi] \tag{2.10}$$

entonces la fórmula para la frecuencia nos queda:

$$f(t) = f_0 + d[\phi(t)/2\pi]/dt = f_0 + \Delta f(t) \tag{2.11}$$

donde $\Delta f(t)$ es la desviación de la frecuencia respecto a la nominal.

$$ROCOF(t) = d^2[\phi(t)/2\pi]/dt^2 = d(\Delta f(t))/dt \tag{2.12}$$

La frecuencia en las mediciones del fasor puede ser reportada como la frecuencia real $f(t)$ o la desviación de la frecuencia respecto a la nominal, $\Delta f(t)$. En condiciones de estado estable, $\Delta f(t)$ puede ser representado como un número escalar Δf.

2.3. Evaluación de las medidas [2]

2.3.1. Evaluación en la medida del sincrofasor

Los valores teóricos de una representación de sincrofasores de una sinusoide, obtenidos desde un PMU pueden incluir diferencias tanto en amplitudes como en fase. Aunque pueden especificarse por separado, las diferencias en amplitud y fase son consideradas juntas en el estándar respecto a calidad, se le llama total vector error (TVE). TVE es una expresión de la diferencia entre una muestra "perfecta"de un sincrofasor teórico y un estimado dado por la unidad bajo análisis en el mismo instante de tiempo. El valor es normalizado y expresado por unidad de fasor teórico. Se define TVE como:

$$TVE(n) = \sqrt{\frac{(\hat{X}_r(n) - X_r(n))^2 + (\hat{X}_i(n) - X_i(n))^2}{(X_r(n))^2 + (X_i(n))^2}} \tag{2.13}$$

Donde $\hat{X}_r(n)$ y $\hat{X}_i(n)$ son las secuencias de valores estimados por la unidad bajo análisis, $X_r(n)$ y $X_i(n)$ son las secuencias de los valores teóricos de la señal de entrada en el instante de tiempo n asignado por la unidad. Los valores $X_r(n)$ y $X_i(n)$ pueden ser determinados en forma muy precisa en determinadas situaciones bien definidas, tales como a frecuencia constante o cambios de fase constante. Las medidas del sincrofasor deberán ser evaluadas usando el criterio TVE de la ecuación (2.13).

2.3.2. Evaluación de la medida de la frecuencia y ROCOF

La medida de la frecuencia y ROCOF será evaluada usando la siguiente definición. Con este criterio, los errores en frecuencia y ROCOF son el valor absoluto entre la diferencia del valor teórico y el valor estimado en Hz y Hz/s respectivamente. Por lo tanto los errores en la medición resultan:

$$FE == |f_{verdadero} - f_{medido}| = |\Delta f_{verdadero} - \Delta f_{medido}| \tag{2.14}$$

$$RFE == |(df/dt)_{verdadero} - (df/dt)_{medido}| \tag{2.15}$$

Los valores medido y verdadero son para el mismo instante de tiempo y estarán dados por la etiqueta de tiempo de los valores estimados.

2.4. Introducción a las PMU

2.4.1. Definición según norma IEEE [2]

Una PMU es un equipo que produce estimaciones de fasores sincronizados, de frecuencia y de variación de frecuencia (ROCOF o Rate Of Change Of Frequency) a partir de señales de tensión y/o corriente y una señal de sincronización horaria o temporal. El mismo equipo IED (Intelligent Electronic Device) además de la función de PMU puede desarrollar otras funciones y hasta tener otro nombre funcional; por ejemplo un equipo que también realice registros oscilográficos llamado DFR (Digital Fault Recorder), o que también realice funciones de protección numérica y sea llamado protección o relé de protección.

Una PMU debe calcular los sincrofasores y poder reportarlos a una tasa constante. Las estimaciones deben incluir sincrofasores de fase o de secuencia positiva o ambos y estas medidas deben poder ser seleccionables por el usuario.

Una PMU puede realizar otras medidas de forma sincronizada con las medidas anteriormente especificadas (sincrofasores, frecuencia y ROCOF), como ser estados booleanos, muestreos de formas de onda u otros datos calculados. Evidentemente dos PMU idénticas (con igual hardware y algoritmos) deben producir los mismos fasores en todas las condiciones.

2.4.2. Definición según NASPI [4]

Una PMU es un equipo que provee como mínimo medidas de sincrofasores y de frecuencia de una o más formas de onda de tensión y/o corriente alterna trifásica. Los sincrofasores pueden ser por fase o en componentes simétricas. Los sincrofasores y valores de frecuencia deben satisfacer la definición general y precisión mínima requerida en la norma IEEE Std C37.118-2005. El dispositivo debe proporcionar una salida de datos en tiempo real que cumpla con los requisitos de dicha norma.

La PMU puede proporcionar mediciones de otras formas de onda, analógicas y digitales, incluidas muestras crudas sincronizadas, también puede registrar datos de forma local. Dichas funciones son opcionales pero si se incluyen, su especificación se definirá en los manuales del usuario y en las hojas de datos del dispositivo.

La PMU puede realizar otras funciones, como la retransmisión, la medición o la grabación de la señal de fallo, pero estas otras funciones no deben interferir con el rendimiento de la función PMU.

2.5. Introducción a los PDC [3]

Un PDC (Phasor Data Concentrator) funciona como un nodo en una red de comunicación, donde los sincrofasores de un número de PMU o PDC se correlaciona y se envía como una sola palabra a los PDC y / o aplicaciones de mayor nivel.

Los PDC correlacionan los datos del sincrofasor con la etiqueta de tiempo para crear un conjunto de medidas del sistema. Pueden proporcionarse funciones adicionales, como las siguientes:

1. Varios controles de calidad de los datos del fasor e inserción de indicadores apropiados en los datos correlacionados.

2. Comprueba los indicadores de perturbación y las grabaciones de los archivos de datos para su análisis.

3. Seguimiento del sistema general de medición y visualización de los resultados, así como registro de los eventos.

4. Número de salidas especializadas, como una interfaz directa con un sistema SCADA o EMS.

Los PDC locales, como se muestra en la Figura 2.4, agregan y sincronizan en tiempo los datos de múltiples PMU y proporcionan datos a las aplicaciones. Los PDC regionales de nivel medio recogen datos de sincrofasores de múltiples PDC, realizan controles de calidad de datos y los proporcionan a las aplicaciones. Los PDC de nivel superior (SuperPDC) agrupan y archivan los datos de los sincrofasores.

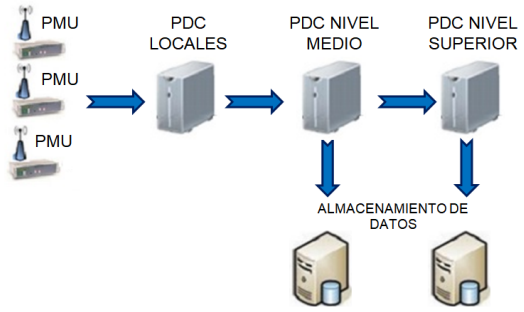


Figura 2.4: Formato de la red de recolección de datos

Puede formarse una jerarquía estructurada de PDC distribuidos para servir a una jerarquía de sistemas: subestación, utilidad, área de control, coordinador de confiabilidad y nivel de interconexión. Cada capa de la jerarquía puede satisfacer diferentes requisitos de datos (latencia, calidad, resolución), con requisitos de captura y almacenado de datos impulsados por aplicaciones.

Puesto que los PDC locales representan un punto local de falla para las cadenas de datos, son necesarias las copias de seguridad y opciones de derivación (bypass) para reducir tales fallas.

2.6. Normas utilizadas en el estudio de Sincrofasores

El estándar donde originalmente se definió al sincrofasor fue el IEEE Std 1344-1995, el cual fue sustituido en 2005 por IEEE Std C37.118-2005. Luego, éste fue dividido en dos partes, el IEEE Std C37.118.1-2011 y el IEEE Std C37.118.2-2011, para armonizar con la norma IEC 61850, norma de la IEC para la comunicación entre IED.

En la norma IEEE Std C37.118.1-2011 se definen las características en las medidas del sincrofasor (fasor sincronizado), frecuencia, ROCOF. Describiendo etiquetas de tiempo y los requerimientos para la medida de estas tres cantidades, especificando métodos de evaluación y requerimientos mínimos para cumplir con el estándar bajo condiciones estáticas y dinámicas. También define la PMU (phasor measurement unit) aunque el estándar no especifica hardware, software o métodos para manipular los fasores, frecuencia o ROCOF.

El IEEE Std C37.118.2-2011 define la forma de intercambio entre los equipos del sistema eléctrico de los datos de los sincrofasores. Especificando los mensajes incluyendo tipos, uso, contenido y formato de datos para la comunicación en tiempo real entre las unidades de medida (PMU, PDC).

La norma IEC 61850, surgió en los 80-90s como una norma de comunicación entre IED en subestaciones y a evolucionado hasta convertirse en un sistema de comunicación ampliamente utilizado, ha sido natural considerarla para los sincrofasores.

Debido a que la IEEE Std C37.118.2-2011 no es específicamente un protocolo de comunicación, sino un formato con métodos básicos para la transferencia de datos, y la IEC 61850 es un protocolo de comunicación muy difundido, se buscó utilizar lo mejor de ambos. Los datos desde la IEEE Std C37.118.2-2011 pueden ser migrados directamente a la IEC 61850, el problema es que muchas de las características de ventajas de IEC 61850 no son usadas. Igualmente hay aspectos de la IEC 61850 como la administración y configuración remota que no contempla IEEE Std C37.118.2-2011. Ambos grupos, IEEE y IEC trabajan en la integración de los dos protocolos, esto hace que los equipos usen interfaces que integren funcionalidades de ambas normas.

El estándar IEEE Std C37.244-2013 describe las funcionalidades, rendimiento y comunicaciones que deben presentar los concentradores de datos (PDC) para protección, control y monitoreo de los sistemas de potencia.

Capítulo 3

Líneas de transmisión - Modelos y algoritmos

Las líneas de transmisión son los elementos del sistema de potencia que se encargan de transportar energía desde el sitio en dónde es generada hasta el punto donde se consume o distribuye. Ellas son unos de los ejes centrales de este proyecto, por este motivo se estudiará el modelo de cuadripolo pasivo, para llegar al cálculo de los parámetros de la línea. Luego se considerarán los distintos efectos debido a los flujos de carga y al efecto Ferranti para asegurar un correcto uso de los algoritmos de detección y localización de fallas.

3.1. Generalidades [1]

Las líneas de transporte de energía eléctrica se encuentran modeladas mediante cuatro parámetros, que se distribuyen en forma uniforme a lo largo de toda la línea:

- La resistencia serie por unidad de longitud (r), dada por las pérdidas por efecto Joule y por el efecto skin.
- La conductancia paralela por unidad de longitud (g), representando las pérdidas a través de los aisladores y por efecto corona. Este parámetro es generalmente despreciable en líneas aéreas de media y alta tensión.
- La inductancia serie por unidad de longitud (l), producida por la autoinductancia de los conductores y el enlace magnético existente entre los mismos. Dado que la inductancia es altamente dependiente de la disposición física de los conductores, ésta varía según el diseño de la torre de alta tensión. En particular, se debe notar que al no respetarse una disposición física de tresbolillo, la inductancia asociada a cada fase será diferente, y por lo tanto, la línea presentará un desequilibrio natural en las impedancias de fase. Esto puede ser compensado mediante la transposición de fases, que tenderá a equilibrar la impedancia vista por fase de la línea en forma global.

- La capacidad paralela por unidad de longitud (c), debida a la diferencia de potencial existente entre los conductores de la línea y de los conductores a tierra. Por este fenómeno, existe circulación de corrientes de carga y asociada a estas capacidades, que deberán ser tomadas en consideración a los efectos del correcto modelado de la línea.

No se profundiza sobre el cálculo de estos parámetros en este documento. Solamente se asume a efectos de los análisis, que la línea se encuentra perfectamente transpuesta, y bajo esa hipótesis es posible realizar el análisis por fase, habitual en los sistemas trifásicos equilibrados, operando en régimen permanente.

3.2. Modelo matemático de parámetros distribuidos [1]

El modelo de una línea de transporte en parámetros distribuidos servirá para:

- Comprender como evolucionan la tensión y la corriente de una línea en función de la posición respecto a sus extremos y las condiciones de borde que le sean impuestas.
- Conocer el comportamiento de la línea en sus extremos a partir de la deducción de un modelo en parámetros concentrados. El interés real de este modelo, radica en su utilización como insumo en los problemas de flujo de potencia, donde no interesa el comportamiento de la línea en sí misma, sino el comportamiento en sus extremos (tensión de operación y potencia que fluye hacia el nodo terminal).

Asumiendo transposición perfecta de las líneas, y que las mismas pueden ser modeladas a partir de los parámetros distribuidos descritos anteriormente, se define:

- $\bar{z} = r + jwl$ como impedancia serie por unidad de longitud de una fase.
- $\bar{y} = g + jwc$ como admitancia paralela por unidad de longitud de una fase.

El modelo que se desarrolla, asume que la línea, como conductor continuo, puede discretizarse como una sucesión de infinitos cuadripolos infinitesimales, donde se manifiestan los efectos dados por la admitancia paralela e impedancia serie. Esto se encuentra representado en la figura 3.1, donde se focaliza la atención en uno de estos cuadripolos infinitesimales. Asimismo, y considerando el comportamiento de la línea en sus extremos, se define:

- V_S: tensión de la línea en el nodo emisor (o local).
- V_R: tensión de la línea en el nodo receptor (o remoto).
- I_S: corriente de la línea en el nodo emisor (o local).
- I_R: corriente de la línea en el nodo receptor(o remoto).

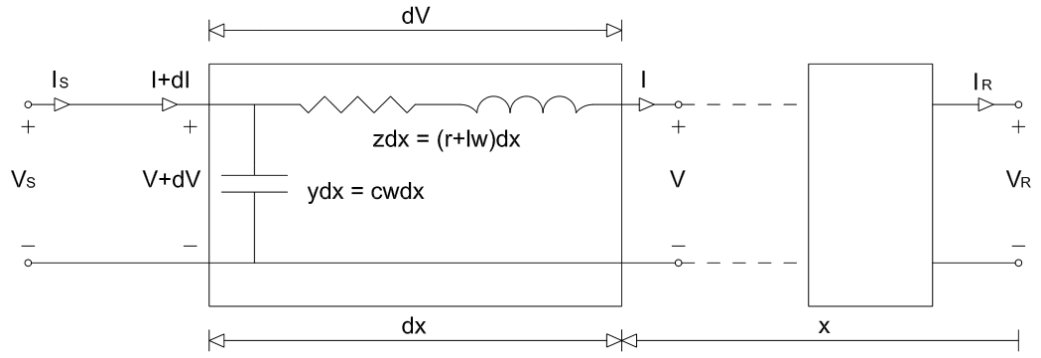


Figura 3.1: Modelo de línea con parámetros distribuidos

Para el desarrollo del análisis que continúa, es importante observar con ayuda de la figura 3.1, que se asume que el eje x tiene como origen el extremo receptor de la línea, y x aumenta de derecha a izquierda.

Considerando un cuadripolo infinitesimal de longitud dx, ubicado a una distancia x del extremo receptor, se puede calcular el diferencial de tensión dV entre sus terminales como:

$$\frac{d\overline{V}}{dx} = \overline{z}\overline{I} \tag{3.1}$$

Por otra parte, el diferencial de corriente circulando por la susceptancia se puede expresar como:

$$\frac{d\overline{I}}{dx} = \overline{y}\,\overline{V} \tag{3.2}$$

Derivando ambas ecuaciones respecto a la posición x, y sustituyendo una en otra, se deducen las ecuaciones diferenciales que determinan la evolución de la tensión y la corriente como función de la distancia x[1].

$$\begin{aligned} \frac{d^2\overline{V}}{dx^2} &= \overline{z}\frac{d\overline{I}}{dx} = \overline{z}\,\overline{y}\,\overline{V} \\ \frac{d^2\overline{I}}{dx^2} &= \overline{y}\frac{d\overline{V}}{dx} = \overline{z}\,\overline{y}\,\overline{I} \end{aligned} \tag{3.3}$$

Este sistema de ecuaciones diferenciales es fácilmente resoluble en forma analítica. A los efectos de su resolución, se necesita conocer al menos dos condiciones de borde. Si se considera conocida la tensión (V_R) y la corriente (I_R) en $x = 0$, la solución de las ecuaciones diferenciales es la siguiente:

[1]Recordar que x se define como la distancia desde el receptor de la línea. Es decir, el punto receptor corresponde a $x = 0$, y dicha variable es creciente hacia el extremo de la fuente (derecha a izquierda, según dibujo).

$$\begin{aligned} \overline{V}(x) &= \overline{V_R}\, ch(\overline{\gamma}x) + \overline{I_R}\,\overline{Z_c}\, sh(\overline{\gamma}x) \\ \overline{I}(x) &= \frac{\overline{V_R}}{\overline{Z_c}} sh(\overline{\gamma}x) + \overline{I_R}\, ch(\overline{\gamma}x) \end{aligned} \tag{3.4}$$

Donde se define:

- $\overline{\gamma} = \sqrt{\overline{z}\,\overline{y}} = \alpha + j\beta$ como la constante de propagación; siendo α la constante de atenuación y β la constante de fase.
- $\overline{Z_c} = \sqrt{\frac{\overline{z}}{\overline{y}}}$ es la impedancia característica de la línea.

Se debe recordar que tanto z como y, son parámetros por unidad de longitud. Más adelante, toma relevancia la longitud de la línea, que se denomina como L.

3.3. Modelado de línea como cuadripolo pasivo [1]

Para el problema que estamos tratando, solo consideramos las tensiones y corrientes en el extremo de la línea. Por lo tanto si se resuelven las ecuaciones diferenciales considerando las condiciones de borde, llegamos a una descripción de una línea como un cuadripolo.

De acuerdo a la figura 3.1, y considerando conocidas las condiciones de borde: V_R e I_R, correspondientes al punto $x = 0$; V_S e I_S, correspondientes al punto $x = L$, se observa que el sistema de ecuaciones que definen el comportamiento de la línea se reduce a:

$$\begin{aligned} \overline{V_S} &= \overline{V_R} ch(\overline{\gamma}L) + \overline{Z_c}\,\overline{I_R}\, sh(\overline{\gamma}L) \\ \overline{I_S} &= \frac{\overline{V_R}}{\overline{Z_c}} sh(\overline{\gamma}L) + \overline{I_R}\, ch(\overline{\gamma}L) \end{aligned} \tag{3.5}$$

Si se compara la expresión (3.5) con la genérica que definen los cuadripolos, se tiene que:

$$\begin{aligned} \overline{V_S} &= \overline{A}\,\overline{V_R} + \overline{B}\,\overline{I_R} \\ \overline{I_S} &= \overline{C}\,\overline{V_R} + \overline{D}\,\overline{I_R} \end{aligned} \tag{3.6}$$

Por lo cual, se deduce que una línea como cuadripolo pasivo, estará definida por los siguientes parámetros:

$$\begin{aligned} \overline{A} &= ch(\overline{\gamma}L) \\ \overline{B} &= \overline{Z_c}\, sh(\overline{\gamma}L) \\ \overline{C} &= \frac{sh(\overline{\gamma}L)}{\overline{Z_c}} \\ \overline{D} &= ch(\overline{\gamma}L) \end{aligned} \tag{3.7}$$

Se observa que el cuadripolo obtenido es naturalmente simétrico ($\overline{A} = \overline{D}$) y pasivo:

$$\overline{AD} - \overline{BC} = ch^2(\overline{\gamma}L) - sh^2(\overline{\gamma}L) = 1 \tag{3.8}$$

El principal interés de estos cálculos radican en determinar correctamente los parámetros de las líneas mediante la lectura de tensiones y corrientes en los extremos. Una vez determinados dichos parámetros, se utilizan luego, para localizar de forma precisa los puntos de falla. Resulta entonces necesario la representación de las líneas mediante el modelo de parámetros concentrados (Ver figura 3.2), lo cual se puede abordar mediante la teoría de cuadripolos.

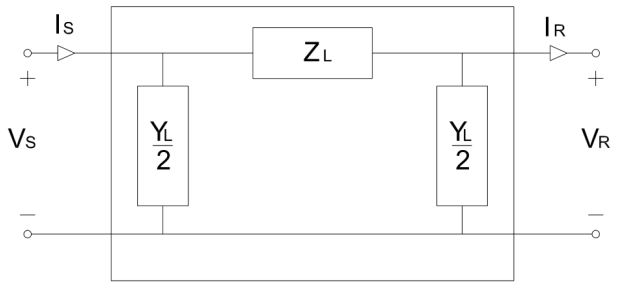


Figura 3.2: Modelo de línea con parámetros concentrados

Las constantes del cuadripolo en función de los parámetros concentrados de la línea resultan:

$$\begin{aligned} \overline{A} = \overline{D} &= 1 + \frac{\overline{Y_L}}{2}\overline{Z_L} \\ \overline{B} &= \overline{Z_L} \end{aligned} \tag{3.9}$$

De forma directa surge que:

$$\overline{Z_L} = \overline{Z_c}\, sh(\overline{\gamma}L) \tag{3.10}$$

Por otra parte:

$$\overline{A} = ch(\overline{\gamma}L) = 1 + \frac{\overline{Y_L}}{2}\overline{Z_c}\, sh(\overline{\gamma}L) \tag{3.11}$$

Despejando $\frac{\overline{Y_L}}{2}$ resulta:

$$\frac{\overline{Y_L}}{2} = \frac{ch(\overline{\gamma}L) - 1}{\overline{Z_c}\, sh(\overline{\gamma}L)} \tag{3.12}$$

Utilizando la identidad $tanh\left(\frac{\overline{\gamma}L}{2}\right) = \frac{ch(\overline{\gamma}L)-1}{sh(\overline{\gamma}L)}$ se obtiene:

$$\frac{\overline{Y_L}}{2} = \frac{1}{\overline{Z_c}} tanh\left(\frac{\overline{\gamma}L}{2}\right) \tag{3.13}$$

Desarrollando el modelo de línea en formato de cuadripolos, se puede llegar a una relación explícita entre la impedancia Z_L con la impedancia z y la admitancia $\frac{Y_L}{2}$ con la admitancia y.

$$\overline{Z_L} = \overline{Z_c}\, sh\,(\overline{\gamma} L) = \sqrt{\frac{z}{y}} sh\,(\overline{\gamma} L) = zL \frac{sh\,(\overline{\gamma} L)}{\sqrt{zy} L} \tag{3.14}$$

Se denomina impedancia serie Z como el producto de la impedancia serie por unidad de longitud multiplicada por la longitud de la línea: $Z = zL$

$$\overline{Z_L} = Z \frac{sh\,(\overline{\gamma} L)}{\overline{\gamma} L} \tag{3.15}$$

Procediendo de la misma forma, para el caso de la admitancia serie:

$$\frac{\overline{Y_L}}{2} = \frac{1}{\overline{Z_c}} tanh\left(\frac{\overline{\gamma} L}{2}\right) = \sqrt{\frac{y}{z}} tanh\left(\frac{\overline{\gamma} L}{2}\right) = \frac{2L\sqrt{y}}{2L\sqrt{y}} \sqrt{\frac{y}{z}} tanh\left(\frac{\overline{\gamma} L}{2}\right) \tag{3.16}$$

Se denomina admitancia paralelo Y como el producto de la admitancia paralelo por unidad de longitud multiplicada por la longitud de la línea: $Y = yL$

$$\frac{\overline{Y_L}}{2} = \frac{\overline{Y}}{2} \frac{tanh\left(\frac{\overline{\gamma} L}{2}\right)}{\frac{\overline{\gamma} L}{2}} \tag{3.17}$$

Estas expresiones vinculan la representación de la línea en parámetros concentrados Z_L e Y_L, con los parámetros distribuidos correspondientes. Estas expresiones son válidas para líneas de cualquier tensión y longitud.

3.4. Algoritmo para determinar parámetros de la línea

Para determinar los parámetros de la línea de transmisión se utilizan los valores de tensión y corriente medidos en ambos extremos de la línea.
A continuación se describe el método utilizando las constantes generales de un cuadripolo.

- Del punto 3.3 se deducen las constantes generales del cuadripolo de la siguiente manera:

$$\begin{aligned} \overline{A} &= \frac{\overline{I}_S \overline{V}_S + \overline{I}_R \overline{V}_R}{\overline{I}_S \overline{V}_R + \overline{I}_R \overline{V}_S} \\ \overline{C} &= \frac{\overline{I}_S^2 - \overline{I}_R^2}{\overline{I}_S \overline{V}_R + \overline{I}_R \overline{V}_S} \\ \overline{B} &= \frac{\overline{A}^2 - 1}{\overline{C}} \\ \overline{A} &= \overline{D} \end{aligned} \tag{3.18}$$

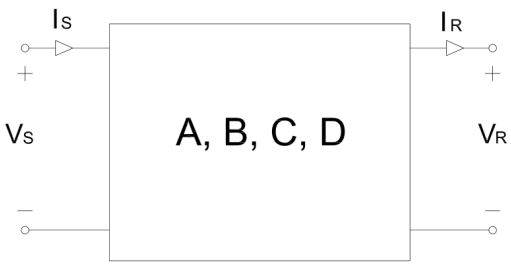


Figura 3.3: Cuadripolo de línea

Con las constantes generales del cuadripolo determinadas se calcula la constante de propagación y la impedancia característica de la línea.

- De la ecuaciones 3.7 y 3.18 , se obtienen $\overline{Z_c}$ y $\overline{\gamma}$.

$$\begin{aligned} \overline{\gamma} &= \frac{ch^{-1}(\overline{A})}{L} \\ \overline{Z_c} &= \frac{\overline{B}}{sh(\overline{\gamma}L)} \end{aligned} \tag{3.19}$$

Luego la impedancia y admitancia total de la línea quedan:

$$\begin{aligned} \overline{Z}_L = \overline{Z}_c\, sh(\overline{\gamma}L) &= \overline{B} \\ \frac{\overline{Y}_L}{2} &= \frac{\overline{A}-1}{\overline{B}} \end{aligned} \tag{3.20}$$

3.5. Consideraciones para una correcta implementación del algoritmo

3.5.1. Flujos de potencia

Para determinar los parámetros del cuadripolo, los sentidos de las corrientes medidas por las PMU deben estar acordes al modelo de la figura 3.3.
En condiciones normales de operación de la línea, en uno de sus extremos la corriente medida por el transformador de corriente es opuesta a la utilizada en el modelo. Por lo tanto se debe invertir esta corriente para determinar las constantes generales del cuadripolo (ecuación 3.18).
Para determinar cual es la corriente invertida se calcula la potencia aparente S en un extremo de la línea y se evalúa el signo de la potencia activa, si es menor a cero se debe invertir esa corriente, de lo contrario se invierte la corriente en el otro extremo.

3.5.2. Efecto Ferranti en líneas de transmisión

Cuando una línea de transmisión se encuentra en circuito abierto o con muy poca carga en su extremo final, se produce un efecto en el cual la tensión en el extremo final es mayor que la tensión en el extremo inicial. Este efecto se lo conoce como efecto Ferranti.
Este efecto se ve reflejado en los resultados de las ecuaciones 3.18 y 3.20, obteniéndose valores de resistencia serie menores a cero.
Por lo tanto el algoritmo deberá evaluar los niveles de tensión en el extremo local y remoto de la línea, corrigiendo el valor de la resistencia serie si se detecta este efecto.

3.5.3. Reactores en las líneas

Algunas de las líneas de transmisión de 500KV de la red uruguaya presentan reactores en uno o ambos extremos. Para determinar los parámetros de la línea según el modelo de la figura 3.3 debemos operar con las corrientes que circulan por ella. La corriente medida en un extremo que tiene un reactor, no es la corriente que circula exactamente por la línea. Se debe analizar si la corriente en ese extremo es entrante o saliente a la línea y como es influida por la corriente del reactor.

A continuación se analizan las tres conexiones posibles:

En la figura 3.4 se observa la presencia de reactores en el extremo s de la línea. La corriente I'_s es la corriente a utilizar para el cálculo de los parámetros de la línea. Dependiendo del sentido del flujo de potencia, esta corriente sera entrante o saliente a la línea.

$$I'_s = I_s + I_{reactor} \tag{3.21}$$

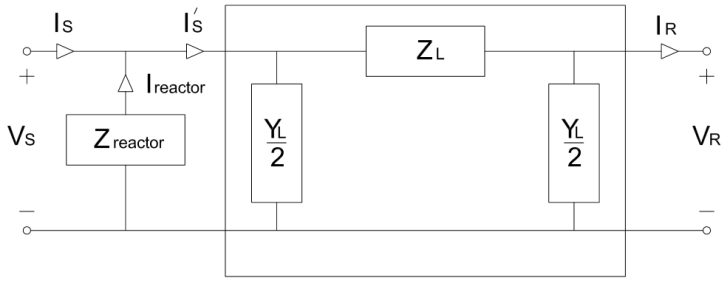


Figura 3.4: Circuito con reactor en extremo S

Análogamente al caso anterior en la figura 3.5 se observa la presencia de reactores en el extremo R de la línea. La corriente a utilizar en el cálculo de los parámetros de la línea es:

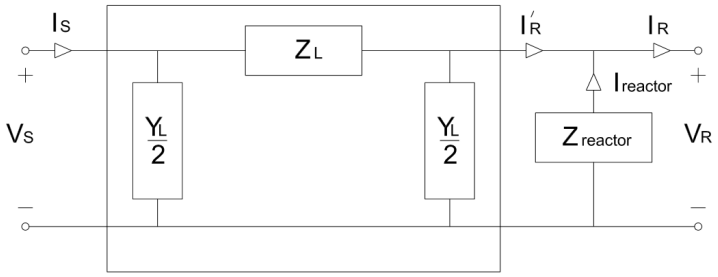


Figura 3.5: Circuito con reactor en extremo R

$$I'_r = I_r - I_{reactor} \tag{3.22}$$

Por último en la figura 3.6 se observa la presencia de reactores en ambos extremos de la línea. Las corrientes a utilizar en el cálculo de los parámetros de la línea son:

$$\begin{aligned} I'_s &= I_s + I_{reactor} \\ I'_r &= I_r - I_{reactor} \end{aligned} \tag{3.23}$$

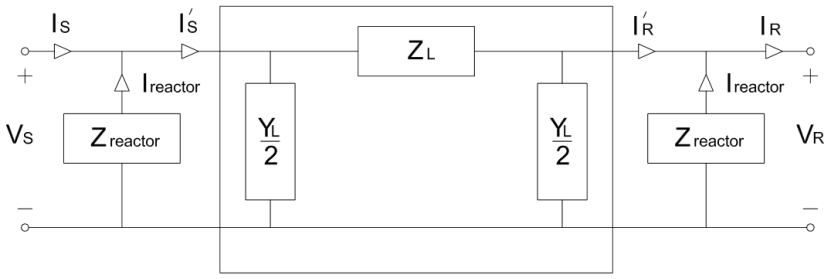


Figura 3.6: Circuito con reactor en extremo S y R

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 4

Fallas eléctricas en líneas de transmisión

Las fallas eléctricas en las líneas de transmisión son inevitables, por distintas razones que se analizarán más adelante. Los esfuerzos de los responsables de los servicios se concentran en tratar de minimizar los daños.

Existen dos grandes tipos de fallas:

- falla de aislación: cortocircuito
- falla en la conducción: línea abierta.

4.1. Causas, naturaleza y efectos de los cortocircuitos [6]

Se agrupa bajo el nombre de "cortocircuito" a todos los incidentes provocados por un contacto entre un conductor y la tierra o entre un conductor y otra pieza metálica, o bien entre conductores. En lo que concierne a instalaciones de alta tensión, casi la totalidad de los casos, el contacto tiene lugar por intermedio de un arco y no necesariamente un contacto físico.

Los cortocircuitos tienen distintas causas:

1. de origen puramente eléctrico: provienen del deterioro de un aislador, que se vuelve incapaz de soportar la tensión.

2. de origen mecánico: se debe a la ruptura de conductores o de aisladores, a la caída de un objeto extraño (por ejemplo, ramas de árbol sobre una línea aérea), daños sobre un cable subterraneo debido a una excavación, etc.

3. de origen atmosférico: son causados por un rayo que incide sobre los conductores de una línea o de una subestación o por la disminución de la capacidad dieléctrica del aislante (aire) debido a la niebla o en particular en nuestro país por humo debido a la quema de cubiertas.

4. Originadas por falsas maniobras: por ejemplo, la apertura en carga de un seccionador.

Estos contactos accidentales no afectan en general a todos los conductores simultáneamente. En el caso de redes trifásicas cuya tensión de servicio es mayor que 60 kV, la experiencia muestra que en 70 % a 80 % de los casos, los cortocircuitos se producen (o por lo menos se inician) entre una fase y tierra; si el defecto no es eliminado en forma suficientemente rápida, el arco puede afectar una segunda fase o mismo a la tercera. Los defectos trifásicos que afectan a las líneas aéreas pueden deberse a:

- Fenómenos mecánicos susceptibles de establecer un contacto entre los 3 conductores o de ponerlos a tierra simultáneamente (destrucción de una torre, caída de un árbol sobre una línea, etc.)
- Caída directa de un rayo que puede llevar a la torre a un potencial suficientemente alto como para que el cebado se produzca simultáneamente sobre las 3 fases.
- Falsa maniobra, como la apertura en carga de un seccionador en una subestación en que las fases no están suficientemente separadas.

Los defectos trifásicos que ocurren en los cables subterráneos (en general son cables unipolares) son muy poco frecuentes y se deben en su gran mayoría a causas mecánicas. En las redes de baja tensión, dónde se usan cables tripolares aumenta el riesgo de cortocircuito.

Por último, los cortocircuitos entre 2 fases, sin puesta a tierra, aparecen excepcionalmente y se deben casi exclusivamente a fallas mecánicas.

En general, la presencia de un cortocircuito en una red provoca sobrecorrientes, caídas de tensión y desequilibrios entre las fases de las mismas.

4.2. Ecuaciones para los diferentes tipos de falla [6]

Los diferentes tipos de falla encontrados en los sistemas de potencia son:

- trifásica (FFF)
- una fase a tierra (FT)
- entre dos fases (FF)
- entre dos fases a tierra (FFT)
- una fase abierta

4.2.1. Cortocircuito FFF

- Es equilibrado.
- No hay corrientes ni tensiones inversas ni homopolares.

- Las tensiones de fase colapsan en el punto del defecto. Es lo mismo que FFFT.

$$\begin{cases} \bar{I_A} + \bar{I_B} + \bar{I_C} = 0 \\ \bar{V_A} = \bar{V_B} \\ \bar{V_B} = \bar{V_C} \end{cases} \Longrightarrow \begin{cases} \bar{I_0} = 0 \\ \bar{V_1} = \bar{V_2} = 0 \end{cases} \tag{4.1}$$

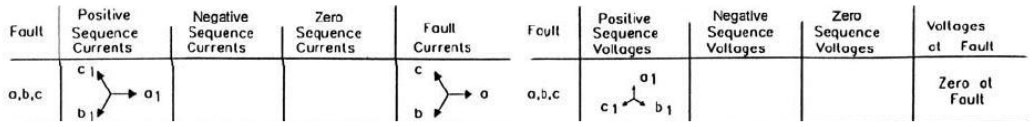


Figura 4.1: Cortocircuito FFF: Corrientes y tensiones. Protective Relaying Theory and Applications, Walter A. Elmore, Mar- cel Dekker Inc. 2nd ed. 2004

4.2.2. Cortocircuito FT

- Existen corrientes de las 3 secuencias y sus módulos son iguales.
- Solo hay corrientes en la fase en falta.
- La tensión de la fase en falla colapsa en el punto del defecto.

$$\begin{cases} \bar{I_B} = 0 \\ \bar{I_C} = 0 \\ \bar{V_A} = 0 \end{cases} \Longrightarrow \begin{cases} \bar{I_1} = \bar{I_2} = \bar{I_0} = \frac{1}{3}\bar{I_A} \\ \bar{V_1} = -(\bar{V_2} + \bar{V_0}) \end{cases} \tag{4.2}$$

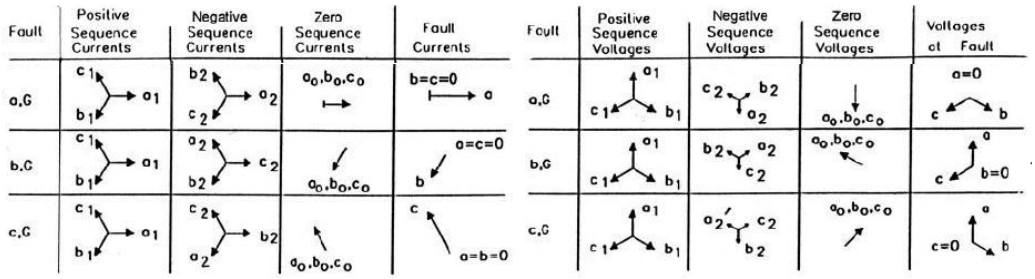


Figura 4.2: Cortocircuito FT: Corrientes y tensiones. Protective Relaying Theory and Applications, Walter A. Elmore, Mar- cel Dekker Inc. 2nd ed. 2004

4.2.3. Cortocircuito FF

- No existen corrientes de secuencia homopolar. No hay tensiones de secuencia homopolar.
- Las corrientes de secuencia positiva y negativa tienen igual módulo.

- Las corrientes de las fases en falta son de igual módulo y en oposición.
- Las tensiones de las fases en falta son iguales en el punto del defecto.

$$\begin{cases} \bar{I_A} = 0 \\ \bar{I_B} = -\bar{I_C} \\ \bar{V_B} = \bar{V_C} \end{cases} \implies \begin{cases} \bar{I_1} = -\bar{I_2} \\ \bar{I_0} = 0 \\ \bar{V_1} = \bar{V_2} \end{cases} \tag{4.3}$$

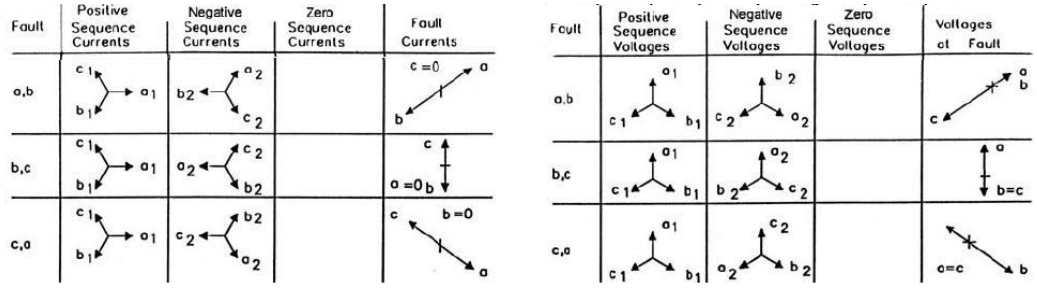


Figura 4.3: Cortocircuito FF: Corrientes y tensiones. Protective Relaying Theory and Applications, Walter A. Elmore, Mar- cel Dekker Inc. 2nd ed. 2004

4.2.4. Cortocircuito FFT

- Existen corrientes de las 3 secuencias.
- Las corrientes de las fases en falla son de igual módulo.
- Las tensiones de las fases en falla colapsan en el punto del defecto.

$$\begin{cases} \bar{I_A} = 0 \\ \bar{V_A} = 0 \\ \bar{V_C} = 0 \end{cases} \implies \begin{cases} \bar{I_1} = -(\bar{I_2} + \bar{I_0}) \\ \bar{V_1} = \bar{V_2} = \bar{V_0} \end{cases} \tag{4.4}$$

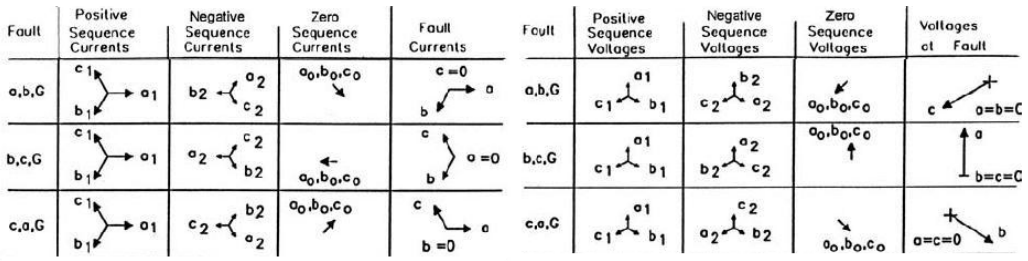


Figura 4.4: Cortocircuito FFT: Corrientes y tensiones. Protective Relaying Theory and Applications, Walter A. Elmore, Mar- cel Dekker Inc. 2nd ed. 2004

4.2.5. Ecuaciones para fase abierta

- Existen corrientes de las 3 secuencias.
- La corriente de la fases abierta es cero.
- Aparece una tensión en bornes del hilo abierto.
- En las fases sanas, la tensiones son iguales a cero.

$$\begin{cases} \bar{I_A} = 0 \\ \bar{V_B} = 0 \\ \bar{V_C} = 0 \end{cases} \implies \{\ \bar{V_0} = \bar{V_1} = \bar{V_2} = \tfrac{1}{3}\bar{V_A} \tag{4.5}$$

4.3. Algoritmos de identificación de fallas

Las ecuaciones para los diferentes tipos de falla vistas en la sección 4.2 son resultado de un análisis ideal, el cual sirve para comprender la idea general.
Teniendo en cuenta que se trabaja con datos reales de corrientes y tensiones medidos por las PMU, se diseña un algoritmo basado en datos reales.

Considerando que las corrientes a través de la línea no sufren grandes cambios en régimen normal, se decide identificar las fallas mediante un incremento de la corriente.

Se buscará un aumento de la corriente entre medidas consecutivas de cada fase superior al $10\,\%I_n$, acompañado de una caída de tensión de cada fase superior al $20\,\%V_n$. La caída de tensión es usada como confirmación de la falla.
Se confirmara la existencia de una falla cuando dos valores de corriente y tensión consecutivos medidos por fase cumplen las condiciones planteadas en 4.6:

$$\begin{cases} \frac{\Delta I_x}{I_n} = \frac{I_i - I_{i-1}}{I_n} > 0{,}1 \\ \\ \frac{\Delta V_x}{V_n} = \frac{V_i - V_{i-1}}{V_n} < -0{,}2 \end{cases} \tag{4.6}$$

En donde los sub-índices de las inecuaciones 4.6, x hace referencia a las fases, i a los números de las muestras medidas y n hace referencia a los valores nominales.

La forma de identificar el tipo de falla se presenta en la tabla 4.1

4.4. Algoritmos para la localización de fallas

Actualmente, hay muchos métodos para localizar las fallas en líneas de transmisión.
En este documento serán analizados dos casos particulares:

1. Métodos utilizando datos de un extremo de la línea:
   - Algoritmo de Reactancia Simple.

Tipo de falla	$\Delta I_A > 0{,}1I_n$	$\Delta I_B > 0{,}1I_n$	$\Delta I_C > 0{,}1I_n$	$\Delta V_A < -0{,}2V_n$	$\Delta V_B < -0{,}2V_n$	$\Delta V_C < -0{,}2V_n$	$\Delta I_0 > 0{,}1I_n$
$A-G$	SI	NO	NO	SI	NO	NO	SI
$B-G$	NO	SI	NO	NO	SI	NO	SI
$C-G$	NO	NO	SI	NO	NO	SI	SI
$A-B$	SI	SI	NO	SI	SI	NO	NO
$A-C$	SI	NO	SI	SI	NO	SI	NO
$B-C$	NO	SI	SI	NO	SI	SI	NO
$A-B-G$	SI	SI	NO	SI	SI	NO	SI
$A-C-G$	SI	NO	SI	SI	NO	SI	SI
$B-C-G$	NO	SI	SI	NO	SI	SI	SI
$A-B-C$	SI	SI	SI	SI	SI	SI	NO
$A-B-C-G$	SI	SI	SI	SI	SI	SI	NO

Tabla 4.1: Identificación del tipo de falla

- Algoritmo de Takagi.
- Algoritmo de Novosel.

Estos algoritmos utilizan los sincrofasores de tensión y corriente de un solo extremo de la línea. El principal inconveniente de estos algoritmos es no conocer la magnitud de la corriente de falla del extremo remoto, generando esto un error en la estimación de la distancia al punto de falla. Su principal ventaja es no requerir comunicación con extremo remoto.

2. Métodos utilizando datos de ambos extremos de la línea. Este algoritmo utiliza los sincrofasores de tensión y corriente de ambos extremos de la línea. Es uno de lo más precisos en la estimación de la distancia al punto de falla ya que no realiza ninguna aproximación.

4.4.1. Algoritmo de la Reactancia Simple [5]

El algoritmo de la Reactancia Simple mide la impedancia de la línea utilizando las tensiones y corrientes de la fase fallada. Al relacionar la componente imaginaria de ésta con la reactancia de secuencia positiva de la línea, se puede estimar la distancia de falla.

Este algoritmo fue uno de los primeros en compensar la falta de la medida de resistencia de falla, asumiendo que este valor es despreciable, por lo que la ecuación 4.7, se transforma en 4.8 .

$$\begin{aligned} \overline{V_S} &= m\overline{Z_L}\,\overline{I_S} + R_F\overline{I_F} \\ \frac{\overline{V_S}}{\overline{I_S}} &= m\overline{Z_L} + \frac{\overline{I_F}}{\overline{I_S}}R_F \end{aligned} \tag{4.7}$$

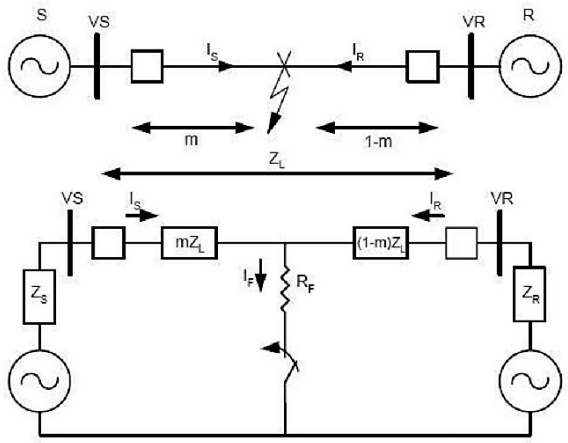


Figura 4.5: Equivalente monofásico de un sistema de potencia en falla

$$\frac{\overline{V_S}}{\overline{I_S}} = m\overline{Z_L} \tag{4.8}$$

Considerando la parte imaginaria a ambos lados de la ecuación 4.8 y despejando m se obtiene:

$$m = \frac{1}{Im\{\overline{Z_L}\}} Im\left\{\frac{\overline{V_S}}{\overline{I_S}}\right\} \tag{4.9}$$

El error de este algoritmo está dado por el término en el que se encuentra la R_F.

$$\varepsilon = \frac{1}{Im\{\overline{Z_L}\}} R_F Im\left\{\frac{\overline{I_F}}{\overline{I_S}}\right\} \tag{4.10}$$

Para el caso de fallas monofásicas, que implican circulación de corrientes de secuencia cero (I_0)[1] y considerando despreciable la resistencia de falla, se obtiene:

$$\overline{V_S} = \overline{V_{S_1}} + \overline{V_{S_2}} + \overline{V_{S_0}} = m\overline{Z_{L_1}}\,\overline{I_{S_1}} + m\overline{Z_{L_2}}\,\overline{I_{S_2}} + m\overline{Z_{L_0}}\,\overline{I_{S_0}} \tag{4.11}$$

Teniendo en cuenta que $\overline{Z_{L_1}} = \overline{Z_{L_2}}$ resulta:

$$\begin{aligned} \bar{V_S} &= m\overline{Z_{L_1}}(\overline{I_{S_1}} + \overline{I_{S_2}} + \overline{I_{S_0}}) + m(\overline{Z_{L_0}} - \overline{Z_{L_1}})\overline{I_{S_0}} \\ \overline{V_S} &= m\overline{Z_{L_1}}\left(\overline{I_{S_1}} + \overline{I_{S_2}} + \overline{I_{S_0}} + \left(\frac{\overline{Z_{L_0}} - \overline{Z_{L_1}}}{\overline{Z_{L_1}}}\right)\overline{I_{S_0}}\right) \end{aligned} \tag{4.12}$$

Considerando que:

$$\begin{aligned} \overline{k_0} &= \frac{\overline{Z_{L_0}} - \overline{Z_{L_1}}}{\overline{Z_{L_1}}} \\ \overline{I_S} &= \overline{I_{S_1}} + \overline{I_{S_2}} + \overline{I_{S_0}} \end{aligned} \tag{4.13}$$

[1]Los subíndices 1, 2 y 0 hacen referencia a las redes de secuencia directa, inversa y homopolar respectivamente.

donde se definió $\overline{k_0}$ que es un coeficiente de impedancia homopolar y $\overline{Z_{L_1}}$ y $\overline{Z_{L_0}}$ hacen referencia a la impedancia de secuencia directa (impedancia síncrona) e impedancia de secuencia cero (impedancia homopolar) de la línea respectivamente.

De esta manera, la distancia al punto de falla resulta:

$$m = \frac{1}{Im\{\overline{Z_{L_1}}\}} Im\left\{\frac{\overline{V_S}}{\overline{I_S} + (\overline{k_0} * \overline{I_{S_0}})}\right\} \tag{4.14}$$

4.4.2. Algoritmo de Takagi [5]

El método de Takagi requiere de datos de pre-falla y datos de falla. De la figura 4.5, que representa un circuito equivalente monofásico de un sistema de potencia en falla, tenemos que:

$$\overline{V_S} = m\overline{Z_L}\,\overline{I_S} + R_F\overline{I_F} \tag{4.15}$$

Se define la corriente superpuesta $\overline{I_{sup}}$, corriente que está en fase con la corriente $\overline{I_F}$.

$$\overline{I_{sup}} = \overline{I_S} - \overline{I_{pre}} \tag{4.16}$$

Siendo $\overline{I_S}$ la corriente de falla y $\overline{I_{pre}}$ la corriente de pre-falla
Multiplicando a ambos lados de la igualdad 4.15 por $\overline{I^*_{sup}}$ [2] y dejando solo la parte imaginaria se obtiene:

$$Im\left\{\overline{V_S}\,\overline{I^*_{sup}}\right\} = mIm\left\{\overline{Z_L}\,\overline{I_S}\,\overline{I^*_{sup}}\right\} + R_F Im\left\{\overline{I_F}\,\overline{I^*_{sup}}\right\} \tag{4.17}$$

Despejando la distancia m, queda:

$$m = \frac{Im\left\{\overline{V_S}\,\overline{I^*_{sup}}\right\}}{Im\left\{\overline{Z_L}\,\overline{I_S}\,\overline{I^*_{sup}}\right\}} \tag{4.18}$$

La llave del éxito de este método es asumir que el ángulo de la corriente $\overline{I_{sup}}$ es igual al ángulo de la corriente de falla $\overline{I_F}$. Si el ángulo entre la corriente $\overline{I_{sup}}$ e $\overline{I_F}$ aumenta, el error de estimar la localización de falla también aumenta.

4.4.3. Algoritmo de Novosel [5]

El algoritmo de Novosel, constituye una versión mejorada del algoritmo de Takagi en que no se requiere conocer las impedancias de las fuentes, tampoco supone que el sistema es homogéneo, la suposición se basa en que las redes de secuencia positiva y negativa son homogéneas pero no la de secuencia cero. En el algoritmo de Novosel se considera que el factor de distribución de corriente de secuencia negativa es un número real, y se obtienen ecuaciones en que el factor de distribución

[2] $\overline{I^*_{sup}}$ hace referencia al complejo conjugado de $\overline{I_{sup}}$

de corriente de secuencia cero no afecta la exactitud del estimado de la localización de la falla cuando ésta involucra tierra. Además, la exactitud del algoritmo no se ve influenciada por la magnitud del factor de distribución de la corriente de secuencia negativa. Esto ocasiona que se tenga una ecuación para cada tipo de falla.

- Falla monofásica a tierra
- Falla bifásicas a tierra
- Falla bifásica
- Falla trifásica

Falla monofásica a tierra

En este tipo de falla las redes de secuencia quedan conectadas en serie.

$$\overline{I_{F_1}} = \overline{I_{F_2}} = \overline{I_{F_0}} = \frac{\overline{I_F}}{3} \tag{4.19}$$

$$\begin{aligned} \overline{I_{sup_{S_1}}} &= \overline{K_1}\,\overline{I_{F_1}} \\ \overline{I_{sup_{S_2}}} &= \overline{K_2}\,\overline{I_{F_2}} = \overline{I_{S_2}} \\ \overline{I_{sup_{S_0}}} &= \overline{K_0}\,\overline{I_{F_0}} = \overline{I_{S_0}} \end{aligned} \tag{4.20}$$

donde $\overline{K_i}$ se define como:

$$\overline{K_i} = K_i e^{j\alpha_i} \tag{4.21}$$

y la corriente medida por el localizador de falla en la fase 'A' es:

$$\overline{I_{S_A}^M} = \overline{I_{S_A}} + \left(\frac{\overline{Z_{L0}} - \overline{Z_{L1}}}{\overline{Z_{L1}}} \right) \overline{I_{sup_{S0}}} \tag{4.22}$$

La tensión medida por el localizador es:

$$\overline{V_{S_A}^M} = m\overline{Z_{L1}}\,\overline{I_{S_A}^M} + R_f \left(\frac{3\overline{I_{sup_{S2}}}}{\overline{K_2}} \right) \tag{4.23}$$

Suponiendo que la red de secuencia negativa es homogénea, es decir $arg(\overline{K_2}) = 0$, se puede considerar $3R_f/\overline{K_2} = K$ resultando

$$\overline{V_{S_A}^M} = m\overline{Z_{L1}}\,\overline{I_{S_A}^M} + K\,\overline{I_{pre_{S2}}} \tag{4.24}$$

Descomponiendo la ecuación anterior en su parte real e imaginaria se obtienen dos ecuaciones con dos incógnitas m y K, resolviendo el sistema de ecuaciones para eliminar K se obtiene la distancia a la falla, quedando:

$$m = \frac{\frac{Re\{\overline{V_S^M}\}}{Re\{\overline{I_{sup_{S2}}^M}\}} - \frac{Im\{\overline{V_S^M}\}}{Im\{\overline{I_{sup_{S2}}^M}\}}}{R_{L1}\left[\frac{Re\{\overline{I_S^M}\}}{Re\{\overline{I_{sup_{S2}}^M}\}} - \frac{Im\{\overline{I_S^M}\}}{Im\{\overline{I_{sup_{S2}}^M}\}}\right] - X_{L1}\left[\frac{Re\{\overline{I_S^M}\}}{Im\{\overline{I_{sup_{S2}}^M}\}} + \frac{Im\{\overline{I_S^M}\}}{Re\{\overline{I_{sup_{S2}}^M}\}}\right]} \tag{4.25}$$

Los factores de distribución de corriente de secuencia positiva y negativa $\overline{K_1}$ y $\overline{K_2}$ pueden considerarse iguales; por lo tanto de (4.20) resulta que en (4.25) puede utilizarse $\overline{I_{sup_{S1}}} = \overline{I_{S1}} - \overline{I_{pre_{S1}}}$ en lugar de $\overline{I_{sup_{S2}}}$. Esto requiere conocer la corriente de pre-falla $\overline{I_{pre_{S1}}}$. Al utilizar la corriente de secuencia positiva se desprecia el desbalance de la línea en estado estable.

$$m = \frac{\frac{Re\{\overline{V_S^M}\}}{Re\{\overline{I_{sup_{S1}}^M}\}} - \frac{Im\{\overline{V_S^M}\}}{Im\{\overline{I_{sup_{S1}}^M}\}}}{R_{L1}\left[\frac{Re\{\overline{I_S^M}\}}{Re\{\overline{I_{sup_{S1}}^M}\}} - \frac{Im\{\overline{I_S^M}\}}{Im\{\overline{I_{sup_{S1}}^M}\}}\right] - X_{L1}\left[\frac{Re\{\overline{I_S^M}\}}{Im\{\overline{I_{sup_{S1}}^M}\}} + \frac{Im\{\overline{I_S^M}\}}{Re\{\overline{I_{sup_{S1}}^M}\}}\right]} \tag{4.26}$$

Falla bifásica a tierra

En este caso las redes de secuencia quedan conectadas en paralelo, la tensión en la falla esta dado por:

$$\overline{V_{S_1}} - \overline{V_{S_2}} = m\overline{Z_{L_1}}(\overline{I_{S_1}} - \overline{I_{S_2}}) + \frac{R_F}{2}(\overline{I_{F_1}} - \overline{I_{F_2}}) \tag{4.27}$$

Aplicando (4.20) a la ecuación anterior, resulta:

$$\overline{V_{S_1}} - \overline{V_{S_2}} = m\overline{Z_{L_1}}(\overline{I_{S_1}} - \overline{I_{S_2}}) + \frac{R_F}{2}\left(\frac{\overline{I_{sup_{S1}}}}{\overline{K_1}} - \frac{\overline{I_{sup_{S2}}}}{\overline{K_2}}\right) \tag{4.28}$$

Suponiendo los factores de distribución iguales ($\overline{K_1} = \overline{K_0}$), sustituyendo las componentes simétricas por magnitudes de fase, en este caso una falla entre las fases b y c:

$$\frac{\overline{V_{S_B}} - \overline{V_{S_C}}}{\overline{a}^2 - \overline{a}} = m\overline{Z_{L1}}\left(\frac{\overline{I_{S_B}} - \overline{I_{S_C}}}{\overline{a}^2 - \overline{a}}\right) + \frac{R_f}{2}\left(\frac{1}{\overline{K_1}}\frac{\overline{I_{sup_{S_B}}} - \overline{I_{sup_{S_C}}}}{\overline{a}^2 - \overline{a}}\right) \tag{4.29}$$

donde $a = 1\angle 120^0$

Suponiendo que la red de secuencia negativa es homogénea, es decir $arg(\overline{K_1}) = 0$, se puede considerar $R_f/2\overline{K_1} = \overline{K}$ resultando:

$$\overline{V_S^M} = m\overline{Z_{L_1}}\overline{I_S^M} + \overline{K}\overline{I_{sup_S}^M} \tag{4.30}$$

Separando en sus partes real e imaginaria y resolviendo para eliminar $\overline{K}$ se obtiene:

$$m = \frac{\frac{Re\{\overline{V_S^M}\}}{Re\{\overline{I_{sup_S}^M}\}} - \frac{Im\{\overline{V_S^M}\}}{Im\{\overline{I_{sup_S}^M}\}}}{R_{L1}\left[\frac{Re\{\overline{I_S^M}\}}{Re\{\overline{I_{sup_S}^M}\}} - \frac{Im\{\overline{I_S^M}\}}{Im\{\overline{I_{sup_S}^M}\}}\right] - X_{L1}\left[\frac{Re\{\overline{I_S^M}\}}{Im\{\overline{I_{sup_S}^M}\}} + \frac{Im\{\overline{I_S^M}\}}{Re\{\overline{I_{sup_S}^M}\}}\right]} \tag{4.31}$$

Falla bifásica

En este caso $\overline{I_{S_1}} = -\overline{I_{S_2}}$ sustituyendo esta condición en (4.27), utilizando (4.20) y reemplazando $\overline{a}^2 - \overline{a}$ por $-j\sqrt{3}$ resulta:

$$\overline{V_S^M} = m\overline{Z_{L_1}}\overline{I_S^M} + R_f\frac{\overline{I_{S_2}}}{\overline{K_2}}\left(-j\sqrt{3}\right) \tag{4.32}$$

Haciendo la suposición de que $arg(\overline{K_2}) = arg(\overline{I_{sup_{S_2}}}/\overline{I_{F_2}}) = 0$ se puede considerar $\sqrt{3}R_f/\overline{K_2} = \overline{K}$ resultando:

$$\overline{V_S^M} = m\overline{Z_{L_1}}\,\overline{I_S^M} + j\overline{K}\,\overline{I_{S_2}} \tag{4.33}$$

Separando la ecuación anterior en partes real e imaginaria y resolviendo para eliminar $\overline{K}$ se obtiene la ecuación de la distancia a la falla:

$$m = \frac{\frac{Re\{\overline{V_S^M}\}}{-Im\{\overline{I_{S_2}^M}\}} - \frac{Im\{\overline{V_S^M}\}}{Re\{\overline{I_{S_2}^M}\}}}{R_{L_1}\left[\frac{Re\{\overline{I_S^M}\}}{-Im\{\overline{I_{S_2}}\}} - \frac{Im\{\overline{I_S^M}\}}{Re\{\overline{I_{S_2}}\}}\right] - X_{L_1}\left[\frac{Re\{\overline{I_S^M}\}}{Re\{\overline{I_{S_2}}\}} + \frac{Im\{\overline{I_S^M}\}}{-Im\{\overline{I_{S_2}}\}}\right]} \tag{4.34}$$

Falla trifásica

En el caso de la falla trifásica solo se utiliza la red de secuencia positiva, de acuerdo al algoritmo de Novosel se pueden utilizar las señales de cualquiera de las tres fases. La tensión medida en la fase A es:

$$\overline{V_{S_A}} = m\overline{Z_{L_1}} + R_f\overline{I_F} + R_f\left(\frac{\overline{I_{sup_{S_A}}}}{\overline{K_1}}\right) \tag{4.35}$$

Haciendo la suposición de que $arg(\overline{K_1}) = 0$ se puede considear $R_f/\overline{K_1} = \overline{K}$, entonces resulta:

$$\overline{V_{S_A}} = m\overline{Z_{L_1}}\,\overline{I_{S_A}} + \overline{K}\,\overline{I_{sup_{S_A}}} \tag{4.36}$$

Descomponiendo la ecuación anterior en partes real e imaginaria y resolviendo para eliminar $\overline{K}$ resulta que la distancia a la falla:

$$m = \frac{\frac{Re\{\overline{V_S^M}\}}{Re\{\overline{I_{sup_{S_A}}}\}} - \frac{Im\{\overline{V_S^M}\}}{Im\{\overline{I_{sup_{S_A}}}\}}}{R_{L_1}\left[\frac{Re\{\overline{I_{S_A}}\}}{Re\{\overline{I_{sup_{S_A}}}\}} - \frac{Im\{\overline{I_{S_A}}\}}{Im\{\overline{I_{sup_{S_A}}}\}}\right] - X_{L_1}\left[\frac{Re\{\overline{I_{S_A}}\}}{Im\{\overline{I_{sup_{S_A}}}\}} + \frac{Im\{\overline{I_{S_A}}\}}{Re\{\overline{I_{sup_{S_A}}}\}}\right]} \tag{4.37}$$

En (4.37) el cálculo de la corriente $\overline{I_{sup_{S_A}}} = \overline{I_{S_A}} - \overline{I_{pre_{S_A}}}$ requiere conocer la corriente de pre-falla $\overline{I_{pre_{S_A}}}$.

4.4.4. Métodos utilizando datos de ambos extremos

La figura 4.5 representa una línea en falla con medidas de tensión y corriente en ambos extremos.

Si se aplica la Ley de Kirchhoff de tensión al circuito de la figura 4.5 se obtiene:

$$\begin{aligned} \overline{V_S} &= m\overline{Z_L}\,\overline{I_S} + \overline{I_F}R_F \\ \overline{V_R} &= (1-m)\overline{Z_L}\,\overline{I_R} + \overline{I_F}R_F \\ \overline{Z_L} &= R_L + jX_L \end{aligned} \tag{4.38}$$

Resolviendo el sistema de ecuaciones 4.38, se llega al valor de m

$$m = \frac{V_S - V_R + Z_L.I_R}{Z_L(I_S + I_R)} \tag{4.39}$$

Capítulo 5

Software ILFUS

En este capitulo se describe la forma en que se implementaron los algoritmos, desde el lenguaje elegido, pasando por la forma en que se recolectan los datos, hasta el pseudocódigo de los casos de uso para correr tanto el algoritmo de un extremo como el de dos extremos.

5.1. Base de datos

La red de potencia de UTE, cuenta con dos PDC instalados. Uno se encuentra en funcionamiento almacenando datos y el otro esta fuera de servicio.
La base de datos de los PDC se encuentra almacenada en formato swave, en 3 archivos de 10GB aproximadamente, los cuales almacenan la información de todas las PMU instaladas durante 24 horas. Este formato esta optimizado para el software del fabricante SEL[1]. Dicho formato no está estandarizado y por tanto no se puede acceder a la información necesaria.
Se estudiaron distintas opciones para acceder a la base de datos. Dentro de las opciones encontradas, el Synchrawave[1] permite la exportación de las bases de datos en distintos formatos de archivo, entre ellos el formato csv. Este formato es sencillo para representar datos en forma de tabla separados por algún delimitador, como por ejemplo coma, dos puntos, etc. Dicho archivo contiene los datos de los sincrofasores con sus respectivas estampas de tiempo. De esta manera se puede acceder a los datos de las PMU, debiendo usar la herramienta de SEL Synchrowave[1].

En la figura 5.1 se observa un ejemplo reducido a una única fase, de los datos obtenidos a través de las mediciones de las PMUs de una línea en formato csv.

Timestamp	2016_MA5PA5_ F91:Current A:Magnitude	2016_MA5PA5_ F91:Current A:Angle	2016_MA5PA5_ F91:Voltage A:Magnitude	2016_MA5PA5_ F91:Voltage A:Angle	2016_PA5MA5_ F96:Current A:Magnitude	2016_PA5MA5_ F96:Current A:Angle	2016_PA5MA5_ F96:Voltage A:Magnitude	2016_PA5MA5_ F96:Voltage A:Angle
2017/04/21 13:00:05.640	3,417,294	-1,237,757	295,186,625	596,292	3,513,583	749,267	2,931,806,875	638,793
2017/04/21 13:00:05.660	3,409,984	-1,245,173	2,951,365,938	589,499	3,509,794	743,023	2,931,515,625	631,915
2017/04/21 13:00:05.680	3,412,122	-1,252,364	2,950,611,563	582,422	3,515,677	735,784	2,931,019,063	624,893

Figura 5.1: Ejemplo de formato de archivo .csv

[1]Synchrowave: software proporcionado por SEL para manejo de la información

Interpretación de los datos en formato .csv

La primer fila da la tabla 5.1, corresponde a los títulos, parámetros que se miden y desde dónde se miden. Mientras que las siguientes 3 filas corresponden a los datos medidos. Los valores de las magnitudes medidas se encuentran multiplicados por $1 * 10^4$ por lo cual para obtener los valores reales en Voltios, Amperios o grados deben dividirse los datos entre $1 * 10^4$.

Haciendo un análisis por columnas:

- **Timestamp:** Son las estampas de tiempo, es decir, el instante de tiempo en el cual fueron tomadas las medidas. Esto incluye año, mes, día y hora con una resolución de milésimas de segundo.
- **2016_MA5PA5_F91:Current A: Magnitude:** Este titulo contiene información sobre el PMU que realiza las mediciones (2016_MA5PA5_F91), en donde se especifica su identidad (F91) y el extremo del cual está midiendo (MA5PA5[2]). Luego, "Current A: Magnitude"significa que las medidas corresponden al módulo de la corriente de la fase A.
- **2016_MA5PA5_F91:Current A: Angle:** Es igual al item anterior con la salvedad de que "Current A: Angle:"significa que las medidas corresponden al ángulo de la corriente de la fase A.
- Las demás columnas se interpretan de igual manera que las anteriores.

5.2. Lenguaje de programación

El lenguaje de programación de propósito general, orientado a objetos, que se utiliza para el desarrollo de la aplicación es Python.[3] Se optó por este lenguaje por ser poderoso y muy práctico. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. Su sintaxis es fácil de entender puesto que es cercana al lenguaje natural, y los programas hechos en Python parecen pseudocódigos, lo cual brinda una gran ayuda en su mantenimiento. Esto hace que Python sea un lenguaje ideal para el desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas.

5.3. Diagrama de flujo

Se presentan a continuación los diagramas de flujo como representación gráfica de las distintas etapas de los algoritmos implementados en la aplicación, para los cálculos de impedancia, identificación de tipo de falla y localización de falla. Son

[2]MA5PA5 hace referencia a la línea Montevideo A-Palmar donde el extremo próximo es Montevideo A

[3]Es administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License.

útiles para comprender el funcionamiento de los algoritmos.

En la figura 5.3 se observa el diagrama correspondiente al cálculo de los parámetros de la línea. Se indica el proceso para determinar los parámetros de línea descritos en el punto 3.4. Se pueden observar todas las consideraciones que se deben tener en cuanta para el calculo de la impedancia, descritas en el punto 3.5.

Por otro lado, en la figura 5.4 se observa el diagrama correspondiente a la localización e identificación usando medidas desde un extremo de la línea. Este diagrama es la representación de los tres algoritmos de localización con medidas en un solo extremo propuestos.

Por último en la figura 5.5 se observa el diagrama correspondiente a la localización e identificación utilizando medidas desde dos extremos de la línea.

5.4. Pseudocodigo

A continuación se describen los pasos a seguir para identificar y localizar una falla utilizando lenguaje común mediante la aplicación ILFUS. Luego se transforman en instrucciones de programación pudiéndose observar en el código de la aplicación, descrito en el apéndice B

1. Cargar archivo en formato **.csv** con datos de la falla. Estos archivos son extraídos desde las PDCs.
2. Seleccionar la línea que contiene la falla. La línea seleccionada debe corresponderse con el archivo cargado.
3. Seleccionar el método de localización **un extremo** o **dos extremos**. El usuario debe considerar que algunas líneas cuentan con medidas desde un extremo y otras de ambos extremos.
4. Si se eligió la opción de un extremo se debe elegir desde cual de los dos extremos se desea localizar. Si la opción fue dos extremos se debe elegir si se desea calcular la impedancia de línea con las medidas o se utiliza la impedancia cargada para los algoritmos de localización.
5. Luego se debe correr la aplicación utilizando el botón **Correr**.

Una ves realizado los pasos mencionados, la aplicaron procesa la información y despliega en pantalla los resultados obtenidos. Esto es, la distancia al punto de falla y los valores de corrientes y tensiones de falla alcanzados.

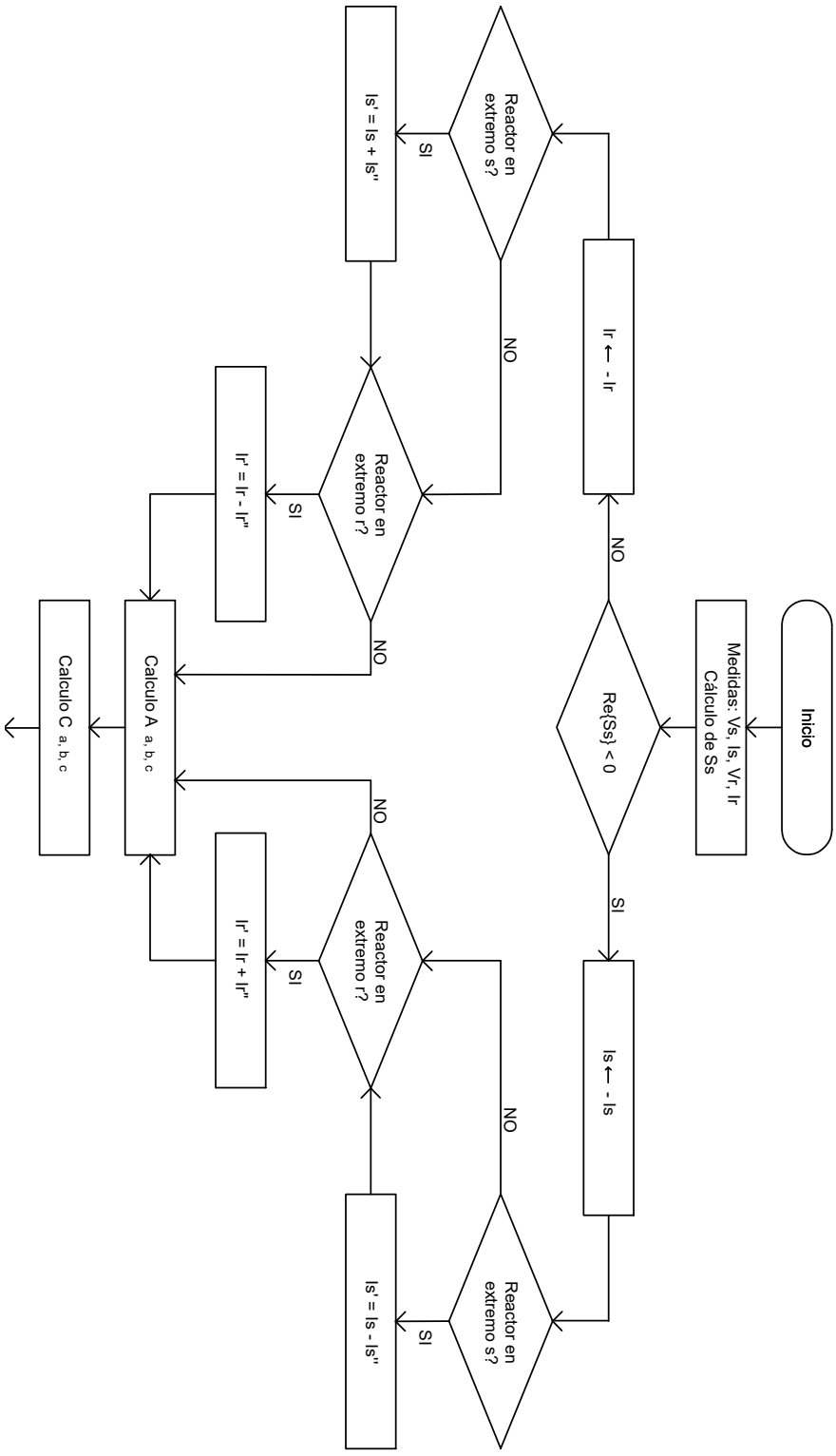


Figura 5.2: Cálculo de impedancia de línea (a)

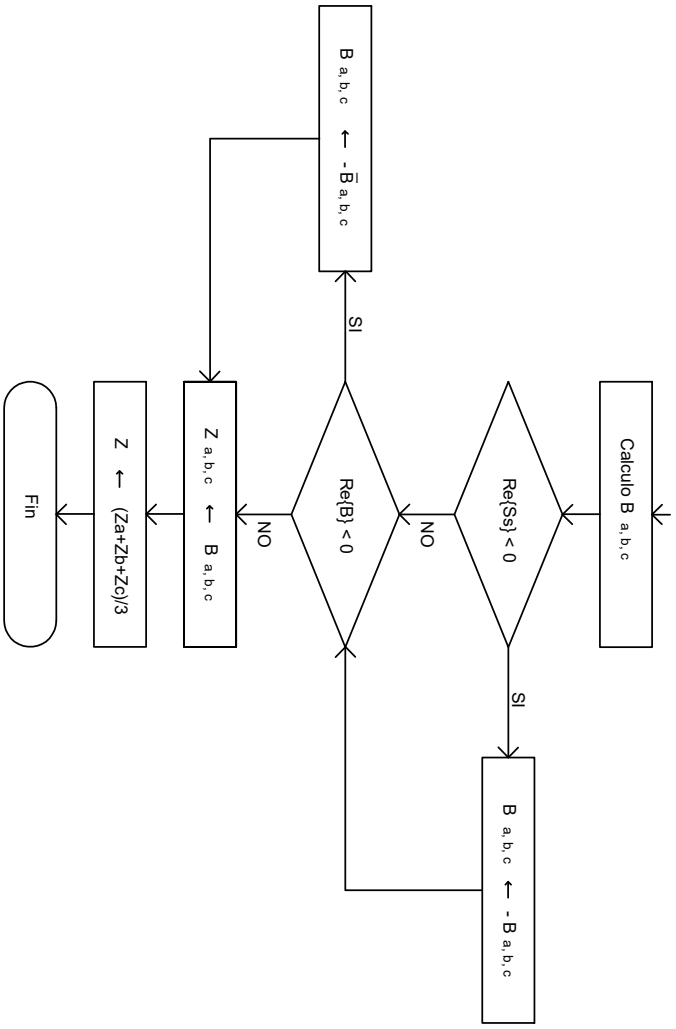


Figura 5.3: Cálculo de impedancia de línea (b)

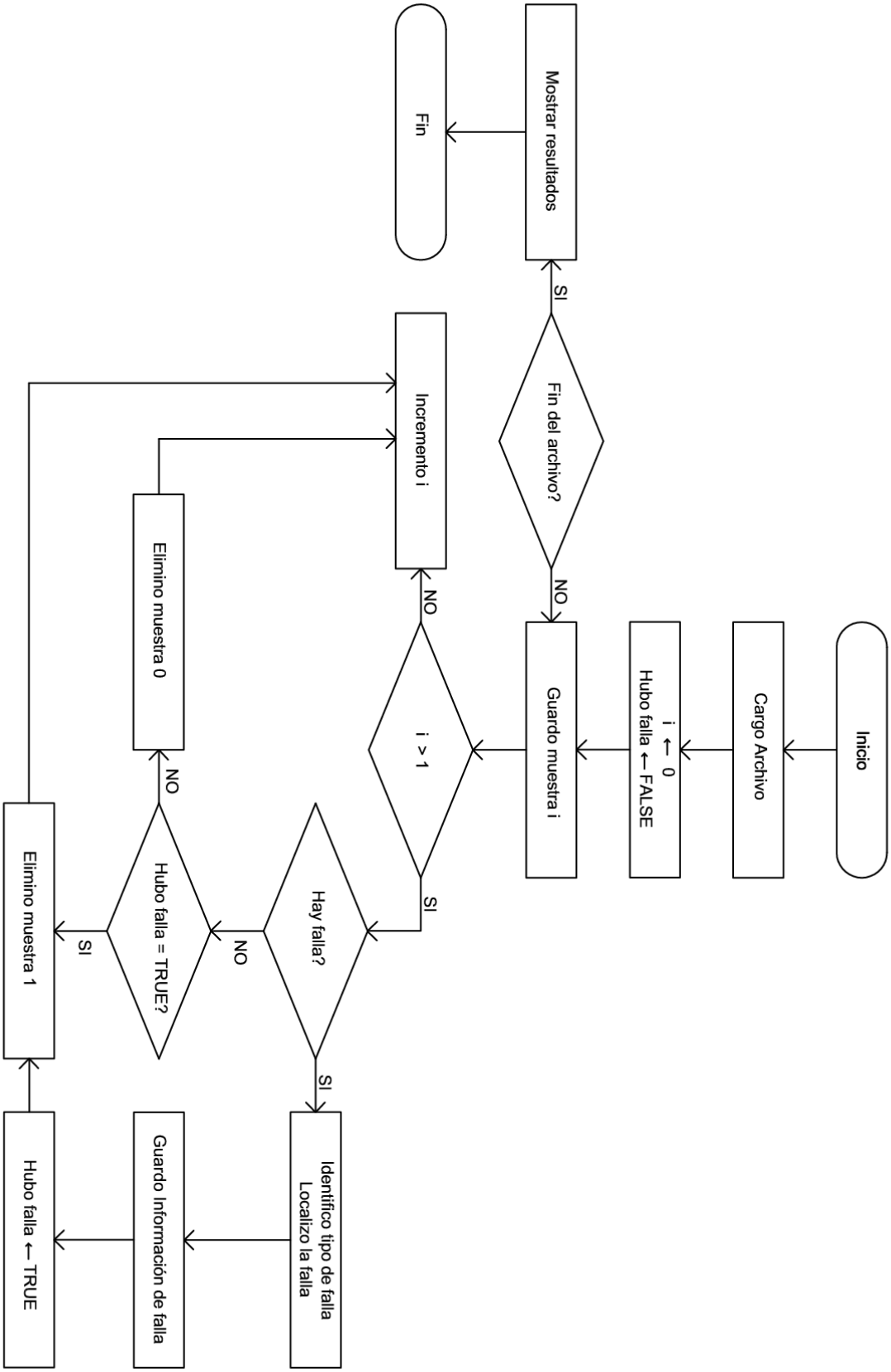


Figura 5.4: Medidas desde un extremo linea

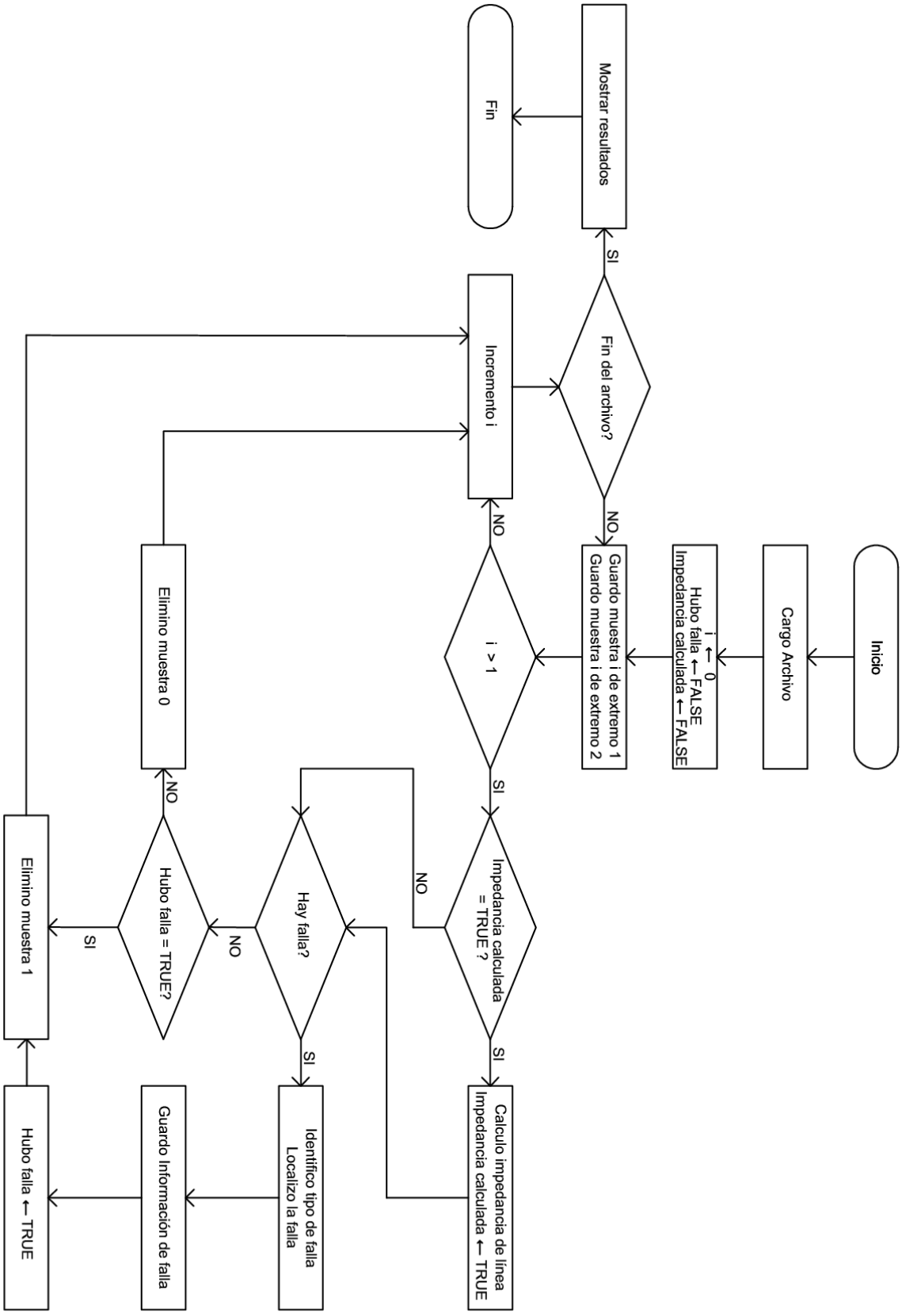


Figura 5.5: Medidas desde dos extremos

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 6

Plan de prueba de la aplicación

El plan de pruebas de software se elabora para atender los objetivos de calidad en un desarrollo de sistemas. Se han contrastado los resultados obtenidos por la aplicación usando bases de datos de fallas reales, con sus respectivas identificaciones y localizaciones. Estos resultados se mostraran con sus respectivos errores, verificando si cumplen los objetivos del proyecto.

6.1. Elaboración del plan de pruebas de la aplicación

Para garantizar el correcto funcionamiento de la aplicación y verificar el cumplimiento de los objetivos propuestos, se elabora un plan de pruebas que consiste en:

- Determinar los parámetros de las líneas de transmisión que posen medidas en sus dos extremos con archivos extraídos de las bases de datos (PDC) y comparar los resultados obtenidos con los proporcionados por UTE.
- Identificar y localizar faltas reales que ocurrieron en las líneas utilizando bases de datos reales.
- Para el caso de medidas en un solo extremo de la línea, la localización se realizará utilizando los tres algoritmos descritos en 4.4. Comparando sus resultados se determina cual es el más preciso.

6.1.1. Determinación de parámetros de la línea

Se determinaron los parámetros de 6 líneas de 500kV que poseen medidas fasoriales en sus dos extremos. Las líneas de $150kV$ tienen medidas solo en un extremo, por lo que no es posible determinar los parámetros de la línea.

En la tabla 6.1 se pueden observar los resultados obtenidos para las 6 líneas. Los valores de impedancias obtenidos son similares a los utilizados por UTE. Alcanza con comprobar que los valores son similares para asegurar un correcto com-

portamiento del algoritmo ya que ambas impedancias fueron determinadas por métodos diferentes.

Nombre	$Z_{UTE}(pu)$	$Z_{ILFUS}(pu)$	*Resultado*
MA5 - MB5	$0,00012 + j0,00109$	$0,00108 + j0,00129$	OK
MA5 - MI5	$0,00020 + j0,00213$	$0,00016 + j0,00243$	OK
MA5 - PA5	$0,00269 + j0,02523$	$0,00251 + j0,02538$	OK
BR5 - PA5	$0,00271 + j0,02252$	$0,00362 + j0,02249$	OK
MI5 - SC5	$0,00129 + j0,01359$	$0,0111 + j0,01335$	OK
ME5 - SC5	$0,00398 + J0,03308$	$0,00373 + j0,02953$	OK

Tabla 6.1: Determinación impedancia de línea

6.1.2. Identificación y localización

Se cuenta con 10 casos de fallas reales, se documentarán 5 de estos casos de fallas, 3 en líneas de 500kV y 2 en líneas de 150kV.
A continuación se describen tres de los casos de prueba y se analizan sus resultados. Los resultados y análisis de los demás casos se encuentran en el apéndice (Ver apéndice A).
Los casos a analizar son:

1. Falla en la línea de $500kV$ Montevideo A - Palmar (MA5-PA5), con medidas en sus dos extremos.
2. Falla en la línea de $500kV$ Palmar-San Javier (PA5 - SJ5), con medias en un extremo.
3. Falla en la línea 1 de $150kV$ Bonete-Pintado (BON-PIA L1), con medida en un extremo.

En los objetivos propuestos (ver sección 1.3) se indica que debe identificarse y localizarse la falla con un error relativo a la medida menor al 10 %. Por lo tanto, cada prueba será considerada satisfactoria si se identifica el tipo de falla descrito en cada caso y si se cumple con la condición del error.
El error relativo de la medida se determinara como:

$$\delta = \frac{|d_r - d_c| * 100}{d_r} \tag{6.1}$$

donde d_r es la distancia de la falla calculada por los rele (se asume como distancia real a la falla en este análisis) y d_c la distancia calculada.
En cada caso se muestran los valores de tensión y corriente de falla y pre-falla, incluyendo el momento exacto del comienzo de la falla. Se presentan tres muestras de pre-falla y en falla se presentan las muestras con valores de corriente en ascenso, hasta que se alcance el máximo valor de corriente de falla. A partir de este valor

se considera que se está despejando la falla por lo que no son valores útiles para los cálculos.

1. **Falla en la línea MA5-PA5**

   - Extremo local: Montevideo A - MA5.
   - Extremo remoto: Palmar - PA5
   - Tipo de falla: monofásica
   - Fase en falla: B
   - Largo de línea L: 228,9Km
   - Distancia d a la falla desde extremo local en Km: 64,9

N_o	$Time\ stamp$	$I_BMA:\ Mag(A)$	$I_BMA:\ Angle(deg)$	$V_BMA:\ Mag(kV)$	$V_BMA:\ Angle(deg)$
1	2017/08/26 03:47:42.020	$394,78$	$-92,56$	$295,88$	$89,69$
2	2017/08/26 03:47:42.040	$394,38$	$-92,89$	$295,88$	$89,40$
3	2017/08/26 03:47:42.060	$280,34$	$-69,45$	$287,19$	$89,52$
4	2017/08/26 03:47:42.080	$1219,64$	$9,77$	$226,16$	$89,19$
5	2017/08/26 03:47:42.100	$2747,39$	$12,59$	$138,48$	$86,62$
6	2017/08/26 03:47:42.120	$2937,20$	$7,25$	$92,85$	$83,35$

Tabla 6.2: Datos de falla en extremo local MA5

En la tabla 6.2 se indican los valores de tensión y corriente en régimen normal y en falla de la fase B, medidos por la PMU ubicada en el extremo local MA5. [1]. En este extremo se mide una corriente de carga en régimen normal de $394A$ y una tensión por fase de $295kV$ aproximadamente.
Mientras que en la tabla 6.3 se indican los valores medidos por la PMU, ubicada en el extremo remoto PA5. En este extremo se mide una corriente de carga en régimen normal de $419A$ y una tensión por fase de $293kV$ aproximadamente.

La falla fue localizada por el método de los dos extremos (ver 4.4.4) Se observa un aumento de la corriente y caída de la tensión a partir de la muestra

[1] A los efectos de simplificar la tabla no se muestran los valores de las fases que no están en falla

N_o	$Time\ stamp$	$I_BPA: Mag(A)$	$I_BPA: Angle(deg)$	$V_BPA: Mag(kV)$	$V_BPA: Angle(deg)$
1	2017/08/26 03:47:42.020	419,33	111,09	293,95	94,69
2	2017/08/26 03:47:42.040	419,43	110,83	293,94	94,40
3	2017/08/26 03:47:42.060	483,70	95,24	288,81	94,42
4	2017/08/26 03:47:42.080	1241,73	39,19	252,03	94,32
5	2017/08/26 03:47:42.100	2591,29	23,63	198,92	93,49
6	2017/08/26 03:47:42.120	2780,11	16,76	157,16	93,19

Tabla 6.3: Datos de falla en extremo local PA5

N_o	$Time\ stamp$	m	$d(km)$
4	2017/08/26 03:47:42.080	0,284	65,01
5	2017/08/26 03:47:42.100	0,297	67,98
6	2017/08/26 03:47:42.120	0,308	70,50

Tabla 6.4: Valores de m y d determinados por algoritmo dos extremos

4, confirmando la existencia de falla.

Con el valor de m determinado se calcula la distancia d a la que ocurre la falla expresada en Km como $m * L$, medida desde el extremo local MA5.

En la tabla 6.4 se muestran los resultados de la ubicación obtenidos.
Con los valores de d de la tabla 6.4 se determinan los errores relativos, siendo 64,9km el valor real de la distancia a la falla.

A continuación se muestra como ejemplo el cálculo del error de la muestra 4.

- Error en muestra 4:

$$\delta_4 = \frac{|64{,}9 - 65{,}01| * 100}{64{,}9} = 0{,}17\,\% \tag{6.2}$$

N_o	$Time\ stamp$	$e\,\%$
4	2017/08/26 03:47:42.080	0,17
5	2017/08/26 03:47:42.100	4,75
6	2017/08/26 03:47:42.120	8,62

Tabla 6.5: Calculo de errores

En la tabla 6.5 se observan los errores relativos en las medidas. Todas las muestras presentan errores menores al 10 % cumpliendo el algoritmo de dos extremos con los objetivos planteados. En particular en la muestra 4 se obtiene la mejor precisión en la medida.

2. **Falla en la línea PA5 - SJ5**

   - Extremo local: Palmar - PA5.
   - Extremo remoto: San Javier - SJ5
   - Tipo de falla: monofásica
   - Fase en falla: A
   - Largo de línea L: $80,2Km$
   - Distancia d a la falla desde extremo local en Km: $40,2$

   En la tabla 6.6 se indican los valores de tensión y corriente en régimen normal y en falla de la fase A, medidos por la PMU ubicada en el extremo local PA5.[2]. En este caso circula por la línea una corriente de carga en régimen normal de $420A$ y tiene una tensión por fase de $293kV$.

   La línea presenta medidas solo en el extremo de Palmar, por lo que la falla es localizada por el método de un extremo. Se observa un aumento de la corriente y caída de la tensión a partir de la muestra 4, confirmando la existencia de falla tal como se explica en los puntos 4.3 y 4.4.

   Con el valor de m determinado se calcula la distancia a la que ocurre la falla d expresada en Km como $m * L$, medida desde el extremo local PA5.

   En la tabla 6.7 se muestran los resultados de la ubicación obtenidos por los tres algoritmos (Reactancia, Takagi y Novosel).

[2]A los efectos de simplificar la tabla no se muestran los valores de las fases que no están en falla

N_o	$Time$ $stamp$	I_A : $Mag(A)$	I_A : $Arg(deg)$	V_A : $Mag(kV)$	V_A : $Arg(deg)$
1	2016/08/03 13:33:44.000	$424,20$	$-19,38$	$293,26$	$157,68$
2	2016/08/03 13:33:44.020	$423,37$	$-1,92$	$293,27$	$157,87$
3	2016/08/03 13:33:44.040	$574,71$	$55,53$	$268,55$	$158,87$
4	2016/08/03 13:33:44.060	$2677,67$	$79,20$	$169,77$	$158,40$
5	2016/08/03 13:33:44.080	$4907,53$	$78,91$	$63,48$	$148,72$
6	2016/08/03 13:33:44.100	$5422,40$	$76,76$	$51,04$	$139,15$
7	2016/08/03 13:33:44.120	$5628,50$	$77,52$	$88,17$	$150,5$
8	2016/08/03 13:33:44.140	$6449,28$	$79,14$	$97,43$	$153,80$
9	2016/08/03 13:33:44.160	$6615,39$	$78,84$	$83,64$	$152,80$

Tabla 6.6: Datos de falla medidos por extremo local PA5

Con los valores de d de la tabla 6.7 se determinan los errores relativos, siendo $40{,}2km$ el valor de la distancia a la falla calculado por el rele, asumido como distancia real en este análisis.

A continuación se muestra el cálculo del error de la muestra 9 determinada por el algoritmo de Takagi.

- Error en muestra 9:

$$\delta_9 = \frac{|40{,}2 - 43,59| * 100}{40{,}2} = 8,43\,\% \tag{6.3}$$

El algoritmo de la Reactancia es el que presenta mayor error en la determinación de la distancia a la falla. Mientras que Takagi y Novosel presentan errores similares. La muestra 5 es la que presenta menor error para ambos algoritmos, $7,2\,\%$ y $6,7\,\%$ respectivamente, verificando ambas la condición del $10\,\%$ propuesta en los objetivos.
Por otro lado en la muestra 9, se da el pico de corriente de falla con un error de $8,4\,\%$ y $8,6\,\%$ respectivamente,también verificando la condición de error máximo.
La muestra 4 presenta un error no aceptable en los tres casos. Esto puede explicarse por el hecho de que, esta medida se da en el instante que se esta

N_o	$Time$ $stamp$	m $Reactancia$	m $Takagi$	m $Novosel$	d $Reactancia$	d $Takagi$	d $Novosel$
4	2016/08/03 13:33:44.060	$1,373$	$2,781$	$2,775$	$\mathbf{1}10,115$	$223,027$	$111,561$
5	2016/08/03 13:33:44.080	$0,275$	$0,537$	$0,535$	$22,047$	$43,077$	$42,888$
6	2016/08/03 13:33:44.100	$0,198$	$0,365$	$0,363$	$15,896$	$29,2913$	$29,124$
7	2016/08/03 13:33:44.120	$0,385$	$0,668$	$0,666$	$30,909$	$53,539$	$53,444$
8	2016/08/03 13:33:44.140	$0,402$	$0,652$	$0,652$	$32,216$	$52,290$	$52,296$
9	2016/08/03 13:33:44.160	$0,333$	$0,544$	$0,544$	$26,747$	$43,590$	$43,659$

Tabla 6.7: Valores de m determinados por 3 algoritmos

N_o	$Time$ $stamp$	$e\,\%$ $Reactancia$	$e\,\%$ $Takagi$	$e\,\%$ $Novosel$
4	2016/08/03 13:33:44.060	$173,917$	$454,793$	$177,514$
5	2016/08/03 13:33:44.080	$45,157$	$7,158$	$6,686$
6	2016/08/03 13:33:44.100	$15,896$	$27,136$	$27,552$
7	2016/08/03 13:33:44.120	$23,111$	$33,181$	$32,944$
8	2016/08/03 13:33:44.140	$19,860$	$30,075$	$30,090$
9	2016/08/03 13:33:44.160	$33,466$	$8,432$	$8,605$

Tabla 6.8: Resultado en % del error en la ubicación de la falla

generando la falla, por lo tanto los valores de corriente y tensión no se deben considerar como datos de falla.

3. **Falla en la línea de $150kV$ BON-PIA L1**

   - Extremo local: Bonete - BON.
   - Extremo remoto: Pintado - PIA
   - Tipo de falla: monofásica
   - Fase en falla: A
   - Largo de línea L: $118Km$

- Distancia d a la falla desde extremo local en Km: $80,5Km$

N_o	$Time$ $stamp$	I_A : $Mag(A)$	I_A : $Arg(deg)$	V_A : $Mag(kV)$	V_A : $Arg(deg)$
1	2017/07/09 08:18:09.560	$152,36$	$-4,00$	$90,70$	$-54,81$
2	2017/07/09 08:18:09.580	$166,93$	$-5,11$	$90,22$	$-54,84$
3	2017/07/09 08:18:09.600	$462,13$	$-10,55$	$83,15$	$-54,62$
4	2017/07/09 08:18:09.620	$1099,69$	$-119,80$	$70,92$	$-54,77$
5	2017/07/09 08:18:09.640	$1446,81$	$-123,87$	$64,19$	$-56,03$

Tabla 6.9: Datos de falla medidos por extremo local BON

En la tabla 6.9 se indican los valores de tensión y corriente en régimen normal y en falla de la fase A, medidos por la PMU ubicada en el extremo local BON.[3]. En este caso circula por la línea una corriente de carga en régimen normal de $155A$ y tiene una tensión por fase de $90kV$.

La línea presenta medidas solo en el extremo de Bonete (BON), por lo tanto la falla es localizada por el método de un extremo. Se observa un aumento de la corriente y caída de la tensión a partir de la muestra 4 confirmando la existencia de falla, tal como se explica en los puntos 4.3 y 4.4.

Con el valor de m determinado se calcula la distancia a la que ocurre la falla d expresada en Km como $m * L$, medida desde el extremo local BON.
En la tabla 6.10 se muestran los resultados de la ubicación obtenidos por los tres algoritmos (Reactancia, Takagi y Novosel).

N_o	$Time$ $stamp$	m $Reactancia$	m $Takagi$	m $Novosel$	d $Reactancia$	d $Takagi$	d $Novosel$
4	2017/07/09 08:18:09.620	$0,896$	$0,986$	$0,985$	$105,681$	$116,396$	$116,242$
5	2017/07/09 08:18:09.640	$0,628$	$0,684$	$0,683$	$74,057$	$80,743$	$80,629$

Tabla 6.10: Valores de m determinados por 3 algoritmos

[3]A los efectos de simplificar la tabla no se muestran los valores de las fases que no están en falla

Con los valores de d de la tabla 6.10 se determinan los errores relativos, siendo $80,5km$ el valor real de la distancia a la falla.

A continuación se muestra el cálculo del error de la muestra 5 determinada por el algoritmo de Takagi.

- Error en muestra 5:

$$\delta_5 = \frac{|80,5 - 80,743| * 100}{80,5} = 0,302\,\% \tag{6.4}$$

N_o	$Time$ $stamp$	$e\,\%$ $Reactancia$	$e\,\%$ $Takagi$	$e\,\%$ $Novosel$
4	2017/07/09 08:18:09.620	$31,280$	$44,592$	$44,400$
5	2017/07/09 08:18:09.640	$8,004$	$0,302$	$0,161$

Tabla 6.11: Resultado en % del error en la ubicación de la falla

En este caso en la muestra 5 se da el pico de corriente de falla, siendo esta muestra la que presenta menor error en los 3 algoritmos. Los algoritmos de Takagi y Novosel son mas precisos que el de la Reactancia, en particular presentan un error menor al 1 %.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 7

Conclusiones y trabajos futuros

Ubicar de forma precisa una falla en una línea de transmisión es de suma importancia para mejorar el rendimiento y reducir los costos de operación y mantenimiento de la misma. Por estas razones se hace necesaria la implementación de una aplicación capaz de localizar las fallas de manera eficiente. Actualmente, UTE cuenta con equipos de protección y medida sincronizados instalados en varios puntos de la red. Estos equipos están relevando información a tiempo real y almacenando la misma para un uso posterior.

El presente proyecto se basó en el desarrollo de una aplicación para identificar y localizar fallas en líneas de transmisión utilizando información de los sincrofasores recogidas en la base de datos. Se debió distinguir los sincrofasores que detectan la falla, y con esa información identificar el tipo de falla y la distancia a la misma desde uno de los extremos.

En la actualidad no se cuenta con medidas de sincrofasores en ambos extremos para todas las líneas, por lo cual se debió estudiar dos casos particulares en cuanto a la obtención de datos, desde un solo extremo de la línea o en ambos. Dependiendo de cada caso se procesó la información utilizando distintos algoritmos.

En resumen, los objetivos de este proyecto consistieron en desarrollar una aplicación que identifique los distintos tipos de fallas que ocurran en las líneas y las localice determinando la distancia a la que ocurren las fallas medidas desde un extremo de la línea. Para ello utiliza información de sincrofasores recogida de una base de datos. Esta herramienta además, cuando se cuente con medidas desde ambos extremos de la línea, será capaz de determinar los parámetros de la misma y utilizarlos para mejorar la precisión en la ubicación de la falla.

7.1. Conclusiones

- Para lograr los objetivos propuestos, primero se estudió el modelo de una línea de transmisión como cuadripolo pasivo y luego se desarrolló un algo-

ritmo para determinar sus parámetros, únicamente para los casos que se posean medidas en ambos extremos de la línea.
Para realizar una correcta implementación del algoritmo, se debió tener en cuenta los sentidos del flujo de carga, realizando correcciones en las medidas de corriente de uno de los extremos de la línea, con el objetivo de adecuar el sentido de la corriente al modelo planteado. Se tuvieron en cuenta los efectos generados por Efecto Ferranti en la líneas de transmisión y para el caso de líneas que posen reactores, se consideraron las corrientes aportadas por éstos.
Una vez implementado el algoritmo para determinar los parámetros de la línea en la aplicación, se realizaron pruebas con archivos extraídos de las bases de datos de líneas con medidas en sus dos extremos, obteniendo resultados correctos en todos los casos. Estos resultados fueron comparados con los valores de impedancias utilizados por UTE, siendo estos valores similares.

- Para identificar los distintos tipos de fallas que ocurren en las líneas, se implementó un algoritmo, el cual consiste en encontrar un aumento de la corriente entre medidas consecutivas acompañado de una caída de tensión en las fases en falla.
  Se realizaron pruebas de este algoritmo con casos de fallas reales y se logró identificar las fallas correctamente en todos los casos.

- Se estudiaron tres métodos de localización de fallas en líneas con medidas en un solo extremo, método de la Reactancia Simple, Takagi y Novosel.
  El algoritmo de la Reactancia hace la aproximación de que la resistencia de falla es cero, eliminando el termino que contiene la caída de tensión en el punto de la falla, provocando un error cuando dicha resistencia no es cero.
  El algoritmo de Takagi elimina el termino de la caída de tensión en el punto de la falla multiplicándolo por una magnitud tal que el resultado sea real y finalmente considera el sistema homogéneo, provocando un error cuando el sistema no lo es.
  Finalmente el algoritmo de Novosel hace las mismas aproximaciones pero en las redes de secuencia positiva y negativa.
  Se realizaron pruebas de los tres métodos con casos de fallas reales (ver 6.1.2).
  En base a los resultados obtenidos se puede concluir que el algoritmo de la Reactancia fue el que presento el mayor porcentaje de error en la mayoría de los casos de prueba, mientras que los algoritmos de Takagi y Novosel mostraron porcentajes de errores similares menores a los de la Reactancia.
  En la mayoría de los casos analizados el algoritmo de Takagi fue el más preciso, aunque el de Novosel también cumplía con el objetivo.
  Por lo tanto el algoritmo de Takagi es el que mejor se adecúa al cálculo de la ubicación de la falla para líneas con medidas en un solo extremo, siendo éste el que utilizará la aplicación.

- El método más preciso para localizar fallas en una línea de transmisión es

el que utiliza medidas de los dos extremos de la línea. Ésto se debe a que no considera la impedancia de falta. Se implemento el algoritmo para la localización y se probó con casos de falla reales.
Se cuentan con dos casos reales de fallas en líneas de $500kV$. En ambos casos se logró localizar el punto donde ocurre la falla con éxito, logrando obtener un error en la medida menor al 10 %.

- En general, para líneas con medidas en uno y dos extremos, las muestras donde se alcanza el pico de corriente de falla, son las que presentan el menor porcentaje de error en la medida. Por lo tanto, este es el valor más adecuado para indicar la distancia al punto de la falla.

- En la etapa de procesamiento de datos para identificar y localizar fallas, algunas muestras deben ser descartadas en el análisis ya que los valores de tensión y corriente en esas muestras no representan exactamente la falla, por ejemplo la muestras correspondiente al instante en que comienza la falla y el momento en que abre el equipo de protección en los extremos de la linea ya que la corriente de falla comienza a disminuir. Utilizando estas muestras para localizar e identificar

- Se creó una interfaz que utiliza los algoritmos antes mencionados y procesa los datos de los sincrofasores, los cuales se extraen de la base de datos de UTE. Dicha interfaz cuenta con una lista de líneas de transmisión de UTE y sus características ya precargadas. También tiene la funcionalidad de modificar los parámetros, agregar nuevas líneas y eliminar líneas. La aplicación está escrita en el lenguaje Python, el cual puede ejecutarse en varios sistemas operativos. En este caso, se realizó un ejecutable para que la aplicación se utilice en Windows. El código de la aplicación se encuentra en el Apéndice B de éste documento (B).

7.2. Recomendaciones para trabajos futuros

En base a los resultados obtenidos en este proyecto, se recomienda para trabajos futuros:

1. Estudio de nuevos algoritmos para localizar las fallas que puedan mejorar la precisión de la aplicación.
2. Estudio de modelos más complejos para modelado de las líneas de trasmisión, lo cual mejoraría la precisión en la localización.
3. Se comparó El tiempo de procesamiento de datos de dos extremos de la aplicación es de $0{,}5ms$, siendo inferior al tiempo de generación de los sincrofasores por parte de las PMU.Por tal motivo, se podría agregar una nueva funcionalidad a la aplicación de manera que sea capaz de procesar datos en tiempo real y procesar datos de varias líneas a la vez.

4. Mejorar la interfaz gráfica, de manera que sea capaz de crear una versión de impresión de los resultados.

Apéndice A

Análisis de casos de prueba

1. **Falla en la línea ME5-SC5**

   - Extremo local: Melo - ME5.
   - Extremo remoto: San Carlos - SC5
   - Tipo de falla: monofásica
   - Fase en falla: B
   - Largo de línea L: $300Km$
   - Distancia d a la falla desde extremo local en Km: 300

N_o	$Time\ stamp$	$I_BME: Mag(A)$	$I_BME: Angle(deg)$	$V_BME: Mag(kV)$	$V_BME: Angle(deg)$
1	2016/12/23 03:00:04.700	$87,58$	$105,6$	$295,39$	$-0,75$
2	2016/12/23 03:00:04.720	$87,58$	$105,51$	$295,39$	$-0,77$
3	2016/12/23 03:00:04.740	$87,58$	$105,51$	$295,39$	$-0,76$
4	2016/12/23 03:00:04.760	$19,01$	$79,57$	$251,72$	$10,31$
5	2016/12/23 03:00:04.780	$158,51$	$-72,02$	$140,35$	$0,027$
6	2016/12/23 03:00:04.800	$297,79$	$-76,66$	$44,08$	$-5,40$

Tabla A.1: Datos de falla en extremo local ME5

En la tabla A.1 se indican los valores de tensión y corriente en régimen normal y en falla de la fase B, medidos por la PMU ubicada en el extremo local

N_o	$Time$ $stamp$	$I_BSC:$ $Mag(A)$	$I_BSC:$ $Angle(deg)$	$V_BSC:$ $Mag(kV)$	$V_BSC:$ $Angle(deg)$
1	2016/12/23 03:00:04.700	$37,81$	$47,81$	$298,03$	$-0,48$
2	2016/12/23 03:00:04.720	$37,81$	$47,88$	$298,00$	$-0,49$
3	2016/12/23 03:00:04.740	$38,02$	$47,65$	$299,13$	$-0,49$
4	2016/12/23 03:00:04.760	$779,29$	$-68,67$	$239,41$	$1,54$
5	2016/12/23 03:00:04.780	$2721,73$	$-76,35$	$94,25$	$6,80$
6	2016/12/23 03:00:04.800	$4130,77$	$-80,09$	$29,34$	$172,44$

Tabla A.2: Datos de falla en extremo remoto SC5

N_o	$Time$ $stamp$	m	$d(km)$
4	2016/12/23 03:00:04.760	$0,99$	$298,14$
5	2016/12/23 03:00:04.780	$1,09$	$328,82$
6	2016/12/23 03:00:04.800	$1,35$	$406,41$

Tabla A.3: Valores de m y d determinados por algoritmo dos extremos

ME5. [1]. En este extremo se mide una corriente de carga en régimen normal de $87A$ y una tensión por fase de $295kV$ aproximadamente.
Mientras que en la tabla A.2 se indican los valores medidos por la PMU, ubicada en el extremo remoto SC5. En este extremo se mide una corriente de carga en régimen normal de $37A$ y una tensión por fase de $298kV$ aproximadamente.

La falla fue localizada por el método de los dos extremos (ver 4.4.4) Se observa un aumento de la corriente y caída de la tensión a partir de la muestra 4, confirmando la existencia de falla.

Con el valor de m determinado se calcula la distancia d a la que ocurre la falla expresada en Km como $m * L$, medida desde el extremo local ME5.

[1] A los efectos de simplificar la tabla no se muestran los valores de las fases que no están en falla

En la tabla A.3 se muestran los resultados de la ubicación obtenidos.
Con los valores de d de la tabla A.3 se determinan los errores relativos, siendo $300km$ el valor real de la distancia a la falla.

A continuación se muestra como ejemplo el cálculo del error de la muestra 5.

- Error en muestra 5:

$$\delta_5 = \frac{|300 - 328{,}82| * 100}{300} = 9{,}61\,\% \tag{A.1}$$

N_o	$Time$ $stamp$	$e\,\%$
4	2016/12/23 03:00:04.760	0,62
5	2016/12/23 03:00:04.780	9,61
6	2016/12/23 03:00:04.800	35

Tabla A.4: Calculo de errores

En la tabla A.4 se observan los errores relativos en las medidas. Las muestras 4 y 5 presentan errores menores al 10 % cumpliendo el algoritmo de dos extremos con los objetivos planteados. En particular en la muestra 4 se obtiene la mejor precisión en la medida.
Mientras que la muestra 6 presenta un error del 35 %, siendo este un valor no aceptado. Esto puede explicarse por el hecho de que, ante una línea larga ($300km$), los algoritmos propuestos son menos precisos.

2. Falla en la línea MA5 - PA5

   - Extremo local: Montevideo A - MA5.
   - Extremo remoto: Palmar - PA5
   - Tipo de falla: monofásica
   - Fase en falla: C
   - Largo de línea: $228,9Km$
   - Distancia d a la falla desde extremo local en Km: 0
   - Distancia d a la falla desde extremo remoto en Km: $228,9$

N_o	$Time$ $stamp$	I_C : $Mag(A)$	I_C : $Angle(deg)$	V_C : $Mag(kV)$	V_C : $Angle(deg)$
1	2015/12/14 12:56:57.340	$446,32$	$-91,18$	$298,52$	$82,38$
2	2015/12/14 12:56:57.360	$443,45$	$-89,05$	$299,61$	$83,15$
3	2015/12/14 12:56:57.380	$372,97$	$-7,70$	$295,04$	$84,26$
4	2015/12/14 12:56:57.400	$1963,42$	$4,36$	$215,94$	$86,46$
5	2015/12/14 12:56:57.420	$5469,42$	$6,28$	$63,72$	$90,97$
6	2015/12/14 12:56:57.440	$6764,02$	$1,82$	$34,81$	$-86,92$

Tabla A.5: Datos de falla en extremo remoto MA5

N_o	$Time$ $stamp$	m $Reactancia$	m $Takagi$	m $Novosel$	d $Reactancia$	d $Takagi$	d $Novosel$
4	2015/12/14 12:56:57.400	$0,824$	$1,714$	$1,445$	$188,54$	$392,4$	$330,69$
5	2015/12/14 12:56:57.420	$0,089$	$0,184$	$0,191$	$20,46$	$42,14$	$43,65$
6	2015/12/14 12:56:57.440	$-0,039$	$-0,082$	$-0,108$	$-8,88$	$-18,84$	$-24,61$

Tabla A.6: Valores de m determinados por 3 algoritmos

La prueba es satisfactoria si la aplicación logra identificar la falla fase tierra (C-G) y determina una distancia a la falla de $0Km$ desde el extremo local, con un error máximo del $10\,\%$

En la tabla A.5 se indican los valores de tensión y corriente en régimen normal y en falla de la fase C, medidos por la PMU ubicada en el extremo local MA5[2]. En este caso por la línea circula una corriente de carga en régimen normal de $446A$ y tiene una tensión por fase de $298kV$.

La falla fue localizada por el método de un extremo (ver 4.4.4). Si bien esta línea presenta medidas en sus dos extremos en este caso por causas que se desconocen no hay registro de medidas tomadas por la PMU ubicada en el extremo remoto. La aplicación maneja este caso como una excepción, cuan-

[2]A los efectos de simplificar la tabla no se muestran los valores de las fases que no están en falla

do identifica que el archivo .csv exportado de la base de datos, corresponde a una linea con medidas en sus dos extremos y no tiene registro de uno de sus dos extremo, solicita al usuario que se utilice el método de un extremo.

Se observa un aumento de la corriente y caída de la tensión a partir de la muestra 4, confirmando la existencia de falla.
Con el valor de m determinado se calcula la distancia d a la que ocurre la falla expresada en Km como $m * L$, medida desde el extremo local MA5.
En la tabla A.6 se muestran los resultados de la ubicación obtenidos por los tres algoritmos (Reactancia, Takagi y Novosel).
Con los valores de d se estiman los errores relativos. Dado que la distancia real a la falla son $0km$, para estimar el error en este caso se utilizara la distancia a la falla tomando como referencia el extremo remoto, quedando $d_c = 228,9km - d$ con $d_r = 228,9km$.

El error relativo de la medida se determina como:

$$\delta = \frac{|d_r - d_c| * 100}{d_r} \tag{A.2}$$

donde d_r es la distancia real de la falla y d_c la distancia calculada.
A continuación se muestra el cálculo del error de la muestra 6, del algoritmo de Takagi.

- error en muestra 6:

$$\delta_6 = \frac{|228,90 - 247,78| * 100}{228,90} = 8,23\,\% \tag{A.3}$$

N_o	$Time\ stamp$	$e\,\%\ Reactancia$	$e\,\%\ Takagi$	$e\,\%\ Novosel$
4	2015/12/14 12:56:57.400	82,37	171,43	144,47
5	2015/12/14 12:56:57.420	8,94	18,41	19,07
6	2015/12/14 12:56:57.440	3,88	8,23	10,75

Tabla A.7: Cálculo de errores

En este caso en la muestra 6 se da el pico de corriente de falla, siendo esta la que presenta menor error en los 3 algoritmos. En este caso el algoritmo de la Reactancia es el que presenta menor error de los tres analizados seguido por el de Takagi. Mientras que Novosel presenta errores superiores al 10 % en sus tres medidas.

3. Falla en la línea de $150kV$ BON-PIA L1

- Extremo local: Bonete - BON.
- Extremo remoto: Pintado - PIA
- Tipo de falla: monofásica
- Fase en falla: B
- Largo de linea: $118Km$
- Distancia d a la falla desde extremo local en Km: $17,6Km$

N_o	$Time$ $stamp$	I_A : $Mag(A)$	I_A : $Arg(deg)$	V_A : $Mag(kV)$	V_A : $Arg(deg)$
1	2017/07/10 07:52:58.600	$216,018$	$3,104$	$89,696$	$23,805$
2	2017/07/10 07:52:58.620	$216,098$	$0,003$	$89,698$	$23,402$
3	2017/07/10 07:52:58.640	$274,659$	$0,001$	$88,193$	$23,259$
4	2017/07/10 07:52:58.660	$1473,24$	$-0,004$	$73,83$	$23,30$
5	2017/07/10 07:52:58.680	$3407,964$	$-0,004$	$51,033$	$22,151$
6	2017/07/10 07:52:58.700	$4312,19$	$-0,005$	$40,26$	$18,99$

Tabla A.8: Datos de falla medidos por extremo local BON

En la tabla A.8 se indican los valores de tensión y corriente en régimen normal y en falla de la fase B, medidos por la PMU ubicada en el extremo local BON.[3]. En este caso circula por la línea una corriente de carga en régimen normal de $216A$ y tiene una tensión por fase de $89,6kV$.

La línea presenta medidas solo en el extremo local BON, entonces la falla es localizada por el método de un extremo. Se observa un aumento de la corriente y caída de la tensión a partir de la muestra 4 confirmando la existencia de falla, tal como se explica en 4.4 y 4.3.

Con el valor de m determinado se calcula la distancia d a la que ocurre la falla expresada en Km como $m * L$, medida desde el extremo local BON.
En la tabla A.9 se muestran los resultados de la ubicación obtenidos por los tres algoritmos (Reactancia, Takagi y Novosel).

[3]A los efectos de simplificar la tabla no se muestran los valores de las fases que no están en falla

N_o	$Time$ $stamp$	m $Reactancia$	m $Takagi$	m $Novosel$	$d(Km)$ $Reactancia$	$d(Km)$ $Takagi$	$d(Km)$ $Novosel$
5	2017/07/09 08:18:09.660	$0,651$	$0,743$	$1,051$	$76,783$	$87,624$	$124,018$
5	2017/07/09 08:18:09.680	$0,204$	$0,228$	$0,226$	$24,013$	$26,923$	$32,391$
6	2017/07/09 08:18:09.700	$0,128$	$0,143$	$0,142$	$15,069$	$16,874$	$19,800$

Tabla A.9: Valores de m determinados por 3 algoritmos

Con los valores de d de la tabla A.9 se determinan los errores relativos, siendo $17,6km$ el valor real de la distancia a la falla.

El error relativo de la medida se determina como:

$$\delta = \frac{|d_r - d_c| * 100}{d_r} \tag{A.4}$$

donde d_r es la distancia real de la falla y d_c la distancia calculada.

A continuación se muestra el cálculo del error de la muestra 6.

- error en muestra 6:

$$\delta_6 = \frac{|17,600 - 16,874| * 100}{17,600} = 4,125\,\% \tag{A.5}$$

N_o	$Time$ $stamp$	$e\,\%$ $Reactancia$	$e\,\%$ $Takagi$	$e\,\%$ $Novosel$
5	2017/07/09 08:18:09.660	$336,235$	$397,866$	$604,648$
5	2017/07/09 08:18:09.680	$36,438$	$52,969$	$84,040$
6	2017/07/09 08:18:09.700	$14,383$	$4,125$	$12,502$

Tabla A.10: Resultado en % del error

El algoritmo de la Reactancia es el que presenta mayor error en la determinación de la distancia a la falla.
Los algoritmos de Takagi y Novosel presentan errores similares. En particular en la muestra 6, se de el pico de corriente de falla y los errores son menores al 5 % en ambos casos.
La muestra 4 presenta un error del 400 %, siendo un valor no aceptado. De la misma manera que en los casos ya explicados, esta medida se da en el

instante que se esta generando la falla, por lo tanto los valores de corriente y tensión no se deben considerar como datos de falla.

Apéndice B

Código de la aplicación

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from _overlapped import NULL
from funcion_dos_extremos import *
from funcion_un_extremo import *
from funciones import *
from manejo_lista import *
import manejo_lista
import math
import sys
import threading
import time
from tkinter import *
from tkinter import messagebox  # para mensajes emergentes
from tkinter import ttk
from tkinter.filedialog import askopenfilename  # para abrir un explorador de archivos

import matplotlib
from matplotlib.backend_bases import key_press_handler
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2TkAgg
from matplotlib.figure import Figure
import matplotlib.pyplot
import numpy
import pandas

import cmath as cm
import tkinter.scrolledtext as tks


matplotlib.use('TkAgg')


class Aplicacion():
    def __init__(self):

#----------------------------------------VENTANA PRINCIPAL ----------------------------------------


        '''defino ventana raíz'''
        self.raiz = Tk()
        self.style = ttk.Style()
        self.style.theme_use("vista")
        '''('winnative', 'clam', 'alt', 'default', 'classic', 'vista', 'xpnative')'''
        self.raiz.geometry('500x400')
        self.raiz.resizable(width = FALSE, height = FALSE)
        self.raiz.title('ILFUS')

        ''' defino variables de control'''

        self.lista = manejo_lista.cargar() #  lista con todas las lineas
        self.lista_nomb = self.cargar_lista_nomb(self.lista)   #LISTA CON los nombre delos extremos TODAS LAS LINEAS
        self.impedancia_calc = 0
        self.var = IntVar()
        self.var_2 = IntVar()
        self.var_3 = IntVar()
        self.progreso = IntVar()
        self.primera_vez = False
        self.boton_para_extremos = False
        self.botones_para_impedancia = False
        self.listvar = StringVar()
        self.edit_stringvar = StringVar()
        self.elim_stringvar = StringVar()
        self.cargar_archivo = False

        ''' defino botones '''
```

```
ttk.Button(self.raiz, text='Salir',command = self.close_window).place(x=75,y=350)
ttk.Button(self.raiz, text= 'Correr', command = self.correr).place(x=350,y=350)
ttk.Button(self.raiz, text= 'Cargar .csv', command= self.cargar).place(x=100,y=50)

'''defino Text para mostrar texto de csv'''

self.dire = Text(self.raiz, width=35, height=2,font='helvetica 8 ')
self.dire.config(state=DISABLED)
self.dire.place(x=165,y=82)

''' defino cuadro para mostrar lista de lineass'''

self.Texto = Text(self.raiz, width = 45,height = 10, font='helvetica 8')
self.Texto.place(x=110,y=190)
self.Texto.config(state= DISABLED)

''' defino etiquetas '''

self.etiq1 = ttk.Label(self.raiz, text="Línea").place(x=100,y=120)
self.etiq1 = ttk.Label(self.raiz, text="Directorio:").place(x=100,y=82)

'''defino Combobox'''

self.listatex = ttk.Combobox(self.raiz, values =self.lista_nomb , textvariable = self.listvar , state = 'readonly')
self.listatex.place(x=100, y= 140)
self.listvar.set('Seleccionar línea')
self.listatex.bind('<<ComboboxSelected>>', self.mostrar_linea)

'''defino RADIOBUTTONS'''

self.r1 = Radiobutton(self.raiz , text="1 extremo", variable=self.var , value = 1,command= self.boton_un_extremo)
self.r1.place(x=260,y=130)
self.r2 = Radiobutton(self.raiz , text="2 extremos", variable=self.var , value = 2,command= self.boton_dos_extremos)
self.r2.place(x=260,y=160)
self.var.set(0)

''' Crear el menu principal'''
menubarra = Menu(self.raiz)

''' Crea un menu desplegable ARCHIVO'''
menuarchivo = Menu(menubarra, tearoff=0)
menuarchivo.add_command(label="Cargar archivo de línea", command=self.cargar)
menuarchivo.add_separator()
menuarchivo.add_command(label="Salir", command = self.close_window)
menubarra.add_cascade(label="Archivo", menu=menuarchivo)

''' Crea un menu desplegable EDICION '''
menueditar = Menu(menubarra, tearoff=0)
menueditar.add_command(label="Editar línea", command = self.edit_linea)
menueditar.add_command(label="Agregar línea", command = self.agreg_linea)
menueditar.add_command(label="Eliminar línea", command = self.elim_linea)
menubarra.add_cascade(label="Editar", menu=menueditar)

''' Crea un menu desplegable AYUDA'''
menuayuda = Menu(menubarra, tearoff=0)
menuayuda.add_command(label="Acerca de...", command=self.acerca_de)
menubarra.add_cascade(label="Ayuda", menu=menuayuda)

''' Mostrar el menu '''
self.raiz.config(menu=menubarra)

''' inicio el loop'''
self.raiz.mainloop()

''' defino FUNCIONES'''
```

```
    def cargar_lista_nomb(self,lista):
        lista_nomb= []
        for i in range (0,len(self.lista)):
            lista_nomb.append(lista[i].string_nombre_linea())
        return lista_nomb

    def acerca_de(self):
        self.ayuda = Toplevel()
        self.ayuda.geometry('400x270')
        texto = '\n Proyecto de grado \n Universidad de la Republica\n Facultad de Ingenieria \nIIE \n\n ILFUS 1.0 \n Identificar y Localizar Fallas
Usando Sincrofasores \n\n Autores: \n Sandino Silva \n Agustin Olivera \n Felipe Ohaco Irabedra \n\nTutor:\n Celia Sena'
        self.eti = Label(self.ayuda, text= texto)
        self.eti.pack()

    def cargar(self):
        self.dir_archivo = askopenfilename()
        if (self.dir_archivo == ''):
            self.cargar_archivo = False
        else:
            self.dire.config(state=NORMAL)
            self.dire.delete(0.0,END)
            self.dire.insert(0.0,self.dir_archivo)
            self.dire.config(state=DISABLED)
            self.cargar_archivo = True

    def imprimir(self):
        print(self.var.get())

    def guardar(self):
        '''aca tiene q agregar a la lista de líneas el dato de la nueva línea o la modificada'''
        print(self.listvar)

    def boton_un_extremo(self):
        if self.boton_para_extremos:
            self.r5.forget()
            self.r6.forget()
            self.var_3.set(0)
            self.boton_para_extremos = False

        if self.var.get() == 1:
            self.r3 = Radiobutton(self.raiz , text="Extremo local", variable=self.var_2 , value = 1)
            self.r3.place(x=350,y=130)
            self.r4 = Radiobutton(self.raiz , text="Extremo remoto", variable=self.var_2 , value = 2)
            self.r4.place(x=350,y=160)
            self.var_2.set(0)
            self.botones_para_impedancia = True

    def boton_dos_extremos(self):
        if self.botones_para_impedancia:
                self.r3.place_forget()
                self.r4.place_forget()
                self.var_2.set(0)
                self.botones_para_impedancia = False
        if self.var.get() == 2  :
            self.var_3.set(0)
            self.r5 = Radiobutton(self.raiz , text="Calcular Z", variable=self.var_3 , value = 1)
            self.r5.place(x=350,y=130)
            self.r6 = Radiobutton(self.raiz , text="Usar Z de UTE", variable=self.var_3 , value = 2)
            self.r6.place(x=350,y=160)
            self.boton_para_extremos = True

    def mostrar_linea (self,event):
        numero = self.listatex.current()
        self.Texto.config(state = NORMAL)
        self.Texto.delete(0.0,END)
        datos = str(self.lista[numero].imprimir_en_cuadro_text())
```

```
        self.Texto.insert(INSERT, datos )
        self.Texto.config(state = DISABLED)

#----------------------------------------VENTANA PARA PROGRESSBAR  ----------------------------------------------------------
    def barra_progreso (self):
        progreso = 0
        self.barra_prog = Toplevel()
        self.barra_prog.geometry('300x50+175+200')
        self.barra_prog.transient(self.raiz)
        self.barra_prog.grab_set()
        self.titulo = ttk.Label(self.barra_prog, text ='Procesando...')
        self.titulo.pack(anchor = CENTER)
        self.progress = ttk.Progressbar(self.barra_prog,value = progreso, orient="horizontal",length=200, mode="indeterminate")
        self.progress.pack(anchor = CENTER)
        self.progress.start()
        self.barra_prog.protocol("WM_DELETE_WINDOW", self.on_closing)

#----------------------------------------FUNCIONES BOTON CORRER ----------------------------------------------------------

    def correr(self):

        self.error = False
#       valor_lineas = self.listvar.get()

        if self.listvar.get() == 'Seleccionar línea':
            messagebox.showerror('Error','Por favor seleccione una línea')
        elif (not self.cargar_archivo):
            messagebox.showerror('Error','Por favor cargar archivo .csv')
        elif (self.var.get() == 0):
            messagebox.showerror('Error','Por favor seleccione tipo de algoritmo')
        elif (self.var_2.get() == 0) and (self.var_3.get() == 0):
            messagebox.showerror('Error','Por favor seleccione desde cual extremo o si quiere calcular la impedancia ')
        else:
        #    print('ESTA CORRIENDO...')
            self.barra_progreso()

            self.progress.start()


            def hilo(var,var2,var3):
                if (var == 1):
                    try:
                        if (var2 == 1):
                            [self.lista_faltas_un_extremo,self.graficar_1] =
un_extremo(self.dir_archivo,self.lista[self.listatex.current()].extremo_local,int(self.lista[self.listatex.current()].U_nom),int(self.lista[self.listatex.c
urrent()].I_nom_local),complex(self.lista[self.listatex.current()].impedancia_linea),complex(self.lista[self.listatex.current()].impedancia_homo))
                        elif (var2==2):
                            [self.lista_faltas_un_extremo,self.graficar_1] =
un_extremo(self.dir_archivo,self.lista[self.listatex.current()].extremo_remoto,int(self.lista[self.listatex.current()].U_nom),int(self.lista[self.listat
ex.current()].I_nom_local),complex(self.lista[self.listatex.current()].impedancia_linea),complex(self.lista[self.listatex.current()].impedancia_ho
mo))
                    except :
                        self.error = True
                        messagebox.showerror('Error',"Error al seleccionar línea o archivo csv")
                        self.barra_prog.destroy()

                elif (var==2):
                    try:
                        if (var3==1):
                        #calcular z
                            [self.lista_faltas_un_extremo,self.graficar_1,self.graficar_2,self.impedancia_calc] =
dos_extremos(self.dir_archivo,self.lista[self.listatex.current()].extremo_local,self.lista[self.listatex.current()].extremo_remoto,int(self.lista[self.li
statex.current()].U_nom),int(self.lista[self.listatex.current()].I_nom_local), int(self.lista[self.listatex.current()].I_nom_remota), 0 ,
int(self.lista[self.listatex.current()].pot_reactor_local), int(self.lista[self.listatex.current()].pot_reactor_remoto) )
                        elif (var3==2):
                        #con z de UTE
```

```
                    [self.lista_faltas_un_extremo,self.graficar_1,self.graficar_2,self.impedancia_calc] =
dos_extremos(self.dir_archivo,self.lista[self.listatex.current()].extremo_local,self.lista[self.listatex.current()].extremo_remoto,int(self.lista[self.li
statex.current()].U_nom),int(self.lista[self.listatex.current()].I_nom_local), int(self.lista[self.listatex.current()].I_nom_remota),
complex(self.lista[self.listatex.current()].impedancia_linea) , int(self.lista[self.listatex.current()].pot_reactor_local),
int(self.lista[self.listatex.current()].pot_reactor_remoto) )
                except :
                    self.error = True
                    messagebox.showerror('Error',"Error al seleccionar línea o archivo csv")
                    self.barra_prog.destroy()

            self.barra_prog.destroy()

        self.hilo = threading.Thread(target = hilo,args=(self.var.get(),self.var_2.get(),self.var_3.get()),)
        self.hilo.start()
        self.raiz.wait_window(self.barra_prog)
        if self.error == False:
            self.ventana_resultados()

    self.var.set(0)
    self.var_2.set(0)
    self.var_3.set(0)

#----------------------------------------VENTANA PARA LOS RESULTADOS ----------------------------------------------------------

  def ventana_resultados(self):
    self.ven_resultados = Toplevel()
    self.ven_resultados.grab_set()
    self.ven_resultados.geometry('1000x500')
    self.ven_resultados.transient(self.raiz)
    self.ven_resultados.grab_set()
    self.ven_resultados.resizable(width=FALSE, height=FALSE)
    self.ven_resultados.title('ILFUS')

    ''' Titulo'''
    self.titulo = ttk.Label(self.ven_resultados, text="Resultados",font =( "Helvetica", "12" ,"bold"))
    self.titulo.pack()

    ''' Boton  '''
    ttk.Button(self.ven_resultados, text='Volver',command = self.ven_resultados.destroy,).place(x=650,y=450)

    ''' VENTANA DE TEXTO PARA RESULTADOS '''
    self.txt = tks.ScrolledText(self.ven_resultados, width=55, height=30,font='helvetica 8 ', relief=SUNKEN)
    self.txt.place(x=20,y=40)
    self.txt.tag_configure('resaltado',font= 'helvetica 9 bold italic') # etiqueta para resaltado
    self.txt.config(state = DISABLED)

    ''' CUADRO DE TEXTO'''
    self.txt.config(state = NORMAL)

#-----------------------------------IMPRIME EL TEXTO DE LOS RESULTADOS-----------------------------------------------------------------------------------

    if (len(self.lista_faltas_un_extremo)==0):
        self.txt.insert(INSERT, 'No se encontro Falla en la línea.\n\n','resaltado')
    elif (self.var_2.get()==1) or (self.var_2.get()==0) :# extremo local CONDICION =0 HACE Q ENTRE PARA DOS EXTREMOS
        for i in range(1,len(self.lista_faltas_un_extremo)-1):
            if (i_no_cae(self.lista_faltas_un_extremo[i], self.lista_faltas_un_extremo[i-1])):

                if (self.lista_faltas_un_extremo[i].m >= -.1) and (self.lista_faltas_un_extremo[i].m <= 1.1):

                    if (not i_no_cae(self.lista_faltas_un_extremo[i+1],self.lista_faltas_un_extremo[i]) and (self.lista_faltas_un_extremo[i].m >= -.1)
and (self.lista_faltas_un_extremo[i].m <= 1.1)):
                        self.txt.insert(INSERT, self.lista_faltas_un_extremo[i].falta+'\n'+ 'M = '+ str(format(self.lista_faltas_un_extremo[i].m, '.3f')) +'\n'+
'Distancia (km): '+str(format(self.lista_faltas_un_extremo[i].m * float(self.lista[self.listatex.current()].largo), '.3f') )+' desde
'+self.lista[self.listatex.current()].extremo_local +'\n'+'Muestra numero: '+str(i)+'\nHora:
'+self.lista_faltas_un_extremo[i].muestras_1[2].tiempo+'\n\n','resaltado')
```

```
                elif ( i_no_cae(self.lista_faltas_un_extremo[i+1],self.lista_faltas_un_extremo[i]) and ( not
i_no_cae(self.lista_faltas_un_extremo[i+2],self.lista_faltas_un_extremo[i+1])) and (not (self.lista_faltas_un_extremo[i+1].m >= -.1) or not
(self.lista_faltas_un_extremo[i+1].m <= 1.1)) ):
                    self.txt.insert(INSERT, self.lista_faltas_un_extremo[i].falta+'\n'+ 'M = '+ str(format(self.lista_faltas_un_extremo[i].m, '.3f')) +'\n'+
'Distancia (km): '+str(format(self.lista_faltas_un_extremo[i].m * float(self.lista[self.listatex.current()].largo), '.3f') )+' desde
'+self.lista[self.listatex.current()].extremo_local +'\n'+'Muestra numero: '+str(i)+'\nHora:
'+self.lista_faltas_un_extremo[i].muestras_1[2].tiempo+'\n\n','resaltado')

                else:
                    self.txt.insert(INSERT, self.lista_faltas_un_extremo[i].falta+'\n'+ 'M = '+ str(format(self.lista_faltas_un_extremo[i].m, '.3f')) +'\n'+
'Distancia (km): '+str(format(self.lista_faltas_un_extremo[i].m * float(self.lista[self.listatex.current()].largo), '.3f') )+' desde
'+self.lista[self.listatex.current()].extremo_local +'\n'+'Muestra numero: '+str(i)+'\nHora:
'+self.lista_faltas_un_extremo[i].muestras_1[2].tiempo+'\n\n')
            else:
                self.txt.insert(INSERT,self.lista_faltas_un_extremo[i].falta+'\n'+ 'Error en la localizacion\n'+'Muestra numero: '+str(i)+'\nHora:
'+self.lista_faltas_un_extremo[i].muestras_1[2].tiempo+'\n\n')
        elif self.var_2.get()==2:# extremo remoto
            for i in range(1,len(self.lista_faltas_un_extremo)-1):
                if (i_no_cae(self.lista_faltas_un_extremo[i], self.lista_faltas_un_extremo[i-1])):

                    if (self.lista_faltas_un_extremo[i].m >= -.1) and (self.lista_faltas_un_extremo[i].m <= 1.1):

                        if (not i_no_cae(self.lista_faltas_un_extremo[i+1],self.lista_faltas_un_extremo[i]) and (self.lista_faltas_un_extremo[i].m >= -.1)
and (self.lista_faltas_un_extremo[i].m <= 1.1)):
                            self.txt.insert(INSERT, self.lista_faltas_un_extremo[i].falta+'\n'+ 'M = '+ str(format(self.lista_faltas_un_extremo[i].m, '.3f')) +'\n'+
'Distancia (km): '+str(format(self.lista_faltas_un_extremo[i].m * float(self.lista[self.listatex.current()].largo), '.3f') )+' desde
'+self.lista[self.listatex.current()].extremo_remoto +'\n'+'Muestra numero: '+str(i)+'\nHora:
'+self.lista_faltas_un_extremo[i].muestras_1[2].tiempo+'\n\n','resaltado')

                        elif ( i_no_cae(self.lista_faltas_un_extremo[i+1],self.lista_faltas_un_extremo[i]) and ( not
i_no_cae(self.lista_faltas_un_extremo[i+2],self.lista_faltas_un_extremo[i+1])) and (not (self.lista_faltas_un_extremo[i+1].m >= -.1) or not
(self.lista_faltas_un_extremo[i+1].m <= 1.1)) ):
                            self.txt.insert(INSERT, self.lista_faltas_un_extremo[i].falta+'\n'+ 'M = '+ str(format(self.lista_faltas_un_extremo[i].m, '.3f')) +'\n'+
'Distancia (km): '+str(format(self.lista_faltas_un_extremo[i].m * float(self.lista[self.listatex.current()].largo), '.3f') )+' desde
'+self.lista[self.listatex.current()].extremo_remoto +'\n'+'Muestra numero: '+str(i)+'\nHora:
'+self.lista_faltas_un_extremo[i].muestras_1[2].tiempo+'\n\n','resaltado')

                        else:
                            self.txt.insert(INSERT, self.lista_faltas_un_extremo[i].falta+'\n'+ 'M = '+ str(format(self.lista_faltas_un_extremo[i].m, '.3f')) +'\n'+
'Distancia (km): '+str(format(self.lista_faltas_un_extremo[i].m * float(self.lista[self.listatex.current()].largo), '.3f') )+' desde
'+self.lista[self.listatex.current()].extremo_remoto +'\n'+'Muestra numero: '+str(i)+'\nHora:
'+self.lista_faltas_un_extremo[i].muestras_1[2].tiempo+'\n\n')
                    else:
                        self.txt.insert(INSERT,self.lista_faltas_un_extremo[i].falta+'\n'+ 'Error en la localizacion\n'+'Muestra numero: '+str(i)+'\nHora:
'+self.lista_faltas_un_extremo[i].muestras_1[2].tiempo+'\n\n')

        if self.var_3.get()==1:#imprime la impedancia si es q la calcula
            self.txt.insert(INSERT,'Impedancia calculada: '+ str(format(self.impedancia_calc, '.3f') )+' ohms\n','resaltado')
        self.txt.config(state = DISABLED)

#-------------------------------------------------IMPRME LAS GRAFICAS ----------------------------------------------------------------------------

        ''' GRAFICAS '''
        fondo = Frame(self.ven_resultados ,bd=2, relief=SUNKEN)
        fondo.place(x=420, y =40)
        nb = ttk.Notebook(fondo,width = 550, height= 375 )
        nb.place(x=600,y=15)
        frame1 = Frame(nb)
        frame2 = Frame(nb)
        if len(self.lista_faltas_un_extremo)!=0:
            if(self.var.get()==1):
                figuras_local = crea_fig_corrientes(self.graficar_1)
                canvas1 = FigureCanvasTkAgg(figuras_local, frame1)
                canvas1.get_tk_widget().pack(side=TOP, fill=BOTH, expand=True)
                canvas1._tkcanvas.pack(side=TOP, fill=BOTH, expand=True)
                canvas1.show()
```

```
            nb.add(frame1, text="Extremo local")
        elif (self.var.get()==2):
            figuras_local = crea_fig_corrientes(self.graficar_1)
            figuras_remota= crea_fig_corrientes(self.graficar_2)
            canvas1 = FigureCanvasTkAgg(figuras_local, frame1)
            canvas2 = FigureCanvasTkAgg(figuras_remota, frame2)
            canvas1.get_tk_widget().pack(side=TOP, fill=BOTH, expand=True)
            canvas2.get_tk_widget().pack(side=TOP, fill=BOTH, expand=True)
            canvas1._tkcanvas.pack(side=TOP, fill=BOTH, expand=True)
            canvas2._tkcanvas.pack(side=TOP, fill=BOTH, expand=True)
            canvas1.show()
            canvas2.show()
            nb.add(frame1, text="Extremo local")
            nb.add(frame2, text="Extremo remoto")
    nb.pack(side=BOTTOM,expand=YES)

#---------------------------------------VENTANA PARA EDITAR UNA LINEA ----------------------------------------------------

  def edit_linea (self):
    self.ventana_editar_linea = Toplevel()
    self.ventana_editar_linea.grab_set()
    self.ventana_editar_linea.transient(self.raiz)
    self.ventana_editar_linea.geometry('500x500')
    self.ventana_editar_linea.resizable(width=False, height=False)
    self.ventana_editar_linea.title('ILFUS')
    self.ventana_editar_linea.resizable(width=False, height=False)

    '''titulo'''
    self.titulo = ttk.Label(self.ventana_editar_linea, text="Editar línea",font =( "Helvetica", "10" ,"bold"))
    self.titulo.place(x=210, y=20)

    '''botones '''
    ttk.Button(self.ventana_editar_linea, text='Volver',command = self.ventana_editar_linea.destroy,).place(x=50,y=450)
    ttk.Button(self.ventana_editar_linea, text='Guardar',command = self.guardar_en_txt).place(x=350,y=450)

    '''Combobox con las lineas para elegir cual editar con su etiqueta'''
    self.etiq_edi= Label(self.ventana_editar_linea, text='Seleccionar línea:').place(x=75,y=100)
    self.combox_edi = ttk.Combobox(self.ventana_editar_linea,values=self.lista_nomb , textvariable = self.edit_stringvar , state = 'readonly')
    self.combox_edi.place(x=180, y= 100)
    self.combox_edi.set('')
    self.combox_edi.bind('<<ComboboxSelected>>', self.linea_editar)

    '''Etiquetas para cada atributo de la linea'''
    self.etiq_nomb_local = Label(self.ventana_editar_linea, text= 'Nombre extremo local:').place(x=75,y=140)
    self.etiq_nomb_remoto = Label(self.ventana_editar_linea, text= 'Nombre extremo remoto:').place(x=75,y=165)
    self.etiq_largo = Label(self.ventana_editar_linea, text= 'Largo de la línea (km):').place(x=75,y=190)
    self.etiq_Impedancia = Label(self.ventana_editar_linea, text= 'Impedancia de línea (ohm):').place(x=75,y=215)
    self.etiq_Impedancia = Label(self.ventana_editar_linea, text= 'Impedancia homopolar (ohm):').place(x=75,y=240)
    self.etiq_I_nominal_loc = Label(self.ventana_editar_linea, text= 'TI_local I_pn (A) :').place(x=75,y=265)
    self.etiq_I_nominal_rem = Label(self.ventana_editar_linea, text= 'TI_remoto I_pn (A):').place(x=75,y=290)
    self.etiq_U_nominal = Label(self.ventana_editar_linea, text= 'U_nominal (V):').place(x=75,y=315)
    self.etiq_Q_reactor_loc = Label(self.ventana_editar_linea, text= 'Q reactor local (VAr):').place(x=75,y=340)
    self.etiq_Q_reactor_rem = Label(self.ventana_editar_linea, text= 'Q reactor remoto (VAr):').place(x=75,y=365)

    '''Cuadros de texto para editar'''
    self.tex_nomb_local = Text(self.ventana_editar_linea, width=25, height=1,font='helvetica 8 ')
    self.tex_nomb_local.place(x=250,y=140)
    self.tex_nomb_local.bind("<Tab>", self.focus_next_window)
    self.tex_nomb_local.bind('<Return>', self.onEnter)
    self.tex_nomb_remoto = Text(self.ventana_editar_linea, width=25, height=1,font='helvetica 8 ')
    self.tex_nomb_remoto.place(x=250,y=165)
    self.tex_nomb_remoto.bind("<Tab>", self.focus_next_window)
    self.tex_nomb_remoto.bind('<Return>', self.onEnter)
    self.tex_largo = Text(self.ventana_editar_linea, width=25, height=1,font='helvetica 8 ',)
    self.tex_largo.place(x=250,y=190)
    self.tex_largo.bind("<Tab>", self.focus_next_window)
```

```
        self.tex_largo.bind('<Return>', self.onEnter)
        self.tex_Impedancia = Text(self.ventana_editar_linea, width=25, height=1,font='helvetica 8 ')
        self.tex_Impedancia.place(x=250,y=215)
        self.tex_Impedancia.bind("<Tab>", self.focus_next_window)
        self.tex_Impedancia.bind('<Return>', self.onEnter)
        self.tex_Impedancia_homo = Text(self.ventana_editar_linea, width=25, height=1,font='helvetica 8 ')
        self.tex_Impedancia_homo.place(x=250,y=240)
        self.tex_Impedancia_homo.bind("<Tab>", self.focus_next_window)
        self.tex_Impedancia_homo.bind('<Return>', self.onEnter)
        self.tex_I_nominal_loc = Text(self.ventana_editar_linea, width=25, height=1,font='helvetica 8 ')
        self.tex_I_nominal_loc.place(x=250,y=265)
        self.tex_I_nominal_loc.bind("<Tab>", self.focus_next_window)
        self.tex_I_nominal_loc.bind('<Return>', self.onEnter)
        self.tex_I_nominal_rem = Text(self.ventana_editar_linea, width=25, height=1,font='helvetica 8 ')
        self.tex_I_nominal_rem.place(x=250,y=290)
        self.tex_I_nominal_rem.bind("<Tab>", self.focus_next_window)
        self.tex_I_nominal_rem.bind('<Return>', self.onEnter)
        self.tex_U_nominal = Text(self.ventana_editar_linea, width=25, height=1,font='helvetica 8 ')
        self.tex_U_nominal.place(x=250,y=315)
        self.tex_U_nominal.bind("<Tab>", self.focus_next_window)
        self.tex_U_nominal.bind('<Return>', self.onEnter)
        self.tex_Q_reactor_loc = Text(self.ventana_editar_linea, width=25, height=1,font='helvetica 8 ')
        self.tex_Q_reactor_loc.place(x=250,y=340)
        self.tex_Q_reactor_loc.bind("<Tab>", self.focus_next_window)
        self.tex_Q_reactor_loc.bind('<Return>', self.onEnter)
        self.tex_Q_reactor_rem = Text(self.ventana_editar_linea, width=25, height=1,font='helvetica 8 ')
        self.tex_Q_reactor_rem.place(x=250,y=365)
        self.tex_Q_reactor_rem.bind("<Tab>", self.focus_next_window)
        self.tex_Q_reactor_rem.bind('<Return>', self.onEnter)

    def linea_editar(self,event):
        numero = self.combox_edi.current()
        self.tex_nomb_local.delete(0.0,END)
        self.tex_nomb_local.insert(INSERT,str(self.lista[numero].extremo_local))
        self.tex_nomb_remoto.delete(0.0,END)
        self.tex_nomb_remoto.insert(INSERT,str(self.lista[numero].extremo_remoto))
        self.tex_largo.delete(0.0,END)
        self.tex_largo.insert(INSERT,str(self.lista[numero].largo))
        self.tex_Impedancia.delete(0.0,END)
        self.tex_Impedancia.insert(INSERT,str(self.lista[numero].impedancia_linea))
        self.tex_Impedancia_homo.delete(0.0,END)
        self.tex_Impedancia_homo.insert(INSERT,str(self.lista[numero].impedancia_homo))
        self.tex_I_nominal_loc.delete(0.0,END)
        self.tex_I_nominal_loc.insert(INSERT,str(self.lista[numero].I_nom_local))
        self.tex_I_nominal_rem.delete(0.0,END)
        self.tex_I_nominal_rem.insert(INSERT,str(self.lista[numero].I_nom_remota))
        self.tex_U_nominal.delete(0.0,END)
        self.tex_U_nominal.insert(INSERT,str(self.lista[numero].U_nom))
        self.tex_Q_reactor_loc.delete(0.0,END)
        self.tex_Q_reactor_loc.insert(INSERT,str(self.lista[numero].pot_reactor_local))
        self.tex_Q_reactor_rem.delete(0.0,END)
        self.tex_Q_reactor_rem.insert(INSERT,str(self.lista[numero].pot_reactor_remoto))

    def guardar_en_txt(self):
        ''' funcion para guardar los cambios y actualizar variables'''
        aux =  Linea(self.tex_nomb_local.get("end-1c linestart", "end-1c").replace(' ', ''),self.tex_nomb_remoto.get("end-1c linestart", "end-
1c").replace(' ', ''),self.tex_largo.get("end-1c linestart", "end-1c").replace(' ', ''),self.tex_Impedancia.get("end-1c linestart", "end-1c").replace(' ',
''),self.tex_Impedancia_homo.get("end-1c linestart", "end-1c").replace(' ', ''),self.tex_I_nominal_loc.get("end-1c linestart", "end-1c").replace(' ',
''),self.tex_I_nominal_rem.get("end-1c linestart", "end-1c").replace(' ', ''),self.tex_U_nominal.get("end-1c linestart", "end-1c").replace(' ',
''),self.tex_Q_reactor_loc.get("end-1c linestart", "end-1c").replace(' ', ''),self.tex_Q_reactor_rem.get("end-1c linestart", "end-1c").replace(' ', '') )
        print(aux)
        self.lista.pop(self.combox_edi.current())
        self.lista.append(aux)
        guardar(self.lista)
        self.lista = manejo_lista.cargar()
        self.lista_nomb = self.cargar_lista_nomb(self.lista)
```

```
        self.listvar.set('Seleccionar línea')
        self.listatex.config(values =self.lista_nomb )
        self.Texto.config(state = NORMAL)
        self.Texto.delete(0.0,END)
        self.Texto.config(state = DISABLED)
        self.ventana_editar_linea.destroy()

#----------------------------------------VENTANA PARA AGREGAR UNA LINEA ----------------------------------------------------------

    def agreg_linea (self):
        self.ventana_agregar_linea = Toplevel()
        self.ventana_agregar_linea.grab_set()
        self.ventana_agregar_linea.transient(self.raiz)
        self.ventana_agregar_linea.geometry('500x400')
        self.ventana_agregar_linea.resizable(width=FALSE, height=FALSE)
        self.ventana_agregar_linea.title('ILFUS')
        self.ventana_agregar_linea.resizable(width=False, height=False)

        '''titulo'''
        self.titulo = ttk.Label(self.ventana_agregar_linea, text="Agregar línea",font =( "Helvetica", "10" ,"bold"))
        self.titulo.place(x=210, y=20)

        '''botones '''
        ttk.Button(self.ventana_agregar_linea, text='Volver',command = self.ventana_agregar_linea.destroy).place(x=50,y=350)
        ttk.Button(self.ventana_agregar_linea, text='Agregar',command = self.agregar_en_txt).place(x=350,y=350)

        '''Etiquetas para cada atributo de la línea'''
        self.etiq_nomb_local = Label(self.ventana_agregar_linea, text= 'Nombre extremo local:').place(x=75,y=90)
        self.etiq_nomb_remoto = Label(self.ventana_agregar_linea, text= 'Nombre extremo remoto:').place(x=75,y=115)
        self.etiq_largo = Label(self.ventana_agregar_linea, text= 'Largo de la línea (km):').place(x=75,y=140)
        self.etiq_Impedancia = Label(self.ventana_agregar_linea, text= 'Impedancia de línea (ohm):').place(x=75,y=165)
        self.etiq_Impedancia = Label(self.ventana_agregar_linea, text= 'Impedancia homopolar (ohm):').place(x=75,y=190)
        self.etiq_I_nominal_loc = Label(self.ventana_agregar_linea, text= 'TI_local I_pn (A):').place(x=75,y=215)
        self.etiq_I_nominal_rem = Label(self.ventana_agregar_linea, text= 'TI_remoto I_pn (A):').place(x=75,y=240)
        self.etiq_U_nominal = Label(self.ventana_agregar_linea, text= 'U_nominal (V):').place(x=75,y=265)
        self.etiq_Q_reactor_loc = Label(self.ventana_agregar_linea, text= 'Q reactor local (VAr):').place(x=75,y=290)
        self.etiq_Q_reactor_rem = Label(self.ventana_agregar_linea, text= 'Q reactor remoto (VAr):').place(x=75,y=315)

        '''Cuadros de texto para editar'''
        self.tex_nomb_local = Text(self.ventana_agregar_linea, width=25, height=1,font='helvetica 8 ')
        self.tex_nomb_local.place(x=250,y=90)
        self.tex_nomb_local.bind("<Tab>", self.focus_next_window)
        self.tex_nomb_remoto = Text(self.ventana_agregar_linea, width=25, height=1,font='helvetica 8 ')
        self.tex_nomb_remoto.place(x=250,y=115)
        self.tex_nomb_remoto.bind("<Tab>", self.focus_next_window)
        self.tex_largo = Text(self.ventana_agregar_linea, width=25, height=1,font='helvetica 8 ')
        self.tex_largo.place(x=250,y=140)
        self.tex_largo.bind("<Tab>", self.focus_next_window)
        self.tex_Impedancia = Text(self.ventana_agregar_linea, width=25, height=1,font='helvetica 8 ')
        self.tex_Impedancia.place(x=250,y=165)
        self.tex_Impedancia.bind("<Tab>", self.focus_next_window)
        self.tex_Impedancia_homo = Text(self.ventana_agregar_linea, width=25, height=1,font='helvetica 8 ')
        self.tex_Impedancia_homo.place(x=250,y=190)
        self.tex_Impedancia_homo.bind("<Tab>", self.focus_next_window)
        self.tex_I_nominal_loc = Text(self.ventana_agregar_linea, width=25, height=1,font='helvetica 8 ')
        self.tex_I_nominal_loc.place(x=250,y=215)
        self.tex_I_nominal_loc.bind("<Tab>", self.focus_next_window)
        self.tex_I_nominal_rem = Text(self.ventana_agregar_linea, width=25, height=1,font='helvetica 8 ')
        self.tex_I_nominal_rem.place(x=250,y=240)
        self.tex_I_nominal_rem.bind("<Tab>", self.focus_next_window)
        self.tex_U_nominal = Text(self.ventana_agregar_linea, width=25, height=1,font='helvetica 8 ')
        self.tex_U_nominal.place(x=250,y=265)
        self.tex_U_nominal.bind("<Tab>", self.focus_next_window)
        self.tex_Q_reactor_loc = Text(self.ventana_agregar_linea, width=25, height=1,font='helvetica 8 ')
        self.tex_Q_reactor_loc.place(x=250,y=290)
        self.tex_Q_reactor_loc.bind("<Tab>", self.focus_next_window)
```

```
        self.tex_Q_reactor_rem = Text(self.ventana_agregar_linea, width=25, height=1,font='helvetica 8 ')
        self.tex_Q_reactor_rem.place(x=250,y=315)
        self.tex_Q_reactor_rem.bind("<Tab>", self.focus_next_window)

    def agregar_en_txt(self):
        ''' funcion para guardar los cambios y actualizar variables'''
        aux =  Linea(self.tex_nomb_local.get("end-1c linestart", "end-1c").replace(' ', ''),self.tex_nomb_remoto.get("end-1c linestart", "end-
1c").replace(' ', ''),self.tex_largo.get("end-1c linestart", "end-1c").replace(' ', ''),self.tex_Impedancia.get("end-1c linestart", "end-1c").replace(' ',
''),self.tex_Impedancia_homo.get("end-1c linestart", "end-1c").replace(' ', ''),self.tex_I_nominal_loc.get("end-1c linestart", "end-1c").replace(' ',
''),self.tex_I_nominal_rem.get("end-1c linestart", "end-1c").replace(' ', ''),self.tex_U_nominal.get("end-1c linestart", "end-1c").replace(' ',
''),self.tex_Q_reactor_loc.get("end-1c linestart", "end-1c").replace(' ', ''),self.tex_Q_reactor_rem.get("end-1c linestart", "end-1c").replace(' ', '') )
        self.lista.append(aux)
        guardar(self.lista)
        self.lista = manejo_lista.cargar()
        self.lista_nomb = self.cargar_lista_nomb(self.lista)
        self.listvar.set('Seleccionar línea')
        self.listatex.config(values =self.lista_nomb )
        self.Texto.config(state = NORMAL)
        self.Texto.delete(0.0,END)
        self.Texto.config(state = DISABLED)
        self.ventana_agregar_linea.destroy()

#----------------------------------------VENTANA PARA ELIMINAR UNA LINEA ----------------------------------------------------------

    def elim_linea (self):
        self.ventana_elim_linea = Toplevel()
        self.ventana_elim_linea.grab_set()
        self.ventana_elim_linea.transient(self.raiz)
        self.ventana_elim_linea.geometry('400x250')
        self.ventana_elim_linea.resizable(width=FALSE, height=FALSE)
        self.ventana_elim_linea.title('ILFUS')
        self.ventana_elim_linea.resizable(width=False, height=False)

        '''titulo'''
        self.titulo = ttk.Label(self.ventana_elim_linea, text="Eliminar línea",font =( "Helvetica", "10" ,"bold"))
        self.titulo.place(x=140, y=20)

        '''botones '''
        ttk.Button(self.ventana_elim_linea, text='Volver',command = self.ventana_elim_linea.destroy).place(x=50,y=200)
        ttk.Button(self.ventana_elim_linea, text='Eliminar',command = self.eliminar_del_txt).place(x=270,y=200)

        '''Combobox con las lineas para elegir cual editar con su etiqueta'''
        self.etiq_edi= Label(self.ventana_elim_linea, text='Seleccionar línea:').place(x=50,y=100)
        self.elim_stringvar = NULL
        self.combox_edi = ttk.Combobox(self.ventana_elim_linea,values=self.lista_nomb , textvariable = self.elim_stringvar , state = 'readonly')
        self.combox_edi.place(x=150, y= 100)
        self.combox_edi.set('')

    def eliminar_del_txt(self):
        ''' funcion para guardar los cambios y actualizar variables'''
        if (self.elim_stringvar != NULL) :
            self.lista.pop(self.combox_edi.current())
            guardar(self.lista)
            self.lista = manejo_lista.cargar()
            self.lista_nomb = self.cargar_lista_nomb(self.lista)
            self.listvar.set('Seleccionar línea')
            self.listatex.config(values =self.lista_nomb )
            self.Texto.config(state = NORMAL)
            self.Texto.delete(0.0,END)
            self.Texto.config(state = DISABLED)
        self.ventana_elim_linea.destroy()

#----------------------------------------MEJORA EVENTOS CON ENTER, TAB EN CUADROS TE TEXTO Y SALIDA DE LA APP-----------------------------------------
---------------
    def close_window(self):
        self.raiz.quit()
```

```
        self.raiz.destroy()
        sys.exit()

    def onEnter(self,event):
        self.guardar_en_txt()

    def focus_next_window(self,event):
        event.widget.tk_focusNext().focus()
        return("break")

    def on_closing(self):
        messagebox.showerror('Error',"Espere a terminar el proceso")

#-----------------------------------------------------------------------------------------------------------------------------------------

def main():
    Aplicacion()
    return 0

if __name__ == '__main__':
    main()
```

```
from Pmu import *
from Funciones_localizacion import *
from funciones import *
from copy import deepcopy

#-----------------------------------TAKAGI-----------------------------------------------------------------------
class Tipo_falta():
   '''Clase que guarda el tipo de falta y sus parametros, se usa la que tiene el algoritmo de takagi,
   pero quedan comentadas las otras dos que usan el algoritmo de la reactancia y el de novosel'''

   def __init__(self,Fasores,delta_i,delta_v,delta_homopolar,I_nom,Z_linea,Z_homopolar):

      if ((delta_i==1 and delta_v in [1,3,5,7]) or (delta_i==3 and delta_v in [1,5]) or (delta_i==5 and delta_v in [1,3]) or(delta_i==7 and
delta_v in [1]) ):
         self.falta = 'FALTA MONOFASICA EN C'
         self.tipo = '001'
         self.m = localiz_takagi (Fasores,Z_linea,0,0,1)#localiz_monofasica_3 (Fasores,3, Z_linea)
      elif ((delta_i==2 and delta_v in [2,3,6,7]) or (delta_i==3 and delta_v in [2,6]) or (delta_i==6 and delta_v in [2,3]) or (delta_i==7 and
delta_v in [2]) ):
         self.falta = 'FALTA MONOFASICA EN B'
         self.tipo = '010'
         self.m = localiz_takagi (Fasores,Z_linea,0,1,0)#localiz_monofasica_3 (Fasores,2, Z_linea)
      elif ((delta_i==4 and delta_v in [4,5,6,7]) or (delta_i==5 and delta_v in [4,6]) or (delta_i==6 and delta_v in [4,5]) or(delta_i==7 and
delta_v in [4]) ):
         self.falta = 'FALTA MONOFASICA EN A'
         self.tipo = '100'
         self.m = localiz_takagi (Fasores,Z_linea,1,0,0)#localiz_monofasica_3 (Fasores,1, Z_linea)

      elif ((delta_i==3 and delta_v in [3,7]) or (delta_i==7 and delta_v in [3]) ):
         if (delta_homopolar):
            self.falta ='FALTA BIFASICA EN B-C A TIERRA'
            self.m = localiz_takagi (Fasores,Z_linea,0,1,1)#(localiz_bifasica_tierra_3 (Fasores,2,3, Z_linea))
            self.tipo = '011'
         else:
            self.falta ='FALTA BIFASICA EN B-C'
            self.m =localiz_takagi (Fasores,Z_linea,0,1,1)#(localiz_bifasica_3 (Fasores,2,3, Z_linea))
            self.tipo = '011'
      elif ((delta_i==5 and delta_v in [5,7]) or (delta_i==7 and delta_v in [5]) ):
         if ( delta_homopolar):
            self.falta ='FALTA BIFASICA EN A-C A TIERRA'
            self.m = localiz_takagi (Fasores,Z_linea,1,0,1)#(localiz_bifasica_tierra_3 (Fasores,1,3, Z_linea))
            self.tipo = '101'
         else:
            self.falta ='FALTA BIFASICA EN A-C'
            self.m = localiz_takagi (Fasores,Z_linea,1,0,1)#(localiz_bifasica_3 (Fasores,1,3, Z_linea))
            self.tipo = '101'
      elif ((delta_i==6 and delta_v in [6,7]) or (delta_i==7 and delta_v in [6]) ):
         if ( delta_homopolar):
            self.falta ='FALTA BIFASICA EN A-B A TIERRA'
            self.m = localiz_takagi (Fasores,Z_linea,1,1,0)#(localiz_bifasica_tierra_3 (Fasores,1,2, Z_linea))
            self.tipo = '110'
         else:
            self.falta ='FALTA BIFASICA EN A-B'
            self.m = localiz_takagi (Fasores,Z_linea,1,1,0)#(localiz_bifasica_3 (Fasores,1,2, Z_linea))
            self.tipo = '110'
      elif ((delta_i==7 and delta_v in [7]) ):
         self.falta = 'FALTA TRIFASICA  A-B-C'
         self.m = localiz_takagi (Fasores,Z_linea,1,1,1)#(localiz_trifasica_3(Fasores,Z_linea))
         self.tipo = '111'
      self.muestras_1 = deepcopy(Fasores)



#-----------------------------------NOVOSEL-----------------------------------------------------------------------
'''class Tipo_falta():
```

```
#    Clase que guarda el tipo de falta y sus parametros

   def __init__(self,Fasores,delta_i,delta_v,delta_homopolar,I_nom,Z_linea,Z_homopolar):

       if ((delta_i==1 and delta_v in [1,3,5,7]) or (delta_i==3 and delta_v in [1,5]) or (delta_i==5 and delta_v in [1,3]) or(delta_i==7 and
delta_v in [1]) ):
          self.falta = 'FALTA MONOFASICA EN C'
          self.tipo = '001'
          self.m = localiz_monofasica_novosel (Fasores,3, Z_linea,Z_homopolar)
       elif ((delta_i==2 and delta_v in [2,3,6,7]) or (delta_i==3 and delta_v in [2,6]) or (delta_i==6 and delta_v in [2,3]) or (delta_i==7 and
delta_v in [2]) ):
          self.falta = 'FALTA MONOFASICA EN B'
          self.tipo = '010'
          self.m = localiz_monofasica_novosel (Fasores,2, Z_linea,Z_homopolar)
       elif ((delta_i==4 and delta_v in [4,5,6,7]) or (delta_i==5 and delta_v in [4,6]) or (delta_i==6 and delta_v in [4,5]) or(delta_i==7 and
delta_v in [4]) ):
          self.falta = 'FALTA MONOFASICA EN A'
          self.tipo = '100'
          self.m = localiz_monofasica_novosel (Fasores,1, Z_linea,Z_homopolar)

       elif ((delta_i==3 and delta_v in [3,7]) or (delta_i==7 and delta_v in [3]) ):
          if (delta_homopolar):
             self.falta ='FALTA BIFASICA EN B-C A TIERRA'
             self.m = localiz_bifasica_tierra_novosel (Fasores,2,3, Z_linea)
             self.tipo = '011'
          else:
             self.falta ='FALTA BIFASICA EN B-C'
             self.m =localiz_bifasica_novosel (Fasores,2,3, Z_linea)
             self.tipo = '011'
       elif ((delta_i==5 and delta_v in [5,7]) or (delta_i==7 and delta_v in [5]) ):
          if ( delta_homopolar):
             self.falta ='FALTA BIFASICA EN A-C A TIERRA'
             self.m = localiz_bifasica_tierra_novosel (Fasores,1,3, Z_linea)
             self.tipo = '101'
          else:
             self.falta ='FALTA BIFASICA EN A-C'
             self.m =localiz_bifasica_novosel (Fasores,1,3, Z_linea)
             self.tipo = '101'
       elif ((delta_i==6 and delta_v in [6,7]) or (delta_i==7 and delta_v in [6]) ):
          if ( delta_homopolar):
             self.falta ='FALTA BIFASICA EN A-B A TIERRA'
             self.m = localiz_bifasica_tierra_novosel (Fasores,2,1, Z_linea)
             self.tipo = '110'
          else:
             self.falta ='FALTA BIFASICA EN A-B'
             self.m =localiz_bifasica_novosel (Fasores,2,1, Z_linea)
             self.tipo = '110'
       elif ((delta_i==7 and delta_v in [7]) ):
          self.falta = 'FALTA TRIFASICA  A-B-C'
          self.m = localiz_trifasica_novosel (Fasores,Z_linea)
          self.tipo = '111'
       self.muestras_1 = deepcopy(Fasores)
   '''
#-------------------------------------REACTANCIA-----------------------------------------------------------------------------
'''class Tipo_falta():
#    Clase que guarda el tipo de falta y sus parametros

   def __init__(self,Fasores,delta_i,delta_v,delta_homopolar,I_nom,Z_linea,Z_homopolar):

       if ((delta_i==1 and delta_v in [1,3,5,7]) or (delta_i==3 and delta_v in [1,5]) or (delta_i==5 and delta_v in [1,3]) or(delta_i==7 and
delta_v in [1]) ):
          self.falta = 'FALTA MONOFASICA EN C'
          self.tipo = '001'
          self.m = localiz_un_extremo_a_tierra_reactancia (Fasores[2],Z_linea,Z_homopolar,0,0,1)#localiz_monofasica_3 (Fasores,3,
Z_linea)
```

```
    elif ((delta_i==2 and delta_v in [2,3,6,7]) or (delta_i==3 and delta_v in [2,6]) or (delta_i==6 and delta_v in [2,3]) or (delta_i==7 and
delta_v in [2]) ):
        self.falta = 'FALTA MONOFASICA EN B'
        self.tipo = '010'
        self.m = localiz_un_extremo_a_tierra_reactancia (Fasores[2],Z_linea,Z_homopolar,0,1,0)#localiz_monofasica_3 (Fasores,2,
Z_linea)
    elif ((delta_i==4 and delta_v in [4,5,6,7]) or (delta_i==5 and delta_v in [4,6]) or (delta_i==6 and delta_v in [4,5]) or(delta_i==7 and
delta_v in [4]) ):
        self.falta = 'FALTA MONOFASICA EN A'
        self.tipo = '100'
        self.m = localiz_un_extremo_a_tierra_reactancia (Fasores[2],Z_linea,Z_homopolar,1,0,0)#localiz_monofasica_3 (Fasores,1,
Z_linea)

    elif ((delta_i==3 and delta_v in [3,7]) or (delta_i==7 and delta_v in [3]) ):
        if (delta_homopolar):
            self.falta ='FALTA BIFASICA EN B-C A TIERRA'
            self.m = localiz_un_extremo_a_tierra_reactancia (Fasores[2],Z_linea,Z_homopolar,0,1,1)#(localiz_bifasica_tierra_3
(Fasores,2,3, Z_linea))
            self.tipo = '011'
        else:
            self.falta ='FALTA BIFASICA EN B-C'
            self.m =localiz_un_extremo_bif_reactancia_reactancia (Fasores[2],Z_linea,Z_homopolar,0,1,1)#(localiz_bifasica_3
(Fasores,2,3, Z_linea))
            self.tipo = '011'
    elif ((delta_i==5 and delta_v in [5,7]) or (delta_i==7 and delta_v in [5]) ):
        if ( delta_homopolar):
            self.falta ='FALTA BIFASICA EN A-C A TIERRA'
            self.m = localiz_un_extremo_a_tierra_reactancia (Fasores[2],Z_linea,Z_homopolar,1,0,1)#(localiz_bifasica_tierra_3
(Fasores,1,3, Z_linea))
            self.tipo = '101'
        else:
            self.falta ='FALTA BIFASICA EN A-C'
            self.m = localiz_un_extremo_bif_reactancia (Fasores[2],Z_linea,Z_homopolar,1,0,1)#(localiz_bifasica_3 (Fasores,1,3, Z_linea))
            self.tipo = '101'
    elif ((delta_i==6 and delta_v in [6,7]) or (delta_i==7 and delta_v in [6]) ):
        if ( delta_homopolar):
            self.falta ='FALTA BIFASICA EN A-B A TIERRA'
            self.m = localiz_un_extremo_a_tierra_reactancia (Fasores[2],Z_linea,Z_homopolar,1,1,0)#(localiz_bifasica_tierra_3
(Fasores,1,2, Z_linea))
            self.tipo = '110'
        else:
            self.falta ='FALTA BIFASICA EN A-B'
            self.m = localiz_un_extremo_bif_reactancia (Fasores[2],Z_linea,Z_homopolar,1,1,0)#(localiz_bifasica_3 (Fasores,1,2, Z_linea))
            self.tipo = '110'
    elif ((delta_i==7 and delta_v in [7]) ):
        self.falta = 'FALTA TRIFASICA  A-B-C'
        self.m = localiz_un_extremo_a_tierra_reactancia (Fasores[2],Z_linea,Z_homopolar,1,1,1)#(localiz_trifasica_3(Fasores,Z_linea))
        self.tipo = '111'
    self.muestras_1 = deepcopy(Fasores)'''
```

```
import pandas as pd
import numpy
from Pmu import Pmu # importa la clase Pmu
from Funciones_localizacion import *
from funciones import *
from Tipo_Falta_dos_extremos import *


def
dos_extremos(archivo,nombre_extremo1,nombre_extremo2,U_nom,I_nom_1,I_nom_2,Z_linea,Q_reactor_extremo_1,Q_reactor_extr
emo_2):
    ''' toma los parametros y itera en el csv buscando fallas, devuelve una lista con los PMU de la falla'''
    V_nom =  U_nom/(math.sqrt(3))  # voltaje nominal de la linea de 500kV fase-tierra
    if U_nom == 500000:
        delta_tension = .2
    else: delta_tension = .1
    delta_corriente = .1

    df = pd.read_csv(archivo)
    numeracion_filas_df_1 = indice_encabezados(df.columns,nombre_extremo1)
    numeracion_filas_df_2 = indice_encabezados(df.columns,nombre_extremo2)

    Fasores_1 = []
    Fasores_2 = []
    datos_grafica_local = []
    datos_grafica_remoto = []
    Falta = []
    hubo_falta = False
    impedancia_calculada = False

    for i ,row in df.iterrows():
        Fasor_1_aux = Pmu(  valores(row,numeracion_filas_df_1)  )
        Fasores_1.append(Fasor_1_aux)
        Fasor_2_aux = Pmu(  valores(row,numeracion_filas_df_2)  )
        Fasores_2.append(Fasor_2_aux)
        if (i > 1) :
            if ( (valores_para_impedacia(Fasores_1,Fasores_2,I_nom_1,I_nom_2,delta_corriente)) and (not impedancia_calculada) and
(Z_linea == 0) ):
                Z_linea = calculo_impedancia(Fasores_1,Fasores_2,Q_reactor_extremo_1,Q_reactor_extremo_2)
                impedancia_calculada = True
            if (hubo_falta == True):
                delta_i_1 = comparar_subida_corr_bin(Fasores_1,I_nom_1,2,0,delta_corriente)
                delta_v_1 = comparar_caidas_tens_bin(Fasores_1,V_nom,2,0,delta_tension)
                delta_homopolar_1 = (abs(Fasores_1[2].i0)-abs(Fasores_1[0].i0))/I_nom_1 > .1
                delta_i_2 = comparar_subida_corr_bin(Fasores_2,I_nom_2,2,0,delta_corriente)
                delta_v_2 = comparar_caidas_tens_bin(Fasores_2,V_nom,2,0,delta_tension)
                delta_homopolar_2 = (abs(Fasores_2[2].i0)-abs(Fasores_2[0].i0))/I_nom_2 > .1
                if len(datos_grafica_local)<8:
                    datos_grafica_local.append(Fasor_1_aux)
                    datos_grafica_remoto.append(Fasor_2_aux)
            else:
                delta_i_1 = comparar_subida_corr_bin(Fasores_1,I_nom_1,2,1,delta_corriente)
                delta_v_1 = comparar_caidas_tens_bin(Fasores_1,V_nom,2,1,delta_tension)
                delta_homopolar_1 = (abs(Fasores_1[2].i0)-abs(Fasores_1[1].i0))/I_nom_1 > .1
                delta_i_2 = comparar_subida_corr_bin(Fasores_2,I_nom_2,2,1,delta_corriente)
                delta_v_2 = comparar_caidas_tens_bin(Fasores_2,V_nom,2,1,delta_tension)
                delta_homopolar_2 = (abs(Fasores_2[2].i0)-abs(Fasores_2[1].i0))/I_nom_2 > .1
                datos_grafica_local.append(Fasor_1_aux)
                datos_grafica_remoto.append(Fasor_2_aux)
            if ((delta_i_1==1 and delta_v_1 in [1,3,5,7]) or (delta_i_1==2 and delta_v_1 in [2,3,6,7]) or (delta_i_1==3 and delta_v_1 in
[1,2,3,5,6,7])
                or (delta_i_1==4 and delta_v_1 in [4,5,6,7]) or (delta_i_1==5 and delta_v_1 in [1,3,4,5,6,7]) or (delta_i_1==6 and delta_v_1 in
[2,3,4,5,6,7])
                or (delta_i_1==7 and delta_v_1 in [1,2,3,4,5,6,7]) or
                (delta_i_2==1 and delta_v_2 in [1,3,5,7]) or (delta_i_2==2 and delta_v_2 in [2,3,6,7]) or (delta_i_2==3 and delta_v_2 in
[1,2,3,5,6,7])
```

```
            or (delta_i_2==4 and delta_v_2 in [4,5,6,7]) or (delta_i_2==5 and delta_v_2 in [1,3,4,5,6,7]) or (delta_i_2==6 and delta_v_2 in
[2,3,4,5,6,7])
            or (delta_i_2==7 and delta_v_2 in [1,2,3,4,5,6,7])):
            hubo_falta = True
            Tipo_falta_aux =
Tipo_falta_2(Fasores_1,Fasores_2,delta_i_1,delta_v_1,delta_homopolar_1,delta_i_2,delta_v_2,delta_homopolar_2,Z_linea)
            Falta.append(Tipo_falta_aux)
        if hubo_falta == True:
            Fasores_1.pop(1)
            Fasores_2.pop(1)
        else:
            Fasores_1.pop(0)
            Fasores_2.pop(0)
            datos_grafica_local.pop(0)
            datos_grafica_remoto.pop(0)
    return (Falta,datos_grafica_local,datos_grafica_remoto,Z_linea)
```

```
import pandas as pd
import numpy
from Pmu import Pmu # importa la clase Pmu
from Funciones_localizacion import *
from funciones import *
from Tipo_Falta_un_extremo import *

def un_extremo(archivo,nombre_extremo,U_nom,I_nom,Z_linea,Z_homopolar):
    ''' toma los parametros y itera en el csv buscando fallas, devuelve una lista con los PMU de la falla'''
    V_nom =  U_nom/(math.sqrt(3))  # voltaje nominal de la linea de 500kV fase-tierra
    df1 = pd.read_csv(archivo)
    nombres_filas = indice_encabezados(df1.columns,nombre_extremo)
    Fasores = []
    Falta = []
    datos_grafica = []
    hubo_falta = False
    if U_nom == 500000:
        delta_tension = .2
    else: delta_tension = .1
    delta_corriente = .1
    for i, row in df1.iterrows():
        Fasor_aux = Pmu(  valores(row,nombres_filas)  )
        Fasores.append(Fasor_aux)
        if (i > 1) :
            if (hubo_falta == True):
                delta_i = comparar_subida_corr_bin(Fasores,I_nom,2,0,delta_corriente)
                delta_v = comparar_caidas_tens_bin(Fasores,V_nom,2,0,delta_tension)
                delta_homopolar = (abs(Fasores[2].i0)-abs(Fasores[0].i0))/I_nom > .1
                if len(datos_grafica)<8:
                    datos_grafica.append(Fasor_aux)
            else:
                delta_i = comparar_subida_corr_bin(Fasores,I_nom,2,1,delta_corriente)
                delta_v = comparar_caidas_tens_bin(Fasores,V_nom,2,1,delta_tension)
                delta_homopolar = (abs(Fasores[2].i0)-abs(Fasores[1].i0))/I_nom > .1
                datos_grafica.append(Fasor_aux)
            if ( (delta_i==1 and delta_v in [1,3,5,7]) or (delta_i==2 and delta_v in [2,3,6,7]) or (delta_i==3 and delta_v in [1,2,3,5,6,7])
                 or (delta_i==4 and delta_v in [4,5,6,7]) or (delta_i==5 and delta_v in [1,3,4,5,6,7]) or (delta_i==6 and delta_v in [2,3,4,5,6,7])
                 or (delta_i==7 and delta_v in [1,2,3,4,5,6,7])  ):
                Tipo_falta_aux = Tipo_falta(Fasores,delta_i,delta_v,delta_homopolar,I_nom,Z_linea,Z_homopolar)
                Falta.append(Tipo_falta_aux)
                hubo_falta = True
            if hubo_falta == True:
                Fasores.pop(1)
            else:
                Fasores.pop(0)
                datos_grafica.pop(0)
    return(Falta,datos_grafica)
```

```
from Pmu import *
from Funciones_localizacion import *
from funciones import *
#-----------------------------------------------------------------------


class Tipo_falta_2():
    '''Clase que guarda el tipo de falta y sus parametros, para dos extremos'''

    def __init__(self,Fasores_1, Fasores_2, delta_i_1, delta_v_1, delta_homopolar_1, delta_i_2,delta_v_2,delta_homopolar_2,Z_linea):

        if ((delta_i_1==1 and delta_v_1 in [1,3,5,7]) or (delta_i_1==3 and delta_v_1 in [1,5]) or (delta_i_1==5 and delta_v_1 in [1,3]) or
(delta_i_1==7 and delta_v_1 in [1])) or ((delta_i_2==1 and delta_v_2 in [1,3,5,7])or (delta_i_2==3 and delta_v_2 in [1,5]) or
(delta_i_2==5 and delta_v_2 in [1,3]) or (delta_i_2==7 and delta_v_2 in [1]) ):
            self.falta = 'FALTA MONOFASICA EN C'
            self.tipo = '001'
            self.m = localiz_dos_extremos (Fasores_1[2],Fasores_2[2],Z_linea,0,0,1)
        elif ((delta_i_1==2 and delta_v_1 in [2,3,6,7]) or (delta_i_1==3 and delta_v_1 in [2,6]) or (delta_i_1==6 and delta_v_1 in [2,3]) or
(delta_i_1==7 and delta_v_1 in [2]) or (delta_i_2==2 and delta_v_2 in [2,3,6,7]) or (delta_i_2==3 and delta_v_2 in [2,6]) or
(delta_i_2==6 and delta_v_2 in [2,3]) or (delta_i_2==7 and delta_v_2 in [2])):
            self.falta = 'FALTA MONOFASICA EN B'
            self.m = localiz_dos_extremos (Fasores_1[2],Fasores_2[2],Z_linea,0,1,0)
            self.tipo = '010'
        elif ((delta_i_1==4 and delta_v_1 in [4,5,6,7]) or (delta_i_1==5 and delta_v_1 in [4,6]) or (delta_i_1==6 and delta_v_1 in [4,5]) or
(delta_i_1==7 and delta_v_1 in [4]) or (delta_i_2==4 and delta_v_2 in [4,5,6,7]) or (delta_i_2==5 and delta_v_2 in [4,6]) or
(delta_i_2==6 and delta_v_2 in [4,5]) or (delta_i_2==7 and delta_v_2 in [4])):
            self.falta = 'FALTA MONOFASICA EN A'
            self.m = localiz_dos_extremos (Fasores_1[2],Fasores_2[2],Z_linea,1,0,0)
            self.tipo = '100'

        elif ( ((delta_i_1==3 and delta_v_1 in [3,7]) or (delta_i_1==7 and delta_v_1 in [3])) or ((delta_i_2==3 and delta_v_2 in [3,7]) or
(delta_i_2==7 and delta_v_2 in [3])) ):
            if (delta_v_1 and delta_homopolar_1):
                self.falta ='FALTA BIFASICA EN B-C A TIERRA'
                self.m = localiz_dos_extremos (Fasores_1[2],Fasores_2[2],Z_linea,0,1,1)
                self.tipo = '011'
            elif (delta_v_2 and delta_homopolar_2):
                self.falta ='FALTA BIFASICA EN B-C A TIERRA'
                self.m = localiz_dos_extremos (Fasores_1[2],Fasores_2[2],Z_linea,0,1,1)
                self.tipo = '011'
            else:
                self.falta ='FALTA BIFASICA EN B-C'
                self.m = localiz_dos_extremos (Fasores_1[2],Fasores_2[2],Z_linea,0,1,1)
                self.tipo = '011'
        elif (((delta_i_1==5 and delta_v_1 in [5,7]) or (delta_i_1==7 and delta_v_1 in [5]) )or((delta_i_2==5 and delta_v_2 in [5,7]) or
(delta_i_2==7 and delta_v_2 in [5]) )):
            if (delta_v_1 and delta_homopolar_1):
                self.falta ='FALTA BIFASICA EN A-C A TIERRA'
                self.m = localiz_dos_extremos (Fasores_1[2],Fasores_2[2],Z_linea,1,0,1)
                self.tipo = '101'
            if (delta_v_2 and delta_homopolar_2):
                self.falta ='FALTA BIFASICA EN A-C A TIERRA'
                self.m = localiz_dos_extremos (Fasores_1[2],Fasores_2[2],Z_linea,1,0,1)
                self.tipo = '101'
            else:
                self.falta ='FALTA BIFASICA EN A-C'
                self.m = localiz_dos_extremos (Fasores_1[2],Fasores_2[2],Z_linea,1,0,1)
                self.tipo = '101'
        elif ( ((delta_i_1==6 and delta_v_1 in [6,7]) or (delta_i_1==7 and delta_v_1 in [6])) or ((delta_i_2==6 and delta_v_2 in [6,7]) or
(delta_i_2==7 and delta_v_2 in [6])) ):
            if (delta_v_1 and delta_homopolar_1):
                self.falta ='FALTA BIFASICA EN A-B A TIERRA'
                self.m = localiz_dos_extremos (Fasores_1[2],Fasores_2[2],Z_linea,1,1,0)
                self.tipo = '110'
            if (delta_v_2 and delta_homopolar_2):
```

```
        self.falta ='FALTA BIFASICA EN A-B A TIERRA'
        self.m = localiz_dos_extremos (Fasores_1[2],Fasores_2[2],Z_linea,1,1,0)
        self.tipo = '110'
    else:
        self.falta ='FALTA BIFASICA EN A-B'
        self.m = localiz_dos_extremos (Fasores_1[2],Fasores_2[2],Z_linea,1,1,0)
        self.tipo = '110'
elif ( ((delta_i_1==7 and delta_v_1 in [7]) ) or ((delta_i_2==7 and delta_v_2 in [7]) ) ):
    self.falta = 'FALTA TRIFASICA  A-B-C'
    self.m = localiz_dos_extremos (Fasores_1[2],Fasores_2[2],Z_linea,1,1,1)
    self.tipo = '111'

self.muestras_1 = deepcopy(Fasores_1)
self.muestras_2 = deepcopy(Fasores_2)
```

```
import cmath as cm
import math
import matplotlib.pyplot as plt
from copy import deepcopy
from Tipo_Falta_un_extremo import *
from matplotlib.ticker import EngFormatter

def valores (datos,indice):
    '''INICIA LOS VALORES DE LOS VECTORES, PARA CREAR LA CLASE PMU, USANDO INDICE_ENCABEZADOS'''
    V1 = cm.rect(datos[indice[6]], math.radians(datos[indice[7]]))
    V2 = cm.rect(datos[indice[8]], math.radians(datos[indice[9]]))
    V3 = cm.rect(datos[indice[10]], math.radians(datos[indice[11]]))
    I1 = cm.rect(datos[indice[0]], math.radians(datos[indice[1]]))
    I2 = cm.rect(datos[indice[2]], math.radians(datos[indice[3]]))
    I3 = cm.rect(datos[indice[4]], math.radians(datos[indice[5]]))
    Tiempo = datos['Timestamp']
    return (V1,V2,V3,I1,I2,I3,Tiempo)

#-------------------------------------------------------------

def comparar_subida_corr_bin(lista_fasores,I_nom,muestra_actual,muestra_anterior,delta_corriente):
    '''COMPARA LA LISTA DE FASOERES Y DEVUELVE UN BINARIO DEL 0 AL 8 CON LAS FASES EN QUE SE ELEVA LA CORRIENTE'''
    A=B=C = 0
    if ( (abs(lista_fasores[muestra_actual].i[0])-abs(lista_fasores[muestra_anterior].i[0]) )/I_nom > delta_corriente):
        A = 1
    if ( (abs(lista_fasores[muestra_actual].i[1])-abs(lista_fasores[muestra_anterior].i[1]) )/I_nom > delta_corriente):
        B = 1
    if ( (abs(lista_fasores[muestra_actual].i[2])-abs(lista_fasores[muestra_anterior].i[2]) )/I_nom > delta_corriente):
        C = 1
    return (int(( str(A)+str(B)+str(C)), 2))


def comparar_caidas_tens_bin(lista_fasores,V_nom,muestra_actual,muestra_anterior,delta_temsion):
    '''COMPARA LA LISTA DE FASOERES Y DEVUELVE UN BINARIO DEL 0 AL 8 CON LAS FASES EN QUE SE BAJA LA TENSION'''
    A=B=C = 0
    if ( (abs(lista_fasores[muestra_actual].v[0])-abs(lista_fasores[muestra_anterior].v[0]) )/V_nom < -delta_temsion):
        A = 1
    if ( (abs(lista_fasores[muestra_actual].v[1])-abs(lista_fasores[muestra_anterior].v[1]) )/V_nom < -delta_temsion):
        B = 1
    if ( (abs(lista_fasores[muestra_actual].v[2])-abs(lista_fasores[muestra_anterior].v[2]) )/V_nom < -delta_temsion):
        C = 1
    return (int(( str(A)+str(B)+str(C)), 2))

def indice_encabezados(columnas,nombre_extremo):
    '''TOMA EL VALOR DE LA COLUMNAS QUE ES LA FILA DE LOS ENCABEZADOS,
    Y DEVUELVE LAS POSICIONES DE LAS COMPONENTES ORDENDADAS I1,I2,I3,V1,V2,V3'''
    for i in range(1, columnas.size):
        if ( ( nombre_extremo.lower() in columnas[i].lower() ) and ( ':current a:magnitude' in columnas[i].lower() ) ):
            indice_iAmod = columnas[i]
        elif ( ( nombre_extremo.lower() in columnas[i].lower() ) and ( ':current a:angle' in columnas[i].lower() ) ):
            indice_iAang = columnas[i]
        elif ( ( nombre_extremo.lower() in columnas[i].lower() ) and ( ':current b:magnitude' in columnas[i].lower() ) ):
            indice_iBmod = columnas[i]
        elif ( ( nombre_extremo.lower() in columnas[i].lower() ) and ( ':current b:angle' in columnas[i].lower() ) ):
            indice_iBang = columnas[i]
        elif ( ( nombre_extremo.lower() in columnas[i].lower() ) and ( ':current c:magnitude' in columnas[i].lower() ) ):
            indice_iCmod = columnas[i]
        elif ( ( nombre_extremo.lower() in columnas[i].lower() ) and (':current c:angle' in columnas[i].lower() ) ):
            indice_iCang = columnas[i]
        elif ( ( nombre_extremo.lower() in columnas[i].lower() ) and ( ':voltage a:magnitude' in columnas[i].lower() ) ):
            indice_vAmod = columnas[i]
        elif ( ( nombre_extremo.lower() in columnas[i].lower() ) and ( ':voltage a:angle' in columnas[i].lower() ) ):
            indice_vAang = columnas[i]
        elif ( ( nombre_extremo.lower() in columnas[i].lower() ) and ( ':voltage b:magnitude' in columnas[i].lower() ) ):
            indice_vBmod = columnas[i]
        elif ( ( nombre_extremo.lower() in columnas[i].lower() ) and ( ':voltage b:angle' in columnas[i].lower() ) ):
```

```
            indice_vBang = columnas[i]
        elif ( ( nombre_extremo.lower() in columnas[i].lower() ) and ( ':voltage c:magnitude' in columnas[i].lower() ) ):
            indice_vCmod = columnas[i]
        elif ( ( nombre_extremo.lower() in columnas[i].lower() ) and ( ':voltage c:angle' in columnas[i].lower() ) ):
            indice_vCang = columnas[i]
    indices =
[indice_iAmod,indice_iAang,indice_iBmod,indice_iBang,indice_iCmod,indice_iCang,indice_vAmod,indice_vAang,indice_vBmod,indice_vBang,in
dice_vCmod,indice_vCang]
    return(indices)

def dibujar_extremos(Fasores1,Fasores2):
    '''dibuja en una figura polar los PMU, utilizado para ver la direccion de los flujos de potencia'''
    colores= ['r','b','y']
    ax = plt.subplot(121, polar=True)
    ax.set_title("Extremo 1", va='bottom')
    for i in range(0,3):
        ax.plot([0,cm.phase(Fasores1.i[i])],[0,1],colores[i])
        nombre_I_fase= 'I'+ str(i+1)
        ax.text(cm.phase(Fasores1.i[i]),1/2.,nombre_I_fase,rotation=0,ha='left',va='baseline')#,color=colores[i])
        ax.plot([0,cm.phase(Fasores1.v[i])],[0,1],colores[i])
        nombre_V_fase= 'V'+ str(i+1)
        ax.text(cm.phase(Fasores1.v[i]),1/2.,nombre_V_fase,rotation=0,ha='left',va='baseline')#,color=colores[i])
    ax = plt.subplot(122, polar=True)
    ax.set_title("Extremo 2", va='bottom')
    for i in range(0,3):
        ax.plot([0,cm.phase(Fasores2.i[i])],[0,1],colores[i])
        nombre_I_fase= 'I'+ str(i+1)
        ax.text(cm.phase(Fasores2.i[i]),1/2.,nombre_I_fase,rotation=0,ha='left',va='baseline')#,color=colores[i])
        ax.plot([0,cm.phase(Fasores2.v[i])],[0,1],colores[i])
        nombre_V_fase= 'V'+ str(i+1)
        ax.text(cm.phase(Fasores2.v[i]),1/2.,nombre_V_fase,rotation=0,ha='left',va='baseline')#,color=colores[i])
    plt.show()

def valores_para_impedacia(Fasores_1,Fasores_2,I_nom_1,I_nom_2,delta_corriente):
    '''DEVUELVE UN BOOLEANO VERDADERO SI LAS MUESTRAS SON APTAS PARA CALCULAR LA IMPEDANCIA, OSEA QUE NO SON MUESTRAS EN
FALTA'''
    if (comparar_subida_corr_bin(Fasores_1,I_nom_1,2,1,delta_corriente)==0) and
(comparar_subida_corr_bin(Fasores_2,I_nom_2,2,1,delta_corriente)==0) :
        if (comparar_subida_corr_bin(Fasores_1,I_nom_1,1,0,delta_corriente)==0) and
(comparar_subida_corr_bin(Fasores_2,I_nom_2,1,0,delta_corriente)==0) :
            return True
    else: return False

def calculo_impedancia(Fasores_1,Fasores_2,Q_reactor_extremo_1,Q_reactor_extremo_2):
    '''CALCULO LA IMPEDANCIA USANDO LAS MUESTRAS DE AMBOS EXTREMOS, POR DOS METODOS'''

#------------------------- INICIALIZO VARIABLES-----------------------------------------------------------
    Id1=[]
    Vd1=[]
    Id2=[]
    Vd2=[]

    impedancia =[0,0,0,0]
    impedanciaS2 = [0 ,0 ,0]
    impedanciaSxunidad2 = [0 ,0 ,0]
    extremo_1 = deepcopy(Fasores_1)
    extremo_2 = deepcopy(Fasores_2)
    potencia_1 = Fasores_1[0].v[0]*Fasores_1[0].i[0].conjugate()

    #---------------------------------- ARREGLO LAS MEDIDAS DE LA LINEA, SENTIDO DE LA CORRIENTE Y LA CORRIJO SI HAY REACTOR----------------------
--------

    if (potencia_1.real<0):
        for i in range (0, len(extremo_1)):    ## INVIERTO EL EXTREMO 1
            for j in range (0,len(extremo_1[i].i)):
                extremo_1[i].i[j] = -extremo_1[i].i[j]
```

```
      if (Q_reactor_extremo_1!=0):
        for i in range (0,len(extremo_1)):    ## REACTOR EXTREMO 1
          for j in range (0,len(extremo_1[i].i)):
            extremo_1[i].i[j] = extremo_1[i].i[j] - Q_reactor_extremo_1/(3*(Fasores_1[i].v[j]))#cm.rect(
Q_reactor_extremo_1/(3*abs(Fasores_1[i].v[j])) ,cm.phase(extremo_1[i].v[j])-cm.pi/2)
#math.radians(math.degrees(cm.phase(extremo_1[i].v[j]))-90) )
      if (Q_reactor_extremo_2!=0):
        for i in range (0,len(extremo_2)):    ## REACTOR EXTREMO 2
          for j in range (0,len(extremo_2[i].i)):
            extremo_2[i].i[j] = extremo_2[i].i[j] +
Q_reactor_extremo_2/(3*(Fasores_2[i].v[j]))#cm.rect(Q_reactor_extremo_2/(3*abs(Fasores_2[i].v[j])) ,cm.phase(extremo_1[i].v[j])-cm.pi/2) #
math.radians(math.degrees(cm.phase(extremo_2[i].v[j]))-90) )
    else:
      for i in range (0, len(extremo_2)):    ## INVIERTO EL EXTREMO 2
        for j in range (0,len(extremo_2[i].i)):
          extremo_2[i].i[j] = -extremo_2[i].i[j]
      if (Q_reactor_extremo_1!=0):
        for i in range (0,len(extremo_1)):    ## REACTOR EXTREMO 1
          for j in range (0,len(extremo_1[i].i)):
            extremo_1[i].i[j] = extremo_1[i].i[j] +
Q_reactor_extremo_1/(3*(Fasores_1[i].v[j]))#cm.rect(Q_reactor_extremo_1/(3*abs(Fasores_1[i].v[j])) ,cm.phase(extremo_1[i].v[j])-cm.pi/2) #
math.radians(math.degrees(cm.phase(extremo_1[i].v[j]))-90) )
      if (Q_reactor_extremo_2!=0):
        for i in range (0,len(extremo_2)):    ## REACTOR EXTREMO 2
          for j in range (0,len(extremo_2[i].i)):
            extremo_2[i].i[j] = extremo_2[i].i[j] -
Q_reactor_extremo_2/(3*(Fasores_2[i].v[j]))#cm.rect(Q_reactor_extremo_2/(3*abs(Fasores_2[i].v[j])) ,cm.phase(extremo_1[i].v[j])-cm.pi/2) #
math.radians(math.degrees(cm.phase(extremo_2[i].v[j]))-90) )

#-----------------------------------------------------------'''METODO DE COMPONENTES DIRECTAS'''-----------------------------------

    for i in range(0,len(extremo_1)):
      '''Calculo Id y Vd'''
      Id1.append( (extremo_1[i].i[0] + (cm.exp(1j*math.pi*2/3))*(extremo_1[i].i[1]) + (cm.exp(1j*math.pi*4/3))*(extremo_1[i].i[2]))/3 )
      Vd1.append( (extremo_1[i].v[0] + (cm.exp(1j*math.pi*2/3))*(extremo_1[i].v[1]) + (cm.exp(1j*math.pi*4/3))*(extremo_1[i].v[2]))/3 )
      Id2.append( (extremo_2[i].i[0] + (cm.exp(1j*math.pi*2/3))*(extremo_2[i].i[1]) + (cm.exp(1j*math.pi*4/3))*(extremo_2[i].i[2]))/3 )
      Vd2.append( (extremo_2[i].v[0] + (cm.exp(1j*math.pi*2/3))*(extremo_2[i].v[1]) + (cm.exp(1j*math.pi*4/3))*(extremo_2[i].v[2]))/3 )
    for i in range(0,len(extremo_1)):
      A = (Id1[i]*(Vd1[i]) + (Id2[i])*Vd2[i] ) / ( (Id1[i] * Vd2[i]) + ( Id2[i] * Vd1[i]) )
      C = ((Id1[i]**2) - (Id2[i]**2)) / (Vd2[i]*Id1[i] + Vd1[i]*Id2[i]  )
      B = (( (A**2) -1) / C)
      if (potencia_1.real<0):
        B = -B
      if (B.real<0):
        B = -B.conjugate()
      impedancia[i]=B
    impedanciaS = (impedancia[0]+impedancia[1]+impedancia[2])/3

#-----------------------------------------------------------''' METODO POR FASE'''-----------------------------------------------

    for i in range(0,len(extremo_1)):
      for j in range(0,3):
        A1 = (extremo_1[i].i[j]*(extremo_1[i].v[j]) + (extremo_2[i].i[j])*extremo_2[i].v[j] )/ ( (extremo_1[i].i[j]) *extremo_2[i].v[j] +
(extremo_2[i].i[j])*extremo_1[i].v[j])
        C1 = (extremo_1[i].i[j]**2 - extremo_2[i].i[j]**2) / (extremo_2[i].v[j]*extremo_1[i].i[j] + extremo_1[i].v[j]*extremo_2[i].i[j]  )
        B1 = (( (A1**2) -1) / C1)
        if(potencia_1.real<0):
          B1 = -B1
        if (B1.real<0):
          B1 = -B1.conjugate()
        impedanciaS2[j]+= B1
    for i in range(0,3):
      impedanciaS2[i] = impedanciaS2[i]/3
      impedanciaSxunidad2[i]= impedanciaS2[i]*100000000/(500000**2)/3
    impedancia_de_la_linea = (impedanciaS2[0]+impedanciaS2[1]+impedanciaS2[2])/3
    return impedancia_de_la_linea
```

```
def i_no_cae(falta_actual,falta_anterior):
    '''Detecta si la corriente para la fase correspondiente sube'''
    if falta_actual.tipo == '100':
        if abs(falta_actual.muestras_1[2].i[0]) >= abs(falta_anterior.muestras_1[2].i[0]):
            return True
    elif falta_actual.tipo == '010':
        if abs(falta_actual.muestras_1[2].i[1]) >= abs(falta_anterior.muestras_1[2].i[1]):
            return True
    elif falta_actual.tipo == '001':
        if abs(falta_actual.muestras_1[2].i[2]) >= abs(falta_anterior.muestras_1[2].i[2]):
            return True
    elif (falta_actual.tipo == '110'):
        if abs(falta_actual.muestras_1[2].i[0]) >= abs(falta_anterior.muestras_1[2].i[0]) and abs(falta_actual.muestras_1[2].i[1]) >=
abs(falta_anterior.muestras_1[2].i[1]):
            return True
    elif (falta_actual.tipo == '101'):
        if abs(falta_actual.muestras_1[2].i[0]) >= abs(falta_anterior.muestras_1[2].i[0]) and abs(falta_actual.muestras_1[2].i[2]) >=
abs(falta_anterior.muestras_1[2].i[2]):
            return True
    elif (falta_actual.tipo == '011'):
        if abs(falta_actual.muestras_1[2].i[1]) >= abs(falta_anterior.muestras_1[2].i[1]) and abs(falta_actual.muestras_1[2].i[2]) >=
abs(falta_anterior.muestras_1[2].i[2]):
            return True
    elif (falta_actual.tipo == '111'):
        if abs(falta_actual.muestras_1[2].i[0]) >= abs(falta_anterior.muestras_1[2].i[0]) and abs(falta_actual.muestras_1[2].i[1]) >=
abs(falta_anterior.muestras_1[2].i[1]) and abs(falta_actual.muestras_1[2].i[2]) >= abs(falta_anterior.muestras_1[2].i[2]):
            return True
    else: return False

def crea_fig_corrientes(muestras):
    ''' CREA LA FIGURA PARA EXPONER LOS RESULTADOS, PASANDOLE EL ARREGLO CONTENIENDO LOS PMU'''
    I_A = []
    I_B = []
    I_C = []
    V_A = []
    V_B = []
    V_C = []

    for j in range(0,len(muestras)):
        I_A.append(abs(muestras[j].i[0]))
        I_B.append(abs(muestras[j].i[1]))
        I_C.append(abs(muestras[j].i[2]))
        V_A.append(abs(muestras[j].v[0]))
        V_B.append(abs(muestras[j].v[1]))
        V_C.append(abs(muestras[j].v[2]))

    f, graf = plt.subplots(2, sharex=True,figsize=(5, 4))
    x =[]
    for i in range(0,len(muestras)):
            x.append(i)

    ''' Figura 1, con las corrientes'''

    graf[0].plot(x, I_A, "r",marker='o', linestyle='--',linewidth = 0.5,label= 'Fase A')
    graf[0].plot(x, I_B, "y",marker='o', linestyle='--',linewidth = 0.5,label= 'Fase B')
    graf[0].plot(x, I_C, "b",marker='o', linestyle='--',linewidth = 0.5,label= 'Fase C')
    formato_I = EngFormatter(unit='A')
    graf[0].yaxis.set_major_formatter(formato_I)
    graf[0].set_title('Corrientes')
    graf[0].grid()
    graf[0].legend()

    ''' Figura 2, con las tensiones'''

    graf[1].plot(x, V_A, "r",marker='o', linestyle='--',linewidth = 0.5)
```

```
graf[1].plot(x, V_B, "y",marker='o', linestyle='--',linewidth = 0.5)
graf[1].plot(x, V_C, "b",marker='o', linestyle='--',linewidth = 0.5)
graf[1].set_xlabel("Muestras", fontsize = 9)
formato_V = EngFormatter(unit='V')
graf[1].yaxis.set_major_formatter(formato_V)
graf[1].set_title('Tensiones')
graf[1].grid()
return f
```

```
import cmath as cm
import math

def localiz_dos_extremos(fasores_1,fasores_2,impedancia,A,B,C):
    ''' Metodo para localizar la falla para 2 extremos'''
    m=0
    alfa_0 = ( cm.phase(fasores_2.v[0])-cm.phase(fasores_1.v[0]) )
    alfa_1 = ( cm.phase(fasores_2.v[1])-cm.phase(fasores_1.v[1]) )
    alfa_2 = ( cm.phase(fasores_2.v[2])-cm.phase(fasores_1.v[2]) )
    if A==1:
        m+= ( fasores_1.v[0].real*math.sin(alfa_0) + fasores_1.v[0].imag*math.cos(alfa_0) - fasores_2.v[0].imag +
(impedancia.real*fasores_2.i[0].imag + impedancia.imag*fasores_2.i[0].real) ) / ((impedancia.real*fasores_1.i[0].real-
impedancia.imag*fasores_1.i[0].imag)*math.sin(alfa_0) + (impedancia.real *fasores_1.i[0].imag +
impedancia.imag*fasores_1.i[0].real)*math.cos(alfa_0) + impedancia.real*fasores_2.i[0].imag + impedancia.imag *fasores_2.i[0].real)
    if B==1:
        m+= ( fasores_1.v[1].real*math.sin(alfa_1) + fasores_1.v[1].imag*math.cos(alfa_1) - fasores_2.v[1].imag +
(impedancia.real*fasores_2.i[1].imag + impedancia.imag*fasores_2.i[1].real) ) / ((impedancia.real*fasores_1.i[1].real-
impedancia.imag*fasores_1.i[1].imag)*math.sin(alfa_1) + (impedancia.real *fasores_1.i[1].imag +
impedancia.imag*fasores_1.i[1].real)*math.cos(alfa_1) + impedancia.real*fasores_2.i[1].imag + impedancia.imag *fasores_2.i[1].real)
    if C==1:
        m+= ( fasores_1.v[2].real*math.sin(alfa_2) + fasores_1.v[2].imag*math.cos(alfa_2) - fasores_2.v[2].imag +
(impedancia.real*fasores_2.i[2].imag + impedancia.imag*fasores_2.i[2].real) ) / ((impedancia.real*fasores_1.i[2].real-
impedancia.imag*fasores_1.i[2].imag)*math.sin(alfa_2) + (impedancia.real *fasores_1.i[2].imag +
impedancia.imag*fasores_1.i[2].real)*math.cos(alfa_2) + impedancia.real*fasores_2.i[2].imag + impedancia.imag *fasores_2.i[2].real)
    divisor = A+B+C
    return (m.real/divisor)

#-----------------------------------------------------------REACTANCIA-----------------------------------------------------------

def localiz_un_extremo_a_tierra_reactancia (fasores,Z_linea,Z_homopolar,A,B,C):
    ''' Metodo para un extremo de la reactancia para el caso monofasico, bifasico a tierra y trifasico'''

    I_0 = fasores.i0
    K = (Z_homopolar - Z_linea)/(Z_linea)
    m = 0
    if A==1:
        IA = fasores.i[0]
        VA = fasores.v[0]
        m += ( VA/(IA+K*I_0) ).imag / (Z_linea.imag)
    if B==1:
        IB = fasores.i[1]
        VB = fasores.v[1]
        m += ( VB/(IB+K*I_0) ).imag / (Z_linea.imag)
    if C==1:
        IC = fasores.i[2]
        VC = fasores.v[2]
        m += ( VC/(IC+K*I_0) ).imag / (Z_linea.imag)
    divisor = A+B+C
    m = m / divisor
    return m

def localiz_un_extremo_bif_reactancia (fasores,Z_linea,Z_homopolar,A,B,C):
    ''' Metodo para un extremo de la reactancia para el bifasico, no a tierra'''
    m = 0
    if A==1:
        IA = fasores.i[0]
        VA = fasores.v[0]
        m += ( VA/(IA) ).imag / (Z_linea.imag)
    if B==1:
        IB = fasores.i[1]
        VB = fasores.v[1]
        m += ( VB/(IB) ).imag / (Z_linea.imag)
    if C==1:
        IC = fasores.i[2]
        VC = fasores.v[2]
        m += ( VC/(IC) ).imag / (Z_linea.imag)
```

```
    m = m / 2
    return m

#-------------------------------------------------------------NOVOSEL-----------------------------------------------------------


def localiz_monofasica_novosel (fasores,fase, Z_linea,Z_homopolar):
    ''' Metodo para un extremo de Novosel para el caso monofasico'''
    IsupA = fasores[2].i[0]-fasores[0].i[0]
    IsupB = fasores[2].i[1]-fasores[0].i[1]
    IsupC = fasores[2].i[2]-fasores[0].i[2]
    Isup = (1/3)*(IsupA + IsupB * cm.exp(1j*math.pi*2/3)+ IsupC * cm.exp(1j*math.pi*4/3) )
    I  = fasores[2].i[fase-1]
    V  = fasores[2].v[fase-1]
    m = ( (V.real/Isup.real)-(V.imag/Isup.imag) )/(Z_linea.real*( (I.real/Isup.real) - (I.imag/Isup.imag) ) - Z_linea.imag*( (I.real/Isup.imag) +
(I.imag/Isup.real) ))
    return m

def localiz_bifasica_tierra_novosel (fasores,fase1,fase2, Z_linea):
    ''' Metodo para un extremo de Novosel para el caso bifasico a tierra'''
    Isup = fasores[2].i[fase1-1]-fasores[0].i[fase1-1]
    I  = fasores[2].i[fase1-1]
    V  = fasores[2].v[fase1-1]
    m1 = ( (V.real/Isup.real)-(V.imag/Isup.imag) )/(Z_linea.real*( (I.real/Isup.real) - (I.imag/Isup.imag) )- Z_linea.imag*( (I.real/Isup.imag) +
(I.imag/Isup.real) ))
    Isup = fasores[2].i[fase2-1]-fasores[0].i[fase2-1]
    I  = fasores[2].i[fase2-1]
    V  = fasores[2].v[fase2-1]
    m2 = ( (V.real/Isup.real)-(V.imag/Isup.imag) )/(Z_linea.real*( (I.real/Isup.real) - (I.imag/Isup.imag) )- Z_linea.imag*( (I.real/Isup.imag) +
(I.imag/Isup.real) ))
    return ((m1+m2)/2)

def localiz_bifasica_novosel (fasores,fase1,fase2, Z_linea):
    ''' Metodo para un extremo de Novosel para el caso bifasico, no a tierra'''
    Is = (1/3) * ((fasores[2].i[0]) + (cm.exp(1j*math.pi*4/3))*(fasores[2].i[1]) + (cm.exp(1j*math.pi*2/3))*(fasores[2].i[2]))
    I  = fasores[2].i[fase1-1]
    V  = fasores[2].v[fase1-1]
    m1 = ( (V.real/-Is.imag)-(V.imag/Is.real) )/(Z_linea.real*( (I.real/-Is.imag) - (I.imag/Is.real) )- Z_linea.imag*( (I.real/Is.real) + (I.imag/-
Is.imag) ))
    I  = fasores[2].i[fase2-1]
    V  = fasores[2].v[fase2-1]
    m2 = ( (V.real/-Is.imag)-(V.imag/Is.real) )/(Z_linea.real*( (I.real/-Is.imag) - (I.imag/Is.real) ) - Z_linea.imag*( (I.real/Is.real) + (I.imag/-
Is.imag) ))
    return ((m1+m2)/2)

def localiz_trifasica_novosel (fasores,Z_linea):
    ''' Metodo para un extremo de Novosel para el caso trifasico'''
    Isup = fasores[2].i[0]-fasores[0].i[0]
    I  = fasores[2].i[0]
    V  = fasores[2].v[0]
    m1 = ( (V.real/Isup.real)-(V.imag/Isup.imag) )/(Z_linea.real*( (I.real/Isup.real) - (I.imag/Isup.imag) )- Z_linea.imag*( (I.real/Isup.imag) +
(I.imag/Isup.real) ))
    Isup = fasores[2].i[1]-fasores[0].i[1]
    I  = fasores[2].i[1]
    V  = fasores[2].v[1]
    m2 = ( (V.real/Isup.real)-(V.imag/Isup.imag) )/(Z_linea.real*( (I.real/Isup.real) - (I.imag/Isup.imag) )- Z_linea.imag*( (I.real/Isup.imag) +
(I.imag/Isup.real) ))
    Isup = fasores[2].i[2]-fasores[0].i[2]
    I  = fasores[2].i[2]
    V  = fasores[2].v[2]
    m3 = ( (V.real/Isup.real)-(V.imag/Isup.imag) )/(Z_linea.real*( (I.real/Isup.real) - (I.imag/Isup.imag) )- Z_linea.imag*( (I.real/Isup.imag) +
(I.imag/Isup.real) ))
    return ((m1+m2+m3)/3)

#-------------------------------------------------------------TAKAGI-----------------------------------------------------------
```

```
def localiz_takagi (Fasores,Z_linea,A,B,C):
    ''' Metodo para un extremo de Takagi'''
    m = 0
    if A==1:
        I = Fasores[2].i[0]
        V = Fasores[2].v[0]
        I2 = I-Fasores[0].i[0]
        m += (V * I2.conjugate()).imag / (Z_linea*I*I2.conjugate()).imag
    if B==1:
        I = Fasores[2].i[1]
        V = Fasores[2].v[1]
        I2 = I-Fasores[0].i[1]
        m += (V* I2.conjugate()).imag / (Z_linea*I*I2.conjugate()).imag
    if C==1:
        I = Fasores[2].i[2]
        V = Fasores[2].v[2]
        I2 = I-Fasores[0].i[2]
        m += (V* I2.conjugate()).imag / (Z_linea*I*I2.conjugate()).imag
    divisor = A+B+C
    m = m / divisor
    return m
```

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

class Linea ():
    '''Clase en la que se guardan las caracteristicas de las lineas dadas por UTE'''
    def __init__(self,extremo_local, extremo_remoto, largo,impedancia_linea,impedancia_homo, I_nom_local,
I_nom_remota,U_nom,pot_reactor_local,pot_reactor_remoto):
        self.extremo_local = extremo_local
        self.extremo_remoto = extremo_remoto
        self.largo = largo
        self.impedancia_linea= impedancia_linea
        self.impedancia_homo= impedancia_homo
        self.I_nom_local = I_nom_local
        self.I_nom_remota = I_nom_remota
        self.U_nom = U_nom
        self.pot_reactor_local = pot_reactor_local
        self.pot_reactor_remoto = pot_reactor_remoto

    def imprimir_linea(self):
        print('Extremo Local:',self.extremo_local)
        print('Extremo remoto:',self.extremo_remoto)
        print('Largo de la linea:',self.largo)
        print('Impedancia de linea:', self.impedancia_linea)
        print('Impedancia homopolar:',self.impedancia_homo)
        print('I nominal local:', self.I_nom_local)
        print('I nominal remota:', self.I_nom_remota)
        print('U nominal:', self.U_nom)
        print('Potencia reactor local:',self.pot_reactor_local)
        print('Potencia ractor remoto:',self.pot_reactor_remoto)
        print()

    def imprimir_en_cuadro_text(self):
        cadena = ('Extremo local:              '+self.extremo_local+'\n'+
                  'Extremo remoto:           '+self.extremo_remoto+'\n'+
                  'Largo de la línea:        '+self.largo +' km\n'+
                  'Impedancia de línea::     '+self.impedancia_linea+' ohm\n'+
                  'Impedancia homopolar:   '+self.impedancia_homo+' ohm\n'+
                  'TI_local I_pn:              '+self.I_nom_local+' A\n'+
                  'TI_remoto I_pn:           '+self.I_nom_remota+' A\n'+
                  'U_nominal:                 '+ self.U_nom+' V\n'+
                  'Q reactor local:          '+self.pot_reactor_local +' VAr \n'+
                  'Q reactor remoto:        '+self.pot_reactor_remoto+' VAr ')
        return cadena

    def string_nombre_linea(self):
        nombre_linea = self.extremo_local +'--'+ self.extremo_remoto
        return nombre_linea
    def string_lista(self):
        string_linea = self.extremo_local +' '+ self.extremo_remoto +' '+ self.largo +' '+ self.impedancia_linea +' '+self.impedancia_homo+'
'+ self.I_nom_local +' '+ self.I_nom_remota+' '+self.U_nom +' '+ self.pot_reactor_local+' '+self.pot_reactor_remoto+' \n'
        return string_linea

    #def buscar_extremo_local(self,extremo):



def iniciar_archivo():
    archivo = open('Listas.txt','a')
    archivo.close

def cargar():
    listas = []
    archivo = open('Listas.txt','r')
    linea = archivo.readline()
    if linea:
        while linea:
```

```
            linea=linea.split(' ')
            listas.append(Linea (linea[0],linea[1],linea[2],linea[3],linea[4],linea[5],linea[6],linea[7],linea[8],linea[9].strip('\n')) )
            linea = archivo.readline()
    archivo.close
    return listas

def guardar(lista):
    lista.sort(key = lambda  Linea : Linea.extremo_local)
    archivo = open('Listas.txt','w')
    for i in range(len(lista)):
        archivo.write(lista[i].string_lista())
    archivo.close
```

```
import cmath as cm
import math
import matplotlib.pyplot as plt

class Pmu():
    ''' CLASE QUE GUARDA LA PMU CON SUS 3 VECTORES DE CORRIENTE, VOLTAJE Y SU FECHA'''
    def __init__(self,parametros):
        self.v = [parametros[0],parametros[1],parametros[2]]
        self.i = [parametros[3],parametros[4],parametros[5]]
        self.tiempo = (parametros[6])
        self.i0 = (self.i[0]+self.i[1]+self.i[2])/3
        self.Id = (self.i[0] + (cm.exp(1j*math.pi*2/3))*(self.i[1]) + (cm.exp(1j*math.pi*4/3))*(self.i[2]))/3
        self.Vd = (self.v[0] + (cm.exp(1j*math.pi*2/3))*(self.v[1]) + (cm.exp(1j*math.pi*4/3))*(self.v[2]))/3

    def imprimir_Pmu(self):
        print('V1:{:14.6f},{:11.6f}   I1:{:14.6f},{:11.6f}'.format(
abs(self.v[0]),math.degrees(cm.phase(self.v[0])),abs(self.i[0]),math.degrees(cm.phase(self.i[0]))))
        print('V2:{:14.6f},{:11.6f}   I2:{:14.6f},{:11.6f}'.format(
abs(self.v[1]),math.degrees(cm.phase(self.v[1])),abs(self.i[1]),math.degrees(cm.phase(self.i[1]))))
        print('V3:{:14.6f},{:11.6f}   I3:{:14.6f},{:11.6f}'.format(
abs(self.v[2]),math.degrees(cm.phase(self.v[2])),abs(self.i[2]),math.degrees(cm.phase(self.i[2]))))
        print('Estampa de tiempo: ',self.tiempo)

    def imprimir_sumas(self):
        suma_corr = self.i[0]+self.i[1]+self.i[2]
        suma_tens = self.v[0]+self.v[1]+self.v[2]
        print('Suma de las tensiones:   ', abs(suma_tens),',',math.degrees(cm.phase(suma_tens) ))
        print('Suma de las corrientes:   ', abs(suma_corr),',',math.degrees(cm.phase(suma_corr) ))

    def imprimir_en_rectangulares(self):
        print('V1: ', self.v[0])
        print('V2: ', self.v[1])
        print('V3: ', self.v[2])
        print('I1: ', self.i[0])
        print('I2: ', self.i[1])
        print('I3: ', self.i[2])

    def dibujar_Pmu(self):
        colores= ['r','b','g']
        ax = plt.subplot(111, polar=True)
        ax.grid
        for i in range(0,3):
            ax.plot([0,cm.phase(self.i[i])],[0,1],colores[i])
            nombre_I_fase= 'I'+ str(i+1)
            ax.text(cm.phase(self.i[i]),1/2.,nombre_I_fase,rotation=0,ha='center',va='center',color=colores[i])
            ax.plot([0,cm.phase(self.v[i])],[0,1],colores[i])
            nombre_V_fase= 'V'+ str(i+1)
            ax.text(cm.phase(self.v[i]),1/2.,nombre_V_fase,rotation=0,ha='center',va='center',color=colores[i])
        plt.show()

    def mostrar_Pmu(self):
        mostrar = ('V1:{:14.6f},{:11.6f}   I1:{:14.6f},{:11.6f}').format(
abs(self.v[0]),math.degrees(cm.phase(self.v[0])),abs(self.i[0]),math.degrees(cm.phase(self.i[0]))) +'\n'+ 'V2:{:14.6f},{:11.6f}
I2:{:14.6f},{:11.6f}'.format( abs(self.v[1]),math.degrees(cm.phase(self.v[1])),abs(self.i[1]),math.degrees(cm.phase(self.i[1]))) +'\n'+
'V3:{:14.6f},{:11.6f}   I3:{:14.6f},{:11.6f}'.format(
abs(self.v[2]),math.degrees(cm.phase(self.v[2])),abs(self.i[2]),math.degrees(cm.phase(self.i[2]))) +'\n'+'Estampa de tiempo: '+
self.tiempo +'\n'
        return mostrar

    def mostrar_tiempo_Pmu(self):
        mostrar_tiempo = ( self.tiempo +'\n')
        return mostrar_tiempo
```

Esta página ha sido intencionalmente dejada en blanco.

Apéndice C

Manual de usuario

Identificación y **L**ocalización de **F**allas **U**sando **S**incrofasores

Manual de Usuario

Índice

1. Aspectos generales

La aplicación ILFUS está diseñada para identificar y localizar fallas en líneas de transmisión de 150kV y 500kV, siendo fácil su ampliación a otros niveles de tensión. El modo de uso de la aplicación es muy intuitivo y de fácil manejo, aunque deben tenerse en cuenta algunas consideraciones:

- ILFUS permite cargar archivos únicamente en formato "*csv*", el cual deberá contener las medidas de sincrofasores durante el tiempo de la falla. Dicho archivo debe por lo menos contener 5 minutos antes de que comience la falla y el tamaño del mismo influirá en el tiempo de procesamiento de ILFUS.
- Cuando se editen o agreguen nuevas líneas, hay que prestar especial atención en las etiquetas con nombres de las mismas. Para que el software pueda procesar los datos del archivo cargado, la etiqueta con el nombre de línea en el archivo debe coincidir con la etiqueta de la línea pre cargada. Además hay que prestar especial atención al ingresar los parámetros de las líneas (Las unidades de cada parámetro están especificadas) ya que una correcta localización depende de ello. Los números decimales deben estar representados con punto en lugar de coma (Ej. 12.34). Al ingresar una impedancia usando números complejos, el formato debe corresponder con a+bj.
- La aplicación cuenta con dos métodos de localización de fallas (1 extremo y dos extremos). Quedará a criterio del usuario elegir cual método es el adecuado en cada caso para obtener un mejor resultado de la estimación del punto de falla.
- Para el caso de líneas que cuenten con medidas en los dos extremos, ILFUS dará la opción de realizar la localización usando los datos de impedancia pre cargados en la lista de líneas de transmisión o calculando la impedancia de línea con las medidas extraídas del archivo *csv*.

1.1. Pantalla principal

La pantalla principal cuenta con el nombre de la aplicación en la esquina superior izquierda, y en la esquina superior derecha se aprecian los botones de minimizar, maximizar y cerrar. También cuenta con opciones de archivo, edición y ayuda. Contiene además cuadros de texto, barras de desplazamiento vertical y botones de selección.

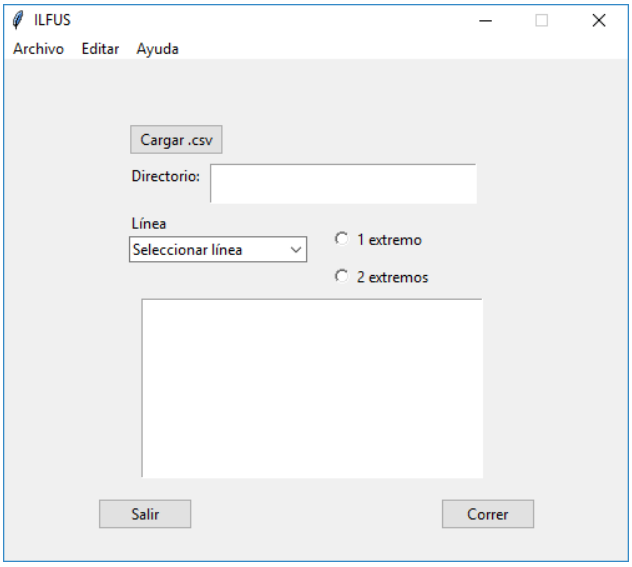


Figura 1.

A continuación se presenta el contenido de la pantalla principal en detalle.

1.1.1. Contenido

Una vez iniciado el software, la imagen que se observa es la siguiente y corresponde a la pantalla principal de la aplicación.

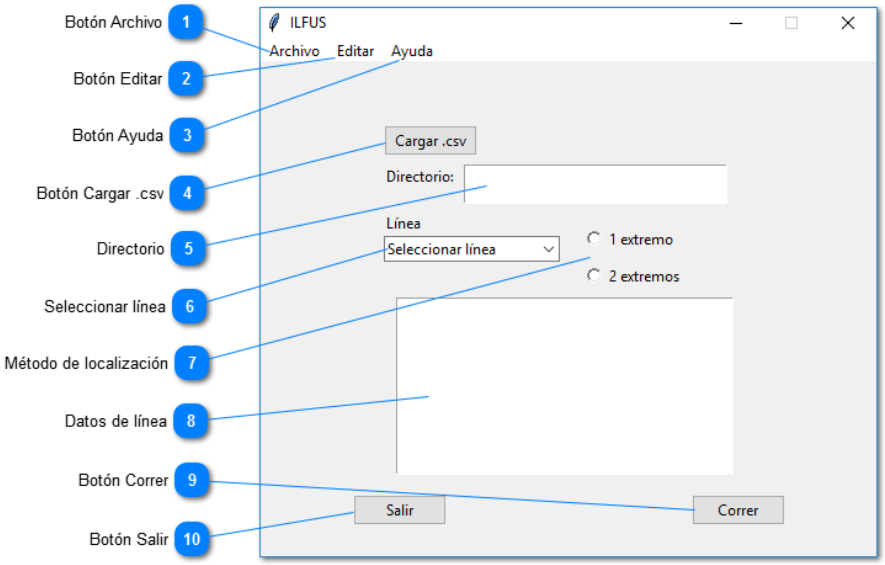


Figura 2.

1. **Botón Archivo:** Permite cargar un archivo en formato *csv* y permite salir de la aplicación.
2. **Botón Editar:** Permite trabajar sobre la información archivada acerca de las líneas.
3. **Botón Ayuda:** Contiene información acerca de la aplicación.
4. **Botón Cargar .csv:** Permite cargar un archivo en formato *csv*.
5. **Directorio:** Muestra el directorio del archivo cargado.
6. **Seleccionar línea:** Permite seleccionar una línea de transmisión de una lista de líneas archivadas.
7. **Método de localización:** Dependiendo de la línea que se seleccione, ésta constará con medidas en uno o dos extremos. Para el caso de líneas con dos extremos se debe seleccionar que método de localización se desea aplicar. Las opciones a seleccionar corresponden a localizar con medidas desde un extremo o desde dos extremos.

8. **Datos de línea:** En este recuadro se muestran los datos de la línea seleccionada, así como también información sobre los reactores si corresponde.
9. **Botón Correr:** Inicia el proceso de detección y localización con los datos seleccionados.
10. **Botón Salir:** Sale de la aplicación.

1.2. Lista de líneas

La aplicación cuenta con una lista de líneas de transmisión de 150kV y 500kV pre cargada. Dicha lista podrá ser modificada por el usuario. El usuario podrá modificar los parámetros de una línea, eliminarla o agregar una nueva línea a la lista.

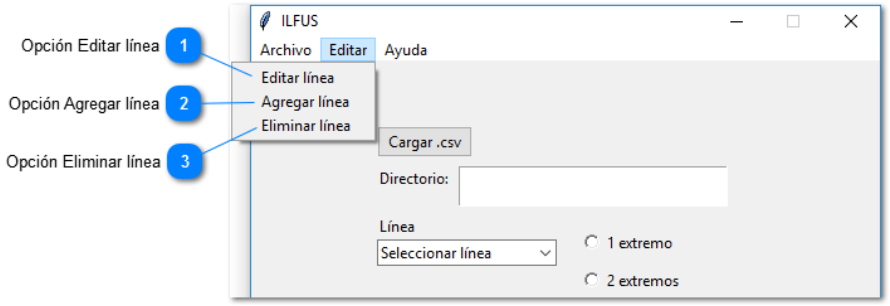


Figura 3.

1. **Opción Editar línea:** Permite editar el contenido de una línea de transmisión perteneciente a la lista de líneas.
2. **Opción Agregar línea:** Permite agregar una nueva línea de transmisión a la lista de líneas.
3. **Opción Eliminar línea:** Permite eliminar una línea de transmisión de la lista de líneas.

1.2.1. Editar línea

Permite modificar el contenido de una línea. Es decir, el usuario podrá modificar los valores de los parámetros de una línea perteneciente a la lista de líneas guardadas. Para acceder a esta opción desde la pantalla principal se debe seguir la siguiente secuencia de pasos: Editar/Editar_línea. Una vez realizada la secuencia de pasos mencionada, se despliega en pantalla la siguiente ventana:

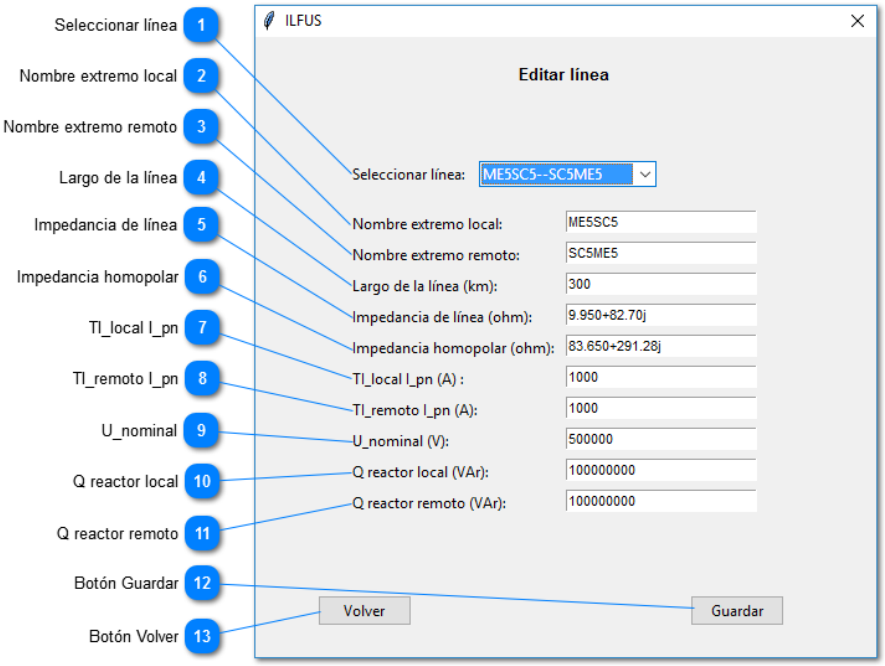


Figura 4.

1. **Seleccionar línea:** Permite seleccionar una línea de la lista de líneas guardada.

2. **Nombre extremo local:** Permite modificar el nombre del extremo local de la línea seleccionada de la lista de líneas. Para que el software pueda procesar los datos del archivo cargado en la pantalla principal (Directorio), el nombre que se ingresa en este campo debe coincidir con la etiqueta del nombre de la línea del archivo *csv* (Ejemplo: MA5PA5; ver sección 1.3).

3. **Nombre extremo remoto:** Permite modificar el nombre del extremo remoto de la línea seleccionada de la lista de líneas. Para que el software pueda procesar los datos del archivo cargado en la pantalla principal (Directorio), el nombre que se ingresa en este campo debe coincidir con la

etiqueta del nombre de la línea del archivo *csv* (Ejemplo: PA5MA5; ver sección 1.3).

4. **Largo de la línea:** Permite modificar la longitud de la línea. El valor a ingresar debe de estar en km y en caso de números con decimales se deben agregar con punto en lugar de coma (Ej: 12.34 km).

5. **Impedancia de línea:** Permite modificar el valor de impedancia de la línea (Impedancia directa o sincrónica). El valor a ingresar debe de estar en ohms y tener el formato $a + bj$.

6. **Impedancia homopolar:** Permite modificar el valor de impedancia homopolar de la línea (Impedancia de secuencia 0). El valor a ingresar debe de estar en ohms y tener el formato $a + bj$.

7. **TI_local I_pn:** Permite modificar el valor de corriente nominal del extremo local. Este valor corresponde a la corriente nominal primaria del transformador de medida ubicado en el extremo local. El valor a ingresar debe de estar en amperes.

8. **TI_remoto I_pn:** Permite modificar el valor de corriente nominal del extremo remoto. Este valor corresponde a la corriente nominal primaria del transformador de medida ubicado en el extremo remoto. El valor a ingresar debe de estar en amperes.

9. **U_nominal:** Permite modificar la tensión nominal compuesta de la línea (Tensión entre fases). El valor debe ser ingresado en voltios.

10. **Q reactor local:** Permite modificar el valor de la potencia del reactor del extremo local. Para los casos en que las líneas no cuentan con reactor local se debe de fijar el valor de la potencia en 0 VAr. El valor de potencia a ingresar debe de estar en VAr.

11. **Q reactor remoto:** Permite modificar el valor de la potencia del reactor del extremo remoto. Para los casos en que las líneas no cuentan con reactor remoto se debe de fijar el valor de la potencia en 0 VAr. El valor de potencia a ingresar debe de estar en VAr.

12. **Botón Guardar:** Permite guardar los cambios realizados a la línea de transmisión.

13. **Botón Volver:** Permite regresar a la pantalla principal, descartando las posibles modificaciones realizadas.

1.2.2. Agregar línea

Permite agregar una nueva línea de transmisión a la lista de líneas.

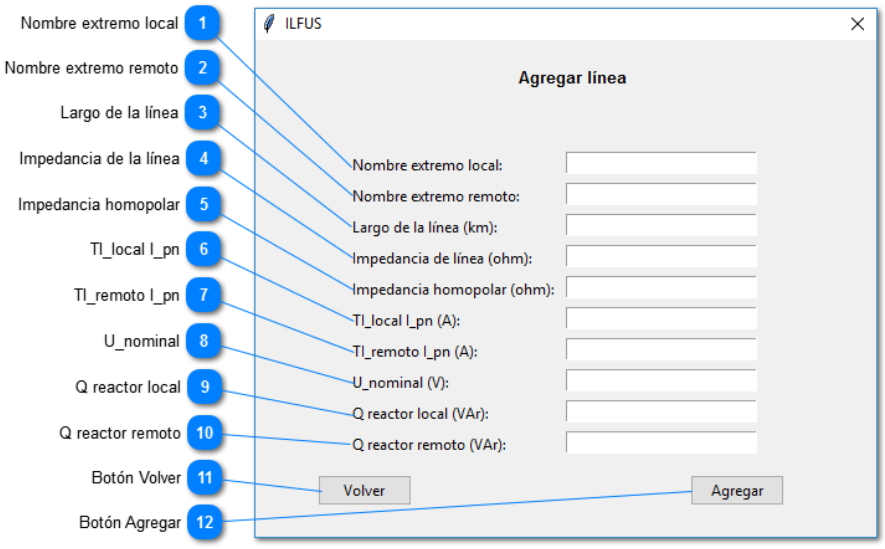


Figura 5.

1. **Nombre extremo local:** Se debe ingresar el nombre del extremo local de la línea a agregar a la lista. Para que el software pueda posteriormente procesar los datos del archivo cargado en la pantalla principal (Directorio), el nombre que se ingresa en este campo debe coincidir con la etiqueta del nombre de la línea del archivo *csv* (Ejemplo: MA5PA5; ver sección 1.3).

2. **Nombre extremo remoto:** Se debe ingresar el nombre del extremo remoto de la línea a agregar a la lista. Para que el software pueda posteriormente procesar los datos del archivo cargado en la pantalla principal(Directorio), el nombre que se ingresa en este campo debe coincidir con la etiqueta del nombre de la línea del archivo *csv*(Ejemplo: PA5MA5; ver sección 1.3).

3. **Largo de la línea:** Se debe ingresar el largo total de la línea en km. El valor a ingresar debe de estar en km y en caso de números con decimales se deben agregar con punto en lugar de coma (Ej: 12.34 km).

4. **Impedancia de línea:** Se debe ingresar la impedancia total en ohms (impedancia de secuencia directa o sincrónica) de la línea a agregar a la lista. El formato de la impedancia debe ser $a + bj$.

5. **Impedancia homopolar:** Se debe ingresar la impedancia homopolar en ohms (impedancia de secuencia 0) de la línea a agregar a la lista. El formato de la impedancia debe ser $a + bj$.

6. **TI_local I_pn:** Se debe ingresar el valor de corriente nominal primaria del transformador de medida del extremo local. El valor a ingresar debe de estar en amperes.

7. **TI_remoto I_pn:** Se debe ingresar el valor de corriente nominal primaria del transformador de medida del extremo remoto. El valor a ingresar debe de estar en amperes.

8. **U_nominal:** Se debe ingresar el valor de tensión compuesta de la línea a ingresar. El valor a ingresar debe de estar en voltios.

9. **Q reactor local:** Se debe ingresar el valor de potencia del reactor local de la línea a ingresar. De no existir reactor en el extremo local de la línea se debe ingresar un 0. El valor de potencia a ingresar debe de estar en VAr.

10. **Q reactor remoto:** Se debe ingresar el valor de potencia del reactor remoto de la línea a ingresar. De no existir reactor en el extremo remoto de la línea se debe ingresar un 0. El valor de potencia a ingresar debe de estar en VAr.

11. **Botón Volver:** Permite regresar a la pantalla principal, descartando los posibles valores ingresados.

12. **Botón Agregar:** Guarda la nueva línea de transmisión en la lista de líneas.

1.2.3. Eliminar línea

Permite eliminar una línea de transmisión de la lista de líneas guardada.

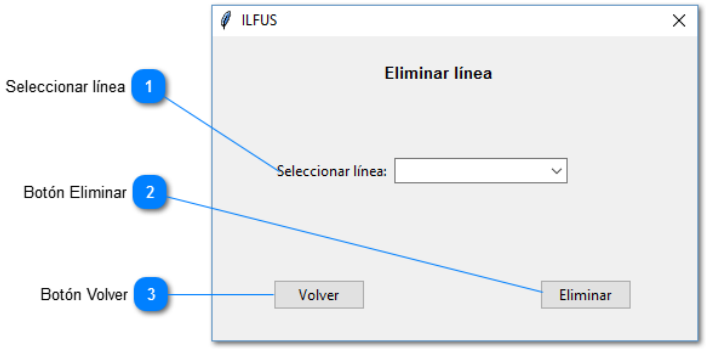
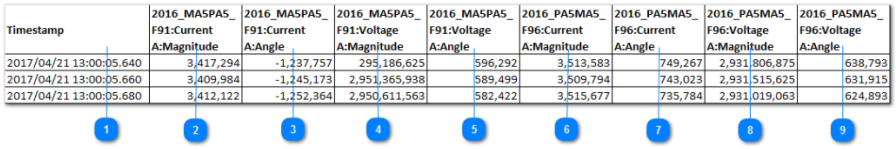


Figura 6.

1. **Seleccionar línea:** Permite seleccionar de la lista de líneas, la línea que se desea eliminar.
2. **Botón Eliminar:** Elimina de la lista de líneas, la línea seleccionada en "Seleccionar línea".
3. **Botón Volver:** Permite regresar a la pantalla principal sin realizar acción alguna.

1.3. Formato de archivo y estructura

El archivo a cargar debe tener el formato "*csv*", y debe contener en su interior información sobre la falla a la cual se le desea aplicar los algoritmos de identificación y localización. La estructura del archivo debe ser tal que como mínimo contenga valores de corrientes y tensiones (módulo y ángulo) de todas las fases desde un extremo de la línea con sus respectivas estampas de tiempo. En la siguiente figura se muestra a modo de ejemplo, parte del contenido de un archivo[1]. La primer fila, contiene en cada celda información sobre el contenido de cada columna y las demás filas contienen los datos relevados en cada instante de tiempo.



Timestamp	2016_MA5PA5_ F91:Current A:Magnitude	2016_MA5PA5_ F91:Current A:Angle	2016_MA5PA5_ F91:Voltage A:Magnitude	2016_MA5PA5_ F91:Voltage A:Angle	2016_PA5MA5_ F96:Current A:Magnitude	2016_PA5MA5_ F96:Current A:Angle	2016_PA5MA5_ F96:Voltage A:Magnitude	2016_PA5MA5_ F96:Voltage A:Angle
2017/04/21 13:00:05.640	3,417,294	-1,237,757	295,186,625	596,292	3,513,583	749,267	2,931,806,875	638,793
2017/04/21 13:00:05.660	3,409,984	-1,245,173	2,951,365,938	589,499	3,509,794	743,023	2,931,515,625	631,915
2017/04/21 13:00:05.680	3,412,122	-1,252,364	2,950,611,563	582,422	3,515,677	735,784	2,931,019,063	624,893

Figura 7.

1. **Estampa de tiempo:** En esta columna se puede ver la fecha y hora en la que fue tomada cada muestra. Esta muestra estará 3 horas adelantada, debido a que está sincronizada con el tiempo UTC/GMT.

2. **2016_MA5PA5_F91: Current A: Magnitude:** En esta columna se pueden ver los módulos de la corriente de la fase A para cada instante de tiempo. "MA5PA5" es la etiqueta del nombre de la línea y corresponde a la línea "Montevideo A - Palmar" de 500kV, donde "Montevideo A" es el extremo en el cual se realiza la medida indicada (extremo local) y "Palmar" el extremo remoto.

3. **2016_MA5PA5_F91: Current A: Angle:** En esta columna se pueden ver los ángulos de la corriente de la fase A para cada instante de tiempo. "MA5PA5" es la etiqueta del nombre de la línea y corresponde a la línea "Montevideo A - Palmar" de 500kV, donde "Montevideo A" es el extremo en el cual se realiza la medida indicada (extremo local) y "Palmar" el extremo remoto.

4. **2016_MA5PA5_F91: Voltage A: Magnitude:** En esta columna se pueden ver los módulos de la tensión (Voltaje fase-tierra) de la fase A para cada instante de tiempo. "MA5PA5" es la etiqueta del nombre de la línea y corresponde a la línea "Montevideo A - Palmar" de 500kV, donde "Montevideo A" es el extremo en el cual se realiza la medida indicada (extremo local) y "Palmar" el extremo remoto.

[1]Debido al tamaño de los archivos se muestra únicamente las corrientes y tensiones de una única fase con medidas desde ambos extremos

5. **2016_MA5PA5_F91: Voltage A: Angle:** En esta columna se pueden ver los ángulos de la tensión (Voltaje fase-tierra) de la fase A para cada instante de tiempo. "MA5PA5" es la etiqueta del nombre de la línea y corresponde a la línea "Montevideo A - Palmar" de 500kV, donde "Montevideo A" es el extremo en el cual se realiza la medida indicada (extremo local) y "Palmar" el extremo remoto.

6. **2016_PA5MA5_F96: Current A: Magnitude:** En esta columna se pueden ver los módulos de la corriente de la fase A para cada instante de tiempo. "PA5MA5" es la etiqueta del nombre de la línea y corresponde a la línea "Montevideo A - Palmar" de 500kV, donde "Palmar" es el extremo en el cual se realiza la medida indicada (extremo local) y "Montevideo A" el extremo remoto.

7. **2016_PA5MA5_F96: Current A: Angle:** En esta columna se pueden ver los ángulos de la corriente de la fase A para cada instante de tiempo. "PA5MA5" es la etiqueta del nombre de la línea y corresponde a la línea "Montevideo A - Palmar" de 500kV, donde "Palmar" es el extremo en el cual se realiza la medida indicada (extremo local) y "Montevideo A" el extremo remoto.

8. **2016_PA5MA5_F96: Voltage A: Magnitude:** En esta columna se pueden ver los módulos de la tensión (Voltaje fase-tierra) de la fase A para cada instante de tiempo. "PA5MA5" es la etiqueta del nombre de la línea y corresponde a la línea "Montevideo A - Palmar" de 500kV, donde "Palmar" es el extremo en el cual se realiza la medida indicada (extremo local) y "Montevideo A" el extremo remoto.

9. **2016_PA5MA5_F96: Voltage A: Angle:** En esta columna se pueden ver los ángulos de la tensión (Voltaje fase-tierra) de la fase A para cada instante de tiempo. "PA5MA5" es la etiqueta del nombre de la línea y corresponde a la línea "Montevideo A - Palmar" de 500kV, donde "Palmar" es el extremo en el cual se realiza la medida indicada (extremo local) y "Montevideo A" el extremo remoto.

2. Identificar y localizar fallas desde un extremo

Para identificar y localizar una falla en una línea de transmisión mediante el procesamiento de los datos obtenidos desde un extremo de la misma, se deberá realizar la siguiente secuencia de pasos:

Paso 1: Abrir la aplicación ILFUS.

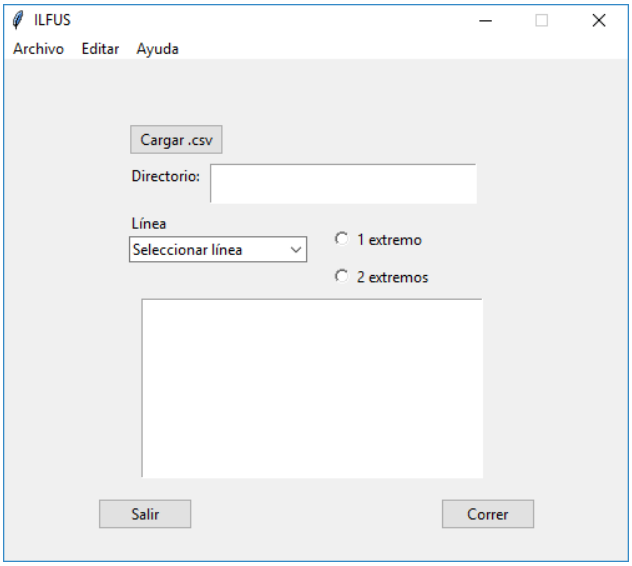


Figura 8.

Paso 2: Cargar el archivo en formato *csv* con las mediciones del momento de la falla a identificar y localizar.

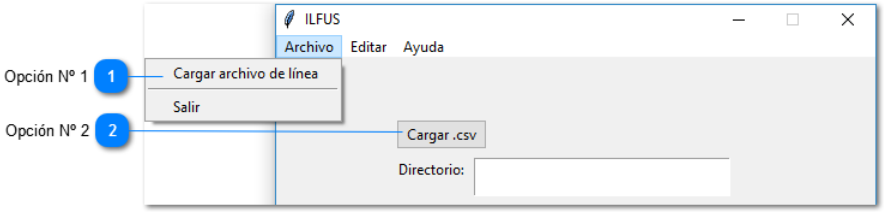


Figura 9.

Opción Nº 1:

1. Seleccionar el menú Archivo.

2. Opción Cargar Archivo de línea.
3. Seleccionar el directorio del archivo deseado en la PC.

Opción Nº 2:

1. Presionar el botón "Cargar .csv"
2. Seleccionar el directorio del archivo deseado en la PC.

Paso 3: Seleccionar la línea de transmisión de la lista de líneas que concuerde con los datos cargados en el archivo .csv.

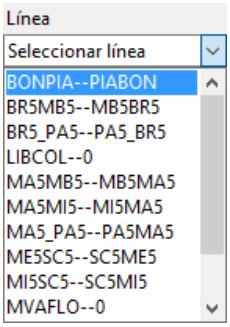


Figura 10.

Si la línea seleccionada no se corresponde con el archivo *csv*, al correr el programa se muestra una ventana emergente comunicando que los datos no se corresponden. En el caso de que la línea a identificar y localizar no se encuentre dentro de la lista de líneas, se deberá primero ingresar la nueva línea a la lista de líneas y luego continuar en el Paso 2.

Paso 4: Seleccionar el método "1 extremo"

Al seleccionar la opción 1 extremo, se despliega a su derecha una segunda columna de opciones.

Paso 5: Seleccionar desde cual de los dos extremos se desea identificar y localizar la falla.

Existen dos posibles casos a tener en cuenta para esta elección:

1. La línea elegida cuenta con medida de un solo extremo: en ese caso se debe elegir el extremo correcto.
2. La línea cuenta con medidas de los dos extremos: en ese caso se elige la opción deseada.

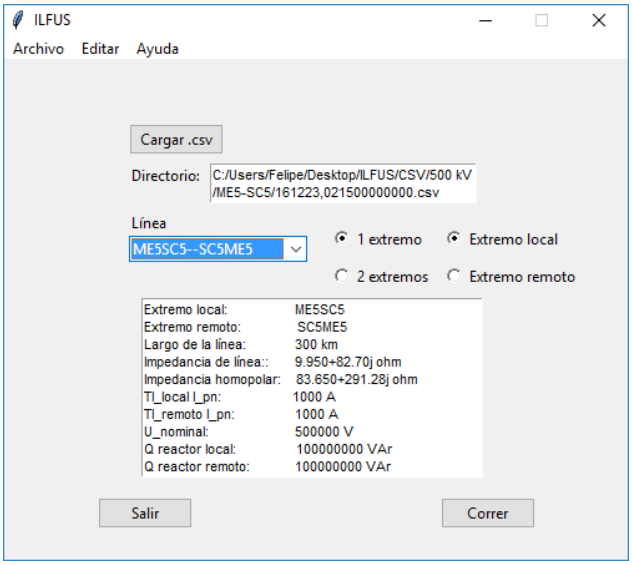


Figura 11.

Paso 6: Presionar el botón Correr

Luego de procesar los datos, mediante una ventana emergente se muestran los resultados obtenidos (Ver sección 4).

3. Identificar y localizar fallas desde dos extremos

Para identificar y localizar una falla en una línea de transmisión mediante el procesamiento de los datos obtenidos desde dos extremo de la misma, se deberá realizar la siguiente secuencia de pasos:

Paso 1: Abrir la aplicación ILFUS.

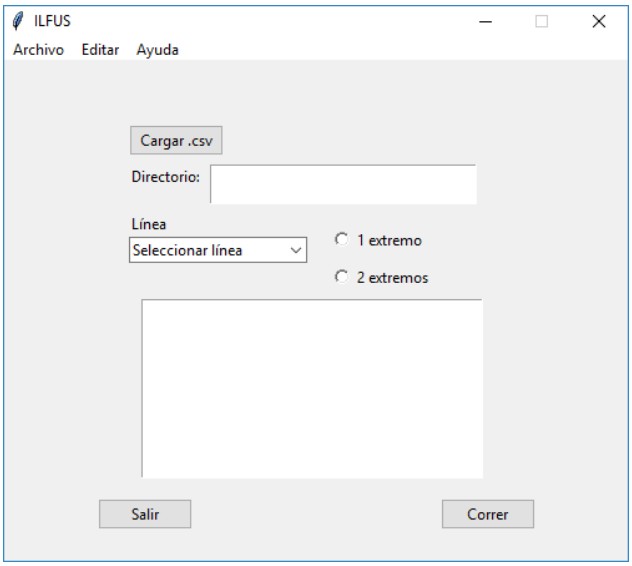


Figura 12.

Paso 2: Cargar el archivo en formato *csv* con las mediciones del momento de la falla a identificar y localizar.

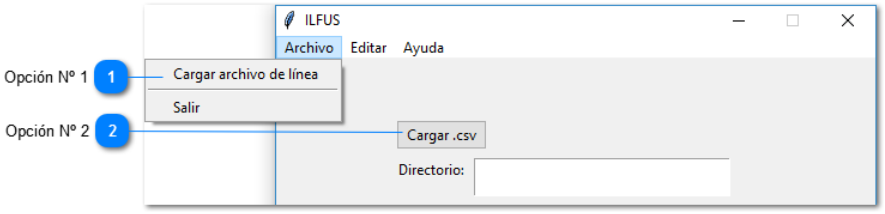


Figura 13.

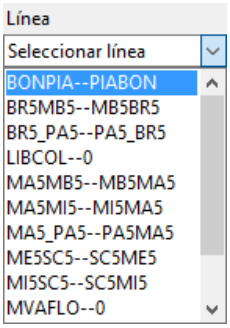
Opción Nº 1:

1. Seleccionar el menú Archivo.
2. Opción Cargar Archivo de línea.
3. Seleccionar el directorio del archivo deseado en la PC.

Opción Nº 2:

1. Presionar el botón "Cargar .csv"
2. Seleccionar el directorio del archivo deseado en la PC.

Paso 3: Seleccionar la línea de transmisión de la lista de líneas que concuerde con los datos cargados en el archivo *csv*.



Si la línea seleccionada no se corresponde con el archivo *csv*, al correr el programa se muestra una ventana emergente comunicando que los datos no se corresponden. En el caso de que la línea a identificar y localizar no se encuentre dentro de la lista de líneas, se deberá primero ingresar la nueva línea a la lista de líneas y luego continuar en el Paso 2.

Paso 4: Seleccionar el método "2 extremo"

Al seleccionar la opción 2 extremo, se despliega a su derecha una segunda columna de opciones.

Paso 5: Seleccionar que impedancia de línea desea utilizar para identificar y localizar la falla.

Existen dos posibles casos a tener en cuenta para esta elección:

1. **Calcular Z:** en este caso, ILFUS implementa un algoritmo para el cálculo de impedancia de línea mediante los datos otorgados por el archivo *csv* y utiliza esa impedancia para la localización de la/s falla/s
2. **Usar Z de UTE:** en ese caso, ILFUS utiliza para la localización de la/s falla/s, la impedancia pre cargada en los datos de la línea.

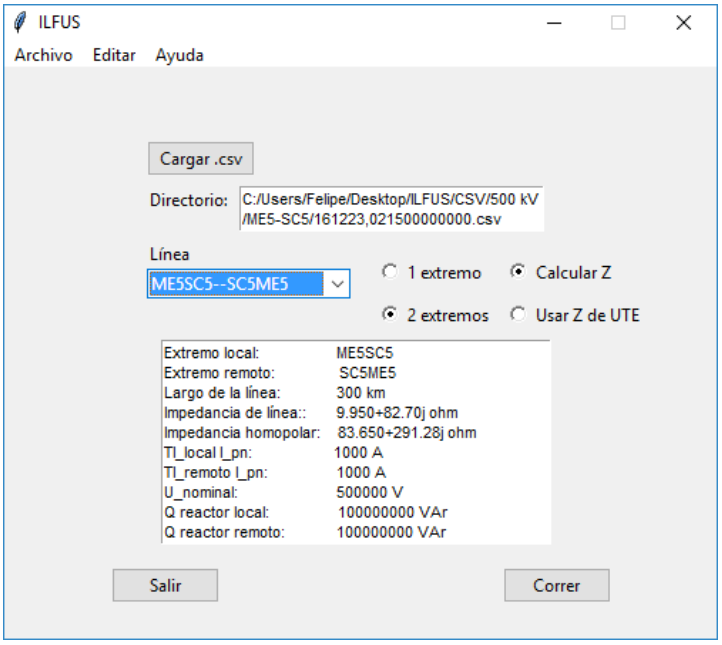


Figura 14.

Paso 6: Presionar el botón Correr

Luego de procesar los datos, mediante una ventana emergente se muestran los resultados obtenidos (Ver sección 4).

4. Interpretación de resultados

Luego de cargar el archivo con fallas, seleccionar las opciones deseadas y correr el programa, mediante una ventana emergente se despliegan los resultados obtenidos. A continuación se presenta un ejemplo de la pantalla de resultados con la descripción de cada parte.

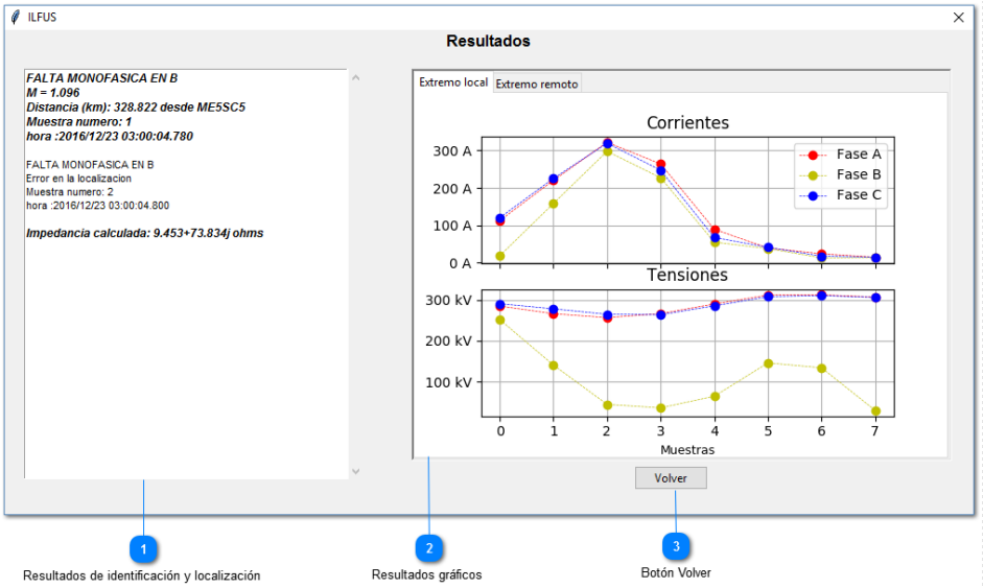


Figura 15.

1. **Resultados de identificación y localización:** En este recuadro se despliegan los resultados obtenidos de los algoritmos de identificación y localización de falla para cada muestra. Es decir, se muestra información sobre el tipo de falla, la distancia a la cual ocurrió la misma desde un extremo[2] y el instante de tiempo en el que fue tomada la muestra. En caso de haber elegido la opción "calcular Z", se despliega además la impedancia de línea calculada. Se resalta en negrita la muestra con mayor precisión en la localización de la falla.

2. **Resultados gráficos:** Se pueden observar las gráficas de corriente y tensión con los valores de muestras medidas durante la falla y sus respectivas referencias. La muestra cero será la primera en la que se encontró la falla y se mostraran las siete siguientes, entre las cuales se encontrará el pico de corriente. En dicho pico se encuentra la muestra más precisa para la localización, salvo en algunos casos en los cuales sera la muestra anterior. La

[2]Las distancias se despliegan en km

muestra más precisa siempre estará resaltada en negrita. Para el caso de líneas con medidas en sus dos extremos, se despliegan dos pestañas de resultados gráficos. En ambas pestañas se representan gráficas de corrientes y tensiones, una para el extremo local y otra para el extremo remoto.

3. **Botón Volver:** Permite regresar a la pantalla principal.

5. Errores y soluciones

En esta sección se contemplan los posibles casos de error y se sugieren procedimientos para evitar los mismos.

1. **Error 1**

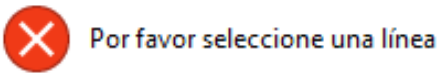


Figura 16.

Este error aparece si no se selecciona una línea de la lista de líneas pre cargadas.

2. **Error 2**

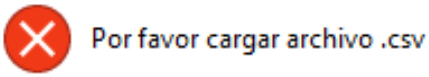


Figura 17.

Este error aparece si no se cargó el archivo con formato "Nombre_archivo.csv".

3. **Error 3**

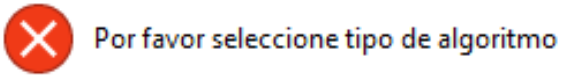


Figura 18.

Este error aparece si luego de cargar el archivo y seleccionar la línea no se seleccionaron las opciones de localización (algoritmos de 1 extremo o 2 extremos).

4. **Error 4**

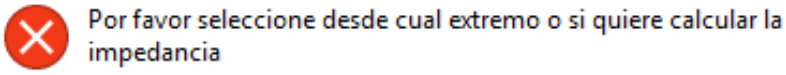


Figura 19.

Este error aparece si luego de cargar el archivo, seleccionar la línea y seleccionar la opción de localización algoritmos de "1 extremo" o "2 extremos", no se seleccionó la segunda columna de opciones.

5. **Error 5**

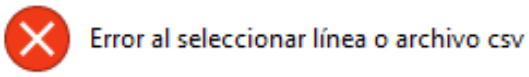


Figura 20.

Este error aparece por varios motivos:

- El archivo cargado no concuerda con la línea seleccionada.
- Se seleccionaron las opciones "1 extremo", "Extremo remoto"para el caso en que la línea seleccionada cuenta únicamente con medidas en el extremo local.
- Se seleccionó la opción "2 extremos"para el caso en que la línea seleccionada cuenta únicamente con medidas en un extremo.
- Al editar o agregar una línea, ingresó mal un dato.

6. **Error 6**

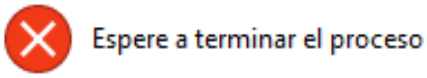


Figura 21.

Este error aparece si luego de correr el programa y mientras el mismo procesa los datos el usuario desea cancelar el proceso.

Para evitar los casos de error 1, 2, 3, 4 y 5, se sugieren seguir los pasos propuestos en los capítulos 2 o 3 de este manual de usuario, según corresponda. Para el caso del error 6, la aplicación no permite cancelar el proceso por lo cual se debe cerrar la ventana de error y esperar a que el proceso finalice.

Esta página ha sido intencionalmente dejada en blanco.

Referencias

[1] Fernando Berrutti. Capítulo 3, repaso líneas de transmisión. In *Transporte de Energía Eléctrica*. Universidad de la República, Facultad de Ingeniería, 2016.

[2] IEEE Std C37.118.1-2011. IEEE Standard for Synchrophasor Measurements for Power Systems., 2011.

[3] IEEE Std C37.118.2-2011. IEEE Standard for Synchrophasor Data Transfer for Power Systems., 2011.

[4] Ricardo Franco. Uso de sincrofasores para la detección de oscilaciones de potencia y pérdidas de sincronismo. Master's thesis, Universidad de la República, Facultad de Ingeniería, 2012.

[5] Ever Benjamin Huerta Leija. Localización de fallas en líneas de transmisión. Master's thesis, Universidad Autónoma de Nuevo León, 2014.

[6] Isi Haim y Mario Vignolo. Capítulo 9, cortocircuitos. In *Redes Eléctricas*. Universidad de la República, Facultad de Ingeniería, 2013.

Esta página ha sido intencionalmente dejada en blanco.

Índice de tablas

Esta página ha sido intencionalmente dejada en blanco.

Índice de figuras

Esta es la última página.
Compilado el martes 23 enero, 2018.
http://iie.fing.edu.uy/

ILFUS
Identificación y Localización de Fallas Utilizando Sincrofasores

Sandino Silva, Agustin Olivera y Felipe Ohaco
Tutor: Celia Sena
Universidad de la República

***Abstract*— Este informe trata del desarrollo de una aplicación para identificar y localizar fallas en líneas de transmisión utilizando información de sincrofasores, recogidas desde una base de datos. Se identifica el o los sincrofasores que detectan la falla, y con esa información se identifica el tipo de falta y la distancia a la misma desde uno de los extremos.**

I. INTRODUCCION

La ubicación rápida y exacta de una falla en una línea de trasmisión es la clave para una mejora en el rendimiento y reducción de los costos de operación y mantenimiento de las mismas. Por estas importantes razones se hace necesario la implementación de una aplicación rápida y eficiente para la localización de las fallas, la cual va a depender de la información obtenida del sistema de protección instalado en la línea.

En todo sistema eléctrico de potencia, los elementos que lo componen están expuestos a fallas. Especialmente las líneas de transmisión están propensas a cortocircuitos debido a la exposición de las mismas a condiciones climáticas así como también debido a sus ubicaciones geográficas.

Por otra parte, los algoritmos de localización deben determinar de forma precisa el punto donde ocurrió la falla. Ésto es de suma importancia ya que de ello depende el tiempo de reparación por parte del personal de mantenimiento.

Contar con un dispositivo que permita localizar el lugar de la falla en una línea de transmisión de manera confiable, permitirá restablecer la línea en el menor tiempo posible, reduciendo el tiempo en el que la misma se encuentre fuera de servicio.

II. RED DE SINCROFASORES

La red de sincrofasores esta integrada por equipos llamados PMU, canales de comunicación y concentradores de datos llamados PDC. En la Fig 1 se observa la red de sincrofasores uruguaya.

Una PMU es un equipo que produce estimaciones de fasores sincronizados a partir de señales de voltaje y corriente y una señal de sincronización horaria.

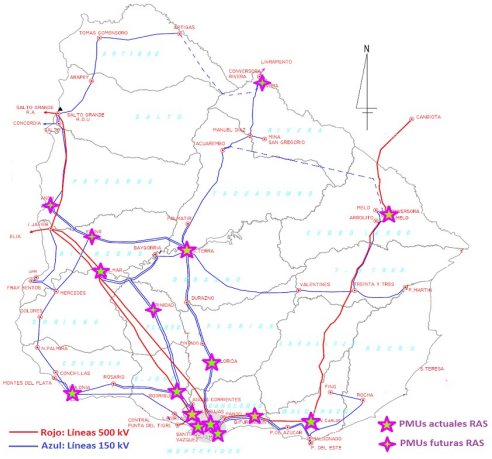


Fig. 1. Red uruguaya de Sincrofasores

Los PDC son los encargados de recibir los datos provenientes de varias PMU, almacenarlos de forma coherente y ponerlos a disposición para el uso de otras aplicaciones.

III. MODELOS Y ALGORITMOS

III-A. LINEA DE TRANSMISION

Conocer los parámetros de una línea previo a una falla puede mejorar la precisión en la localización. Partiendo del modelo de parámetros concentrados Fig 2 y utilizando la teoría de los cuadripolos se pueden determinar los parámetros de la línea, utilizando las componentes simétricas de tensión y corriente calculadas a través de los valores medidos en ambos extremos de la línea. Luego la impedancia y admitancia total de la línea resultan:

$$\overline{Z}_L = \overline{Z}_c.sh(\overline{\gamma}L) = \overline{B} \tag{1}$$

$$\frac{\overline{Y}_L}{2} = \frac{\overline{A}-1}{\overline{B}} \tag{2}$$

Donde A y B son las constantes generales del cuadripolo.

Una ves definido el modelo, se implementa el algoritmo, esto es escribir en código de máquina el proceso para

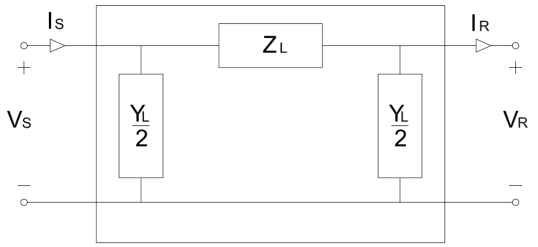


Fig. 2. Modelo de línea con parámetros concentrados

determinar el valor con las medidas. Para implementar el algoritmo, se debe tener en cuenta algunas consideraciones como el sentido del flujo de potencia, presencia de reactores en las líneas y Efecto Ferranti.

III-B. ALGORITMOS DE IDENTIFICCION

Las corrientes de cortocircuitos se caracterizan por un incremento prácticamente instantáneo y varias veces superior a la corriente nominal. Para identificar el tipo de falla, se busca un aumento de la corriente entre medidas consecutivas de cada fase superior al 10 % de In acompañado de una caída de tensión de cada fase superior al 20 % de Vn. Luego para identificar entre fallas bifásicas y bifásicas a tierra se observa la corriente homopolar.

$$\begin{aligned} \frac{\Delta I_x}{I_n} &= \frac{I_i - I_{i-1}}{I_n} > 0{,}1 \\ \frac{\Delta V_x}{V_n} &= \frac{V_i - V_{i-1}}{V_n} < -0{,}2 \end{aligned} \tag{3}$$

III-C. ALGORITMOS DE LOCALIZACION

Para los casos de líneas que presenten medidas de sincrofasores en un solo extremo, se estudiaron e implementaron tres algoritmos distintos.

- Reactancia simple
- Takagi
- Novosel

Cada algoritmo parte del mismo circuito monofásico equivalente (Ver figura 3) obteniendo la ecuación 4 y hace diferentes aproximaciones para reducir el error a causa de no conocer los valores de resistencia de falla (R_F) y de corriente de falla (I_F).

$$\overline{V_S} = m\overline{Z_L}\,\overline{I_S} + R_F\overline{I_F} \tag{4}$$

El algoritmo que presenta menor error en la localizacón para los casos analizados es el algorítmo de Takagi. En la ecuación 5 se observa el cálculo de la distancia al punto de falla.

$$m = \frac{Im\left\{\overline{V_S}\,\overline{I^*_{sup}}\right\}}{Im\left\{\overline{Z_L}\,\overline{I_S}\,\overline{I^*_{sup}}\right\}} \tag{5}$$

I_{sup} representa la corriente superpuesta, definida como la corriente de falla menos la corriente de pre-falla.

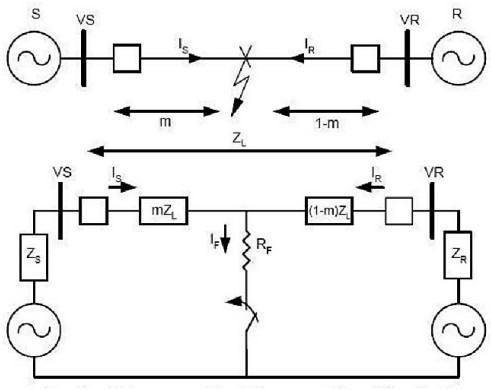


Fig. 3. Equivalente monofásico de un sistema de potencia en falla

Para los casos de líneas que presenten medidas de ambos extremos, permite al algoritmo independizarse del valor desconocido de la R_F. Por este motivo el algoritmo de dos extremos es muy preciso en localización de fallas.

Aplicando las Leyes de Kirchoff al circuito de la figura 3 y despejando m se obtiene:

$$m = \frac{V_S - V_R + Z_L.I_R}{Z_L(I_S + I_R)} \tag{6}$$

IV. CASOS DE PRUEBA

Se elabora un plan de pruebas de software para atender los objetivos de calidad en el desarrollo de sistemas. Se han contrastado los resultados obtenidos por la aplicación usando bases de datos de fallas reales, con sus respectivas identificaciones y localizaciones. Estos resultados se mostraran con sus respectivos errores, verificando si cumplen los objetivos del proyecto.

IV-A. IMPEDANCIA DE LINEA

En la primera instancia de la prueba se verificó el correcto funcionamiento del algoritmo que determina la impedancia de la línea. Se determinaron los parámetros de 6 líneas de 500kV. En la tabla I se pueden observar los 6 casos estudiados.

Nombre	$Z_{UTE}(pu)$	$Z_{ILFUS}(pu)$
MA5 - MB5	$0,00012 + j0,00109$	$0,00108 + j0,00129$
MA5 - MI5	$0,00020 + j0,00213$	$0,00016 + j0,00243$
MA5 - PA5	$0,00269 + j0,02523$	$0,00251 + j0,02538$
BR5 - PA5	$0,00271 + j0,02252$	$0,00362 + j0,02249$
MI5 - SC5	$0,00129 + j0,01359$	$0,0111 + j0,01335$
ME5 - SC5	$0,00398 + J0,03308$	$0,00373 + j0,02953$

TABLE I
DETERMINACIÓN IMPEDANCIA DE LÍNEA

IV-B. ALGORITMOS DE IDENTIFICACION Y LOCALIZACION

La 2da etapa de la prueba es verificar el correcto funcionamiento de los algoritmos de identificación y

localización. Se cuenta con 10 casos de fallas reales. A continuación se analiza una falla en la línea Montevideo A - Palmar.

- Extremo local: Montevideo A - MA5.
- Extremo remoto: Palmar - PA5
- Tipo de falla: monofásica
- Fase en falla: B
- Largo de línea L: 228,9Km
- Distancia d a la falla desde extremo local en Km: 64,9

Se logró identificar una falla monofásica en la fase B, localizada a $70,5Km$ desde el extremo local MA5 utilizando el algoritmo de dos extremos.

Para determinar el error cometido en la medida se utiliza la ecuación del error 7, donde d_r es la distancia de la falla calculada por los relé (se asume como distancia real a la falla en este análisis) y d_c la distancia calculada.

$$\delta = \frac{|d_r - d_c| * 100}{d_r} \tag{7}$$

En este caso analizado obtenemos un error relativo en la medida de $8,2\,\%$, cumpliendo los objetivos planteados.

V. LENGUAJE DE PROGRAMACION

El lenguaje de programación de propósito general, orientado a objetos, utilizado para el desarrollo de la aplicación es Python. Se optó por este lenguaje por ser poderoso y muy práctico. Cuenta con estructuras de datos eficientes y de alto nivel y un enfoque simple pero efectivo a la programación orientada a objetos. Su sintaxis es fácil de entender puesto que es cercana al lenguaje natural, y los programas hechos en Python parecen pseudocódigos, lo cual brinda una gran ayuda en su mantenimiento. Esto hace que Python sea un lenguaje ideal para el desarrollo rápido de aplicaciones en diversas áreas y sobre la mayoría de las plataformas.

VI. APLICACION ILFUS

VI-A. ASPECTOS GENERALES

La aplicación ILFUS está diseñada para identificar y localizar fallas en líneas de transmisión de 150kV y 500kV, siendo fácil su ampliación a otros niveles de tensión. El modo de uso de la aplicación es muy intuitivo y de fácil manejo, aunque deben tenerse en cuenta algunas consideraciones:

- ILFUS permite cargar archivos únicamente en formato "*csv*", el cual deberá contener las medidas de sincrofasores durante el tiempo de la falla. Dicho archivo debe por lo menos contener 5 minutos antes de que comience la falla y el tamaño del mismo influirá en el tiempo de procesamiento de ILFUS.
- Cuando se editen o agreguen nuevas líneas, hay que prestar especial atención en las etiquetas con nombres de las mismas. Para que el software pueda procesar los datos del archivo cargado, la etiqueta con el nombre de línea en el archivo debe coincidir con la etiqueta de la línea pre cargada. Además hay que prestar especial atención al ingresar los parámetros de las líneas (Las unidades de cada parámetro están especificadas) ya que una correcta localización depende de ello. Los números decimales deben estar representados con punto en lugar de coma (Ej. 12.34). Al ingresar una impedancia usando números complejos, el formato debe corresponder con a+bj.
- La aplicación cuenta con dos métodos de localización de fallas (1 extremo y dos extremos). Quedará a criterio del usuario elegir cual método es el adecuado en cada caso para obtener un mejor resultado de la estimación del punto de falla.
- Para el caso de líneas que cuenten con medidas en los dos extremos, ILFUS dará la opción de realizar la localización usando los datos de impedancia pre cargados en la lista de líneas de transmisión o calculando la impedancia de línea con las medidas extraídas del archivo *csv*.

VI-B. INTERFAZ GRAFICA

En la Fig 4 se observa la pantalla principal de la aplicación.

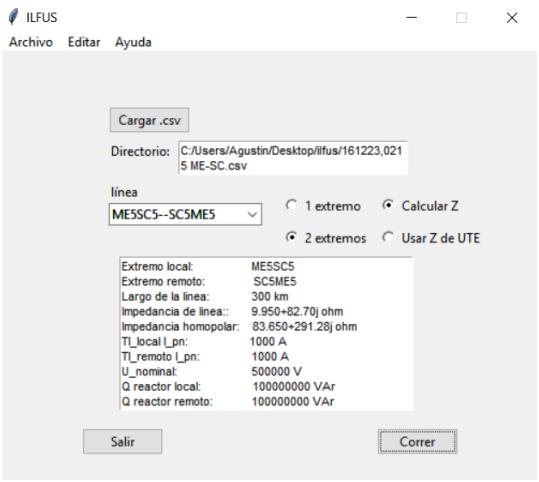


Fig. 4. La fase varia entre 180 y -180 grados

VII. CONCLUSIONES

- Para lograr los objetivos propuestos, primero se estudió el modelo de una línea de transmisión como cuadripolo pasivo y luego se desarrolló un algoritmo para determinar sus parámetros, únicamente para los casos que se posean medidas en ambos extremos de la línea.
  Para realizar una correcta implementación del algoritmo, se debió tener en cuenta los sentidos del flujo de carga, realizando correcciones en las

medidas de corriente de uno de los extremos de la línea, con el objetivo de adecuar el sentido de la corriente al modelo planteado. Se tuvieron en cuenta los efectos generados por Efecto Ferranti en la líneas de transmisión y para el caso de líneas que posen reactores, se consideraron las corrientes aportadas por éstos.
Una vez implementado el algoritmo para determinar los parámetros de la línea en la aplicación, se realizaron pruebas con archivos extraídos de las bases de datos de líneas con medidas en sus dos extremos, obteniendo resultados correctos en todos los casos. Estos resultados fueron comparados con los valores de impedancias utilizados por UTE, siendo estos valores similares.

- Para identificar los distintos tipos de fallas que ocurren en las líneas, se implementó un algoritmo, el cual consiste en encontrar un aumento de la corriente entre medidas consecutivas acompañado de una caída de tensión en las fases en falla.
  Se realizaron pruebas de este algoritmo con casos de fallas reales y se logró identificar las fallas correctamente en todos los casos.

- Se estudiaron tres métodos de localización de fallas en líneas con medidas en un solo extremo, método de la Reactancia Simple, Takagi y Novosel.
  Se realizaron pruebas de los tres métodos con casos de fallas reales. En base a los resultados obtenidos se puede concluir que el algoritmo de la Reactancia fue el que presento el mayor porcentaje de error en la mayoría de los casos de prueba, mientras que los algoritmos de Takagi y Novosel mostraron porcentajes de errores similares menores a los de la Reactancia.
  En la mayoría de los casos analizados el algoritmo de Takagi fue el más preciso, aunque el de Novosel también cumplía con el objetivo.
  Por lo tanto el algoritmo de Takagi es el que mejor se adecúa al cálculo de la ubicación de la falla para líneas con medidas en un solo extremo, siendo éste el que utilizará la aplicación.
- El método más preciso para localizar fallas en una línea de transmisión es el que utiliza medidas de los dos extremos de la línea. Ésto se debe a que no considera la impedancia de falta. Se implemento el algoritmo para la localización y se probó con casos de falla reales.
  Se cuentan con dos casos reales de fallas en líneas de $500kV$. En ambos casos se logró localizar el punto donde ocurre la falla con éxito, logrando obtener un error en la medida menor al 10 %.
- En general, para líneas con medidas en uno y dos extremos, las muestras donde se alcanza el pico de corriente de falla, son las que presentan el menor porcentaje de error en la medida. Por lo tanto, este es el valor más adecuado para indicar la distancia al punto de la falla.
- En la etapa de procesamiento de datos para identificar y localizar fallas, algunas muestras deben ser descartadas en el análisis ya que los valores de tensión y corriente en esas muestras no representan exactamente la falla, por ejemplo la muestras correspondiente al instante en que comienza la falla y el momento en que abre el equipo de protección en los extremos de la línea ya que la corriente de falla comienza a disminuir. Utilizando estas muestras para localizar e identificar
- Se creó una interfaz que utiliza los algoritmos antes mencionados y procesa los datos de los sincrofasores, los cuales se extraen de la base de datos de UTE. Dicha interfaz cuenta con una lista de líneas de transmisión de UTE y sus características ya precargadas. También tiene la funcionalidad de modificar los parámetros, agregar nuevas líneas y eliminar líneas. La aplicación está escrita en el lenguaje Python, el cual puede ejecutarse en varios sistemas operativos. En este caso, se realizó un ejecutable para que la aplicación se utilice en Windows.

References

[1] IEEE Std C37.118.1-2011. IEEE Standard for Synchrophasor Measurements for Power Systems, 2011.

[2] IEEE Std C37.118.2-2011. IEEE Standard for Synchrophasor Data Transfer for Power Systems. 2011.

[3] Fernando Berrutti. Capitulo 3, repaso líneas de transmisión. In Transporte de Energía Eléctrica. Universidad de la República, Facultad de Ingeniería, 2016.

[4] WISCONSIN UNIVERSITY OF WISCONSIN-MADISON. Smart Grid Applications of Wide-Area Measurements. TAB 2 Phasor Measurements and Synchrophasors.

[5] WISCONSIN UNIVERSITY OF WISCONSIN-MADISON. Smart Grid Applications of Wide-Area Measurements. TAB 3 PMU Standar: C37.118

[6] WISCONSIN UNIVERSITY OF WISCONSIN-MADISON. Smart Grid Applications of Wide-Area Measurements. TAB 4 PMU Communications: C37.118 and IEC 61850-90-5

[7] WISCONSIN UNIVERSITY OF WISCONSIN-MADISON. Smart Grid Applications of Wide-Area Measurements. TAB 14 Fault Location

[8] Ricardo Franco. Uso de sincrofasores para la detección de oscilaciones de potencia y pérdidas de sincronismo. Masters thesis, Universidad de la República, Facultad de Ingeniería, 2012.

[9] Ever Benjamin Huerta Leija. Localización de fallas en líneas de transmisión. Master's thesis, Universidad Autónoma de Nuevo Leon, 2014.

[10] Isi Haim y Mario Vignolo. Capítulo 9, cortocircuitos. In Redes Eléctricas. Universidad de la República, Facultad de Ingeniería, 2013.

[11] Python Software. Administrado por la Python Software Foundation. Posee una licencia de código abierto, denominada Python Software Foundation License.

ILFUS

Identificación y Localización de Faltas Usando Sincrofasores

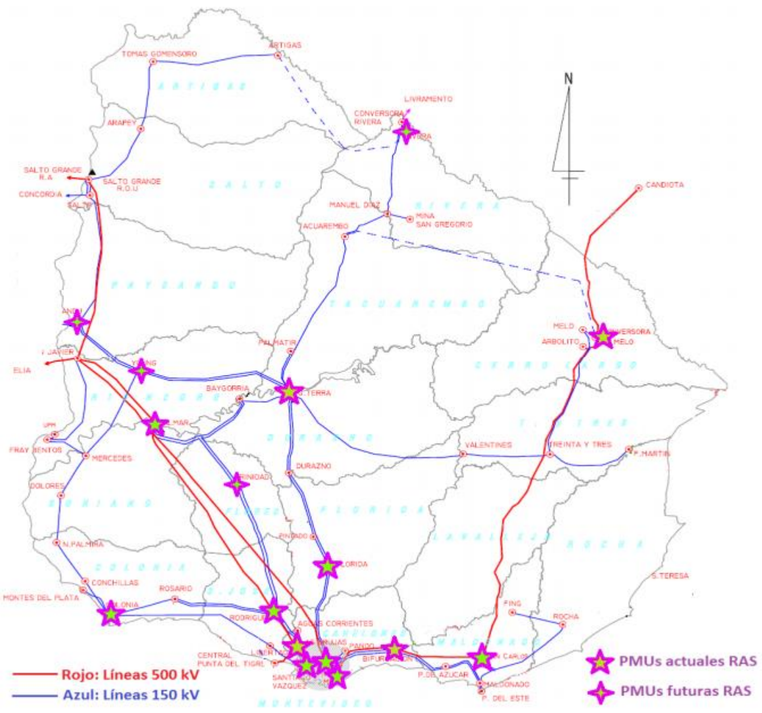
Agustin Olivera, Felipe Ohaco, Sandino Silva

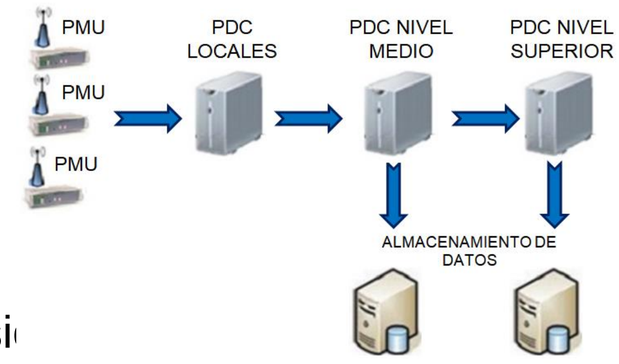
Instituto de Ingeniería Eléctrica

ingeniería de Muestra 2017 ★ Ingeniero Eladio Dieste

Motivación

La ubicación rápida y exacta de una falla en una línea de transmisión, es la clave para mejorar el rendimiento y reducir los costos de operación y mantenimiento de la misma. Por estas razones se hace necesario contar con una aplicación eficiente para localizar fallas. Actualmente los sistemas de protección permiten mediante el uso de las PMU (Phasor Measurement Unit), medir fasores de tensión y corriente en forma sincronizada teniendo como referencia la hora UTC (Universal Time Coordinated), estos son llamados sincrofasores. Los sincrofasores son almacenados en una base de datos centralizada o concentrador de sincrofasores PDC (Phasor Data Concentrators).



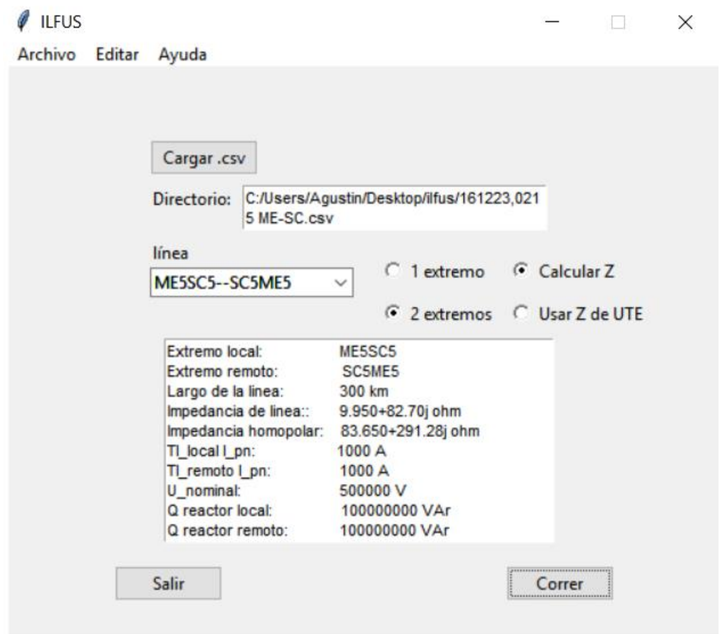


Introducción

La aplicación ILFUS está diseñada localizar fallas en líneas de transmisi 500kV, siendo fácil su ampliación a otros niveles de tensión. Utiliza algoritmos de identificación capaz de identificar diferentes tipos de fallas, algoritmos de localización capaz de localizar fallas con datos obtenidos de un extremos o ambos extremos de la línea.

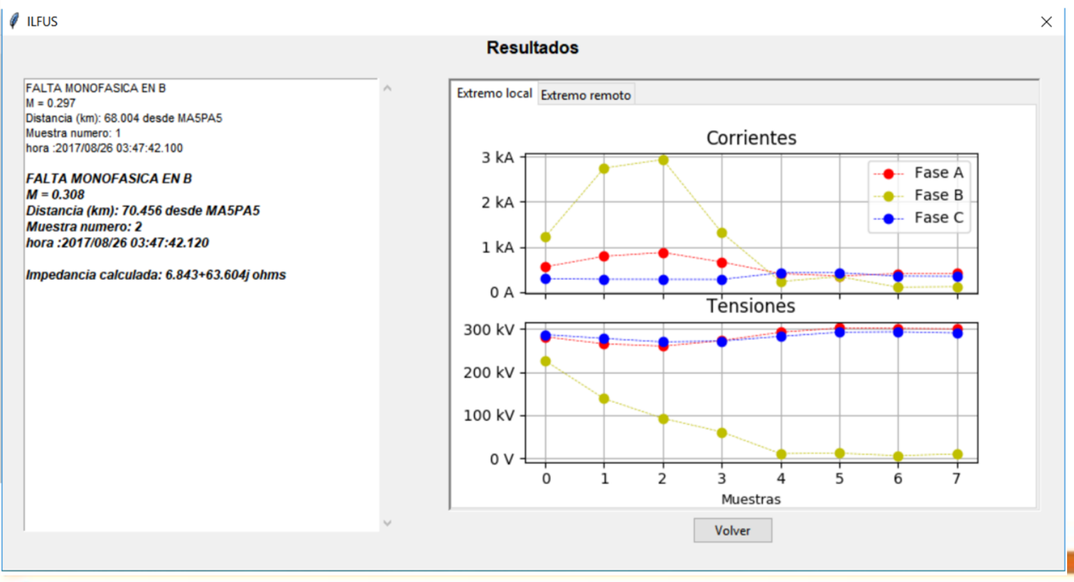
Características

El modo de uso de la aplicación es muy intuitivo y de fácil manejo. Se deberán cargar archivos únicamente en formato ” csv”, cada archivo deberá contener las medidas de sincrofasores de tensión y corriente durante el tiempo de la falla. La aplicación cuenta con una lista pre cargada con las características de las líneas de transmisión de 150kV y 500kV. El usuario podrá modificar los parámetros de una línea, los reactores presentes en las lineas y eliminar o agregar una nueva línea a la lista.



Ventana de resultados

Se despliegan los resultados obtenidos de los algoritmos de identificación y localización de falla para cada muestra. Se muestra información sobre el tipo de falla, la distancia a la cual ocurrió la misma desde un extremo, el instante de tiempo en el que fue tomada la muestra y se despliega además la impedancia de línea calculada. Se pueden observar los gráficos de corriente y tensión con los valores de muestras medidas durante la falla y sus respectivas referencias.



Fundación Julio Ricaldoni — Ingeniería en el Uruguay

Facultad de Ingeniería

Universidad de la República Uruguay