DESARROLLO DE UNA HERRAMIENTA PARA EL ANÁLISIS DE CORTOCIRCUITO EN SISTEMAS DE DISTRIBUCIÓN BASADA EN PYTHON/ATP

Presentado por

FABIO IGNACIO LEÓN SANGUINO

DIEGO ARMANDO VARGAS REYES

Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones
Facultad de Ingenierías Fisicomecánicas
Universidad Industrial de Santander
Bucaramanga
2017

DESARROLLO DE UNA HERRAMIENTA PARA EL ANÁLISIS DE CORTOCIRCUITO EN SISTEMAS DE DISTRIBUCIÓN BASADA EN PYTHON/ATP

Presentado por

FABIO IGNACIO LEÓN SANGUINO

Trabajo de grado para optar por el título de

Ingeniero Electricista

DIEGO ARMANDO VARGAS REYES

Trabajo de grado para optar por el título de

Ingeniero Electricista

Director

Iván David Serna Suárez

Magister en Ingenieria Eléctrica

Codirector

Gilberto Carrillo

Doctor Ingeniero Industrial

Codirector

Gabriel Ordonez Plata

Doctor Ingeniero Industrial

Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones

Facultad de Ingenierías Fisicomecánicas

Universidad Industrial de Santander

Bucaramanga

2017

Este trabajo esta dedicado a mi madre, quien desde el inicio de mi carrera profesional y de mi vida misma a sido mi motivación principal, a mi hijo Samuel a quien espero dar el mejor ejemplo y a Sandy por acompañarme en todo momento.

–Diego Vargas

Este trabajo esta dedicado a mi Padres por su constante apoyo durante mi vida, a mi familia que han sido importantes en el trabajo hecho por mis Padres durante mi formación académica.

–Fabio León

AGRADECIMIENTOS

Primero darle gracias a Dios por darme vida, salud y sabiduría para afrontar el día a día.

A mi familia que sin su apoyo incondicional no podría haberlo logrado, en especial a mi madre Blanca y mi hermana Angela quien me han dado todo el amparo posible aún cuando la situación estaba dificil.

A mis amigos y compañeros, quienes formaron parte fundamental en todo esta etapa de aprendizaje, mil gracias a cada uno que tuve el privilegio de conocer.

A el profesor Iván Serna, por su paciencia y apoyo.

–Diego Vargas

Agradezco a Dios quien me ha dado la inspiración necesaria para vivir correctamente y siempre me ha dado salud.

A mi familia que sin su apoyo no hubiese logrado continuar durante las etapas difíciles de mi vida.

A mi amigo y compañero de proyecto de grado, Diego A. Vargas R., quien siempre fue constante con su trabajo durante este proceso.

A Pilar Luna quien siempre me animó a no rendirme nunca.

A mis amistades que estuvieron a mi lado durante este proceso de formación académica.

Y por último al director de proyecto por su apoyo y paciencia durante el desarrollo de este trabajo.

–Fabio León

Contenido

LISTA DE FIGURAS

LISTA DE TABLAS

Lista de anexos

RESUMEN

TÍTULO:

Desarrollo de una herramienta para el análisis de corto circuito en sistemas de distribución basada en Python/ATP.[1]

AUTOR:

Fabio Ignacio León Sanguino

Diego Armando Vargas Reyes[2]

PALABRAS CLAVE:

ATP, Python, Bokeh, Numpy, Sistemas de distribución, Análisis de fallas, equivalente de Thévenin.

DESCRIPCIÓN:

El análisis de sistemas de distribución ha ido evolucionando gracias al desarrollo tecnológico en equipos de medición, debido a que ofrecen recopilar información en tiempo real. Esto junto a las ventajas de las herramientas informáticas disponibles en la actualidad, permiten manejar gran volumen de información de manera eficiente. A modo de aprovechar esta información, el trabajo actual presenta una herramienta modular para el análisis de sistemas de distribución, construida usando los programas ATP y Python. El software permite al usuario conectar a un sistema un elemento modular en uno o varios nodos, realizar un barrido de variables y por último visualizar los resultados a través de una interfaz gráfica. Las capacidades que posee la herramienta son probadas en dos escenarios: En el caso del primer escenario, se muestran las funcionalidades básicas; mientras que en el caso del segundo escenario se muestra como puede ser extendida la herramienta. Ambos casos se centran en el análisis de cortocircuito de sistemas trifásicos de distribución de energía eléctrica. El método computacional desarrolla un rápido manejo de los datos y velocidad

[1] Trabajo de grado.

[2] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Electrónica y de Telecomunicaciones. Director: Iván David Serna . Codirectores: Gilberto Carrillo, Gabriel Ordonez

en los cálculos realizados gracias a las librerías de Python (Numpy) y la contribución de la velocidad de ATP (aporte de GNU y las propiedades del FORTRAN).

ABSTRACT

TITLE:

Development of a tool for short circuit analysis in distribution systems based on Python/ATP [3]

AUTHORS:

Fabio Ignacio León Sanguino

Diego Armando Vargas Reyes[4]

KEY WORDS:

ATP, Python, Bokeh, Numpy, distribution system, fault analyzes, Thévenin equivalent.

DESCRIPTION:

The power distribution system analysis has been evolving thanks to the technological development of measurement equipment, which offer a way to capture information in real time. This and the advantages of the available tools for information processing, allows the efficient processing of big amounts of information. As a way to take advantage of such information, in this work is presented a modular tool developed for power distribution analysis based on the computer platforms ATP and Python. The tool allows the user the successive connection of an element in several nodes, the parameter sweep function, and result visualization through a graphical interface. The capabilities that the tool possesses are tested with two applications: In the case of the first one shows its basic features and its basic usage with a short-circuit analysis; in the case of the second application shows how the tool can be customized in order to extend its functionalities. Both applications are focused on the short-circuit analysis of three-phase power distribution systems. The computational method allows a fast handling of the data and speed in the calculations made, thanks to the Python

[3] Research work.

[4] Faculty of Physical-Mechanic Engieneering. School of Electrical, Electronical and Telecommunications Engineering. Advisor: Iván David Serna Co-advisor: Gilberto Carrillo, Gabriel Ordonez.

libraries and the contribution of the speed of ATP (contribution of GNU and the properties of FORTRAN).

INTRODUCCIÓN

Cuando una falla en los sistemas de distribución afecta el suministro de energía eléctrica a una empresa o a una residencia, la empresa distribuidora incurre en multas que debe pagar por la interrupción del servicio. En conjunto con el grupo de investigación GISEL se planteó la propuesta de implementar una herramienta que permita efectuar simulaciones y cálculos, con el fin de analizar cambios en sistemas de distribución. Por lo tanto el presente trabajo describe la estructura y aplicación de la herramienta propuesta para analizar variaciones del flujo de carga mediante la inserción de un elemento modular a un sistema de distribución en uno o diferentes nodos consecutivos y variar el valor del elemento insertado para cada nodo de conexión seleccionado, obteniendo la información correspondiente a las simulaciones para todas las variaciones antes descritas.

Como aplicación complementaria del uso de la herramienta, se emplean los datos obtenidos para el cálculo de corrientes de corto-circuito utilizando la matriz de fases de Thevenin para resolver las ecuaciones impuestas por las condiciones de límite de una falla determinada (Dilek *et al.* (2004)). Dicha matriz es calculada perturbando con una carga de prueba el sistema de distribución original, y midiendo el impacto de dichas perturbaciones en las tensiones y corrientes del sistema antes y después de la prueba.

En contraste con otras herramientas similares, ésta fue implementada en un lenguaje de programación de código abierto llamado Python. A través de los capítulos

se mostrarán el uso de Python como herramienta de cálculo y manejo de datos que le dará al usuario la capacidad de realizar simulaciones por medio de ATP y el análisis posterior de los datos obtenidos en una interfaz gráfica interactiva.

La estructura del libro consta de las siguientes capítulos: el primer capítulo describe los trabajos previos realizados para el análisis de fallas basados en ATP y Matlab. El capítulo 2 explica la estructura de la herramienta desarrollada para el análisis de sistemas de distribución mostrando la interacción entre los software Python y ATP. Posteriormente se mostrará la aplicación de la herramienta para así finalmente las conclusiones y los trabajos futuros a realizar.

CAPÍTULO 1

TRABAJOS PREVIOS

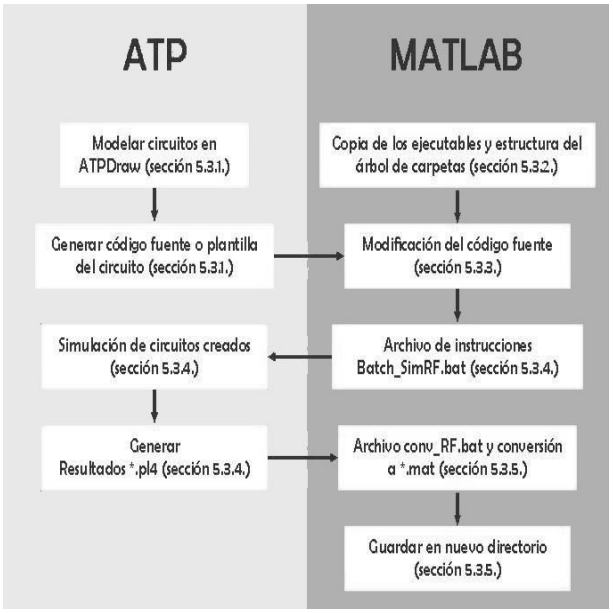
Para el desarrollo de la herramienta software se realizó una búsqueda de documentos con trabajos anteriormente realizados, relacionados al tema de investigación propuesto, con el fin de recopilar información útil para realizar una adecuada implementación y darle eficiencia y precisión a la herramienta.

Con la información recopilada se analizaron las características que ofrecían diferentes plataformas informáticas, por ejemplo en (Jurado *et al.* (2000)) se evalúan dos aspectos: la cantidad de modelos de elementos de circuitos y la flexibilidad y libertad de programación. Adicionalmente se tiene el trabajo realizado en (Ronald J. Villar (2006)) y en (Zhang & Kezunovic (2005)) donde se encuentra el manejo de la interacción que se puede dar entre las plataformas ATP y MATLAB.

También ATPDraw por su libre distribución y su avanzado desarrollo en simulaciones de sistemas es una buena elección para implementar herramientas computacionales cuyo enfoque sea el análisis de los diversos escenarios presentes en sistemas de distribución. MATLAB por su parte, presenta un atractivo al ofrecer flexibilidad y libertad de programación junto a la amplia capacidad de realizar manejo y análisis de datos (Ronald J. Villar (2006)).

En primer lugar se muestra la forma gráfica en la que presentan su proceso (Ronald J. Villar (2006))1.1:

Figura 1.1.: Procedimiento automático para la simulación de fallas utilizando ATP y MATLAB (Ronald J. Villar (2006)).

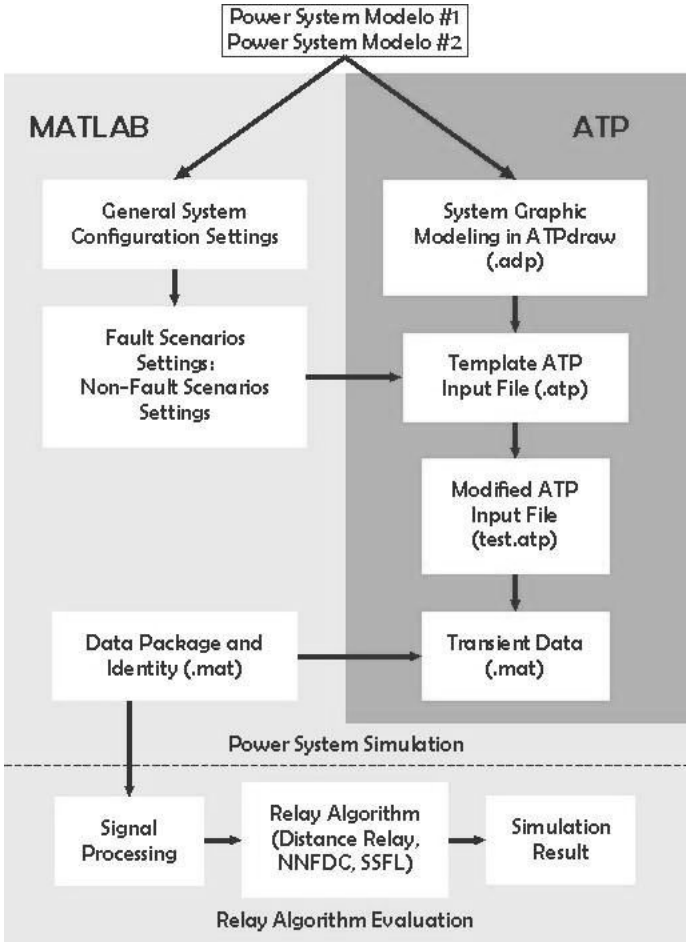


Se observa que el primer paso es modelar un sistema de distribución de energía eléctrica en ATPDraw, cuya función es simplificar la descripción de circuitos según los requerimientos de ATP-EMTP. Gracias a la posibilidad de reducir errores en la implementación de los circuitos a simular, ATPDraw (por ser una interfaz gráfica) permite crear un archivo de extensión *.acp y generar un archivo de extensión .atp, que será la plantilla del cual se crearán archivos copia de extensión .atp para cada tipo de falla en el sistema que se desee simular, realizando los cambios necesarios en las líneas del código fuente en *.atp. Para ejecutar todas las simulaciones utilizaron un archivo de instrucciones de procesamiento por lotes de extensión .bat. Para ello crearon en matlab un programa con el que copiaron los archivos runATP.exe, startup.ini y pL42mat.exe en cada una de las carpetas creadas para cada falla. También crean un archivo $Batch\_SimRF.bat$ en cada carpeta con el que automáticamente se ejecutan los archivos .atp (con $runATP.exe$) generando otros archivos de extensión .lis, .dbg, .tmp, .bin y .pl4 y de estos archivos se desechan todos menos el de extensión .pl4 el cual se mueve a un nuevo directorio archivos *.PL4. Después de lo anterior, los autores crean otro archivo de instrucciones llamado $conv\_RF.bat$ mediante la programación en matlab de una función que hace uso del programa pl42mat.exe que

ejecuta los resultados de las simulaciones almacenados en archivos de extensión *.pl4 para generar archivos con extensión .mat y moverlos a un nuevo directorio archivos *.MAT. Todo el proceso conlleva la generación y tratamiento de una cantidad de archivos que exige tener buena organización y para ello los autores presentan una función llamada directoriosRF programada en matlab con la cual crean un árbol de carpetas.

De igual manera en (Zhang & Kezunovic (2005)) se presenta de manera gráfica, un marco sintetizado de su herramienta.

Figura 1.2.: Marco sintetizado del software desarrollado.(Zhang & Kezunovic (2005))



En este caso, MATLAB se utiliza como una herramienta de control y la interfaz para encontrar automáticamente el componente del sistema y el establecimiento de los parámetros para cada simulación. Las mediciones de tensión y corriente para cada

escenario se disponen y se guardan para el análisis de algoritmos. ATP se utiliza para ajustar los modelos de sistemas de plantilla y la ejecución de la simulación debido a su mayor velocidad.

Esta herramienta de simulación consta de dos partes principales: simulación de sistemas de potencia y evaluación del algoritmo. En la primer parte, el modelo de sistema de potencia de interés se construye en ATPDraw como un archivo plantilla atp, los componentes del sistema y sus parámetros se establecen en MATLAB, el usuario a través de la interfaz en MATLAB define los escenarios de perturbación. Luego el programa MATLAB carga el archivo plantilla .atp y crea un archivo temporal mediante la modificación de los ajustes del archivo plantilla .atp. Después de que se ejecuta ATP las medidas transitorias en cada escenario se almacenan para la evaluación de otros algoritmos para análisis de fallas. En la segunda parte, los datos brutos obtenidos a partir de las mediciones son pre-procesados de acuerdo con los requisitos del algoritmo. Los datos procesados se analizan por diferentes algoritmos y los resultados del análisis se comparan con las características reales de los escenarios. Cuando se aplica para diferentes modelos del sistema, el software sólo tiene que reconstruir el modelo de plantilla de ATP y actualizar la configuración del sistema en MATLAB.

Los documentos (Ronald J. Villar (2006)) y (Zhang & Kezunovic (2005)) muestran como aprovechar la interactividad de las dos plataformas. Ambas herramientas poseen estructuras similares puesto que la base para el desarrollo de las herramientas es la unión de ATP y MATLAB.

Otra información que se estudió y se tuvo en cuenta durante el desarrollo de la herramienta fue (Dilek *et al.* (2004)) para implementar los diferentes modelos de fallas de corto circuito y sus correspondientes cálculos. Los autores, mediante equivalentes de Thevenin, realizan los cálculos de corrientes de falla y voltajes del sistema de distribución, ellos presentan la construcción de la matriz de fases de Thevenin para nodos con conexión a tierra fig.1. y nodos sin conexión a tierra fig.3. junto a su respectiva evaluación de fallas fig.2. fig.4.

Figura 1.3.: La construcción de PTM a un nodo N tierra. (a) Tensiones antes de introducir cualquier carga de prueba. (b) Una carga de prueba se inserta entre la fase A y tierra. (c) Una carga de prueba se inserta entre la fase B y tierra. (d) Una carga de prueba se inserta entre la fase C y tierra. (Dilek *et al.* (2004)).

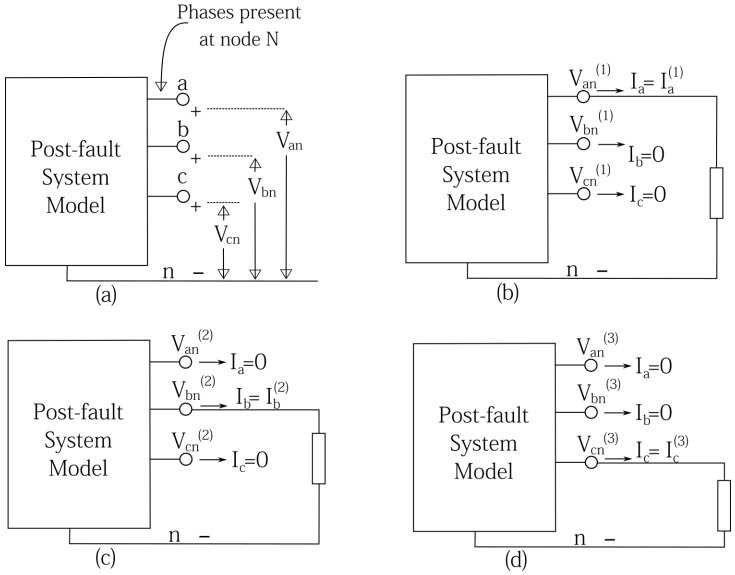
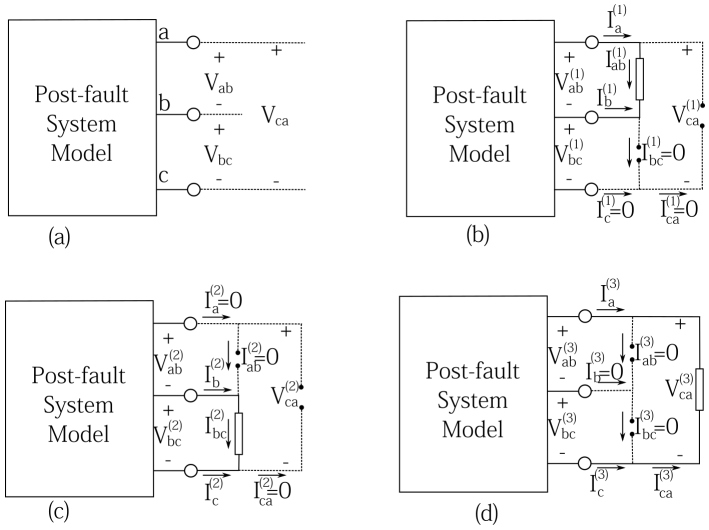


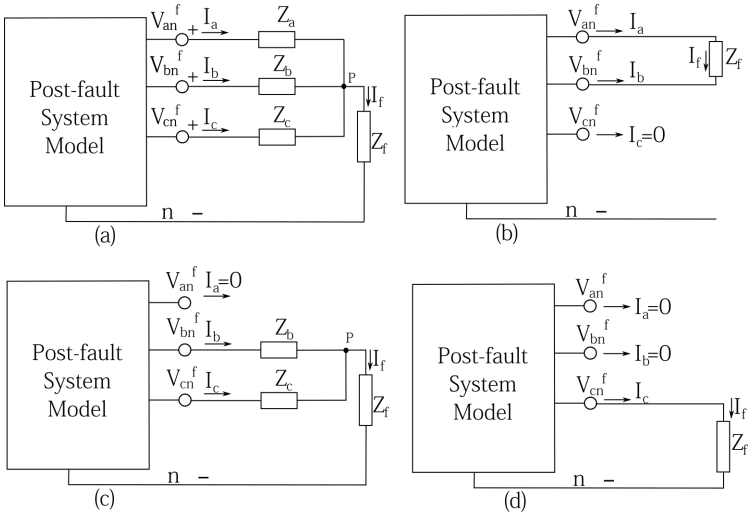
Figura 1.4.: La construcción de PTM en un nodo N sin conexión a tierra. (a) Voltajes antes de insertar alguna carga de prueba. (b) Una carga de prueba es insertada entre la fase A y la fase B. (c) Una carga de prueba es insertada entre la fase B y la fase C. (d) Una carga de prueba es insertada entre la fase C y la fase A. (Dilek *et al.* (2004)).



Para los nodos con conexión a tierra se estudiaron los casos en los que se presenta una falla trifásica a tierra, una falla fase-fase, una falla bifásica a tierra y una falla

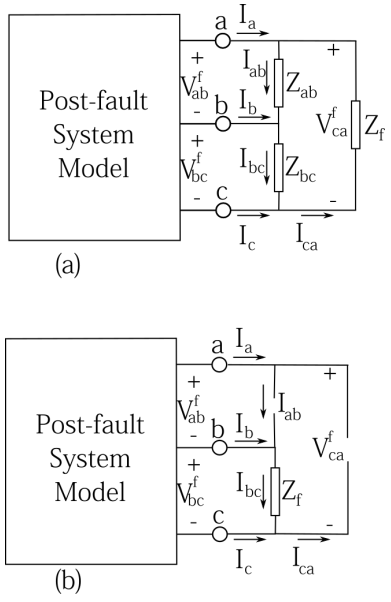
fase-tierra.

Figura 1.5.: Varias fallas en un nodo N a tierra. (a) falla trifásica a tierra. (b) falla fase a fase. (c) falla bifásica a tierra. (d) falla fase a tierra. (Dilek *et al.* (2004))



Para los nodos sin conexión a tierra las fallas estudiadas son falla trifásica, falla fase-fase.

Figura 1.6.: Dos tipos de fallas en un nodo N sin conexión a tierra. (a) Una falla trifásica. (b) una falla fase a fase. (Dilek *et al.* (2004)).

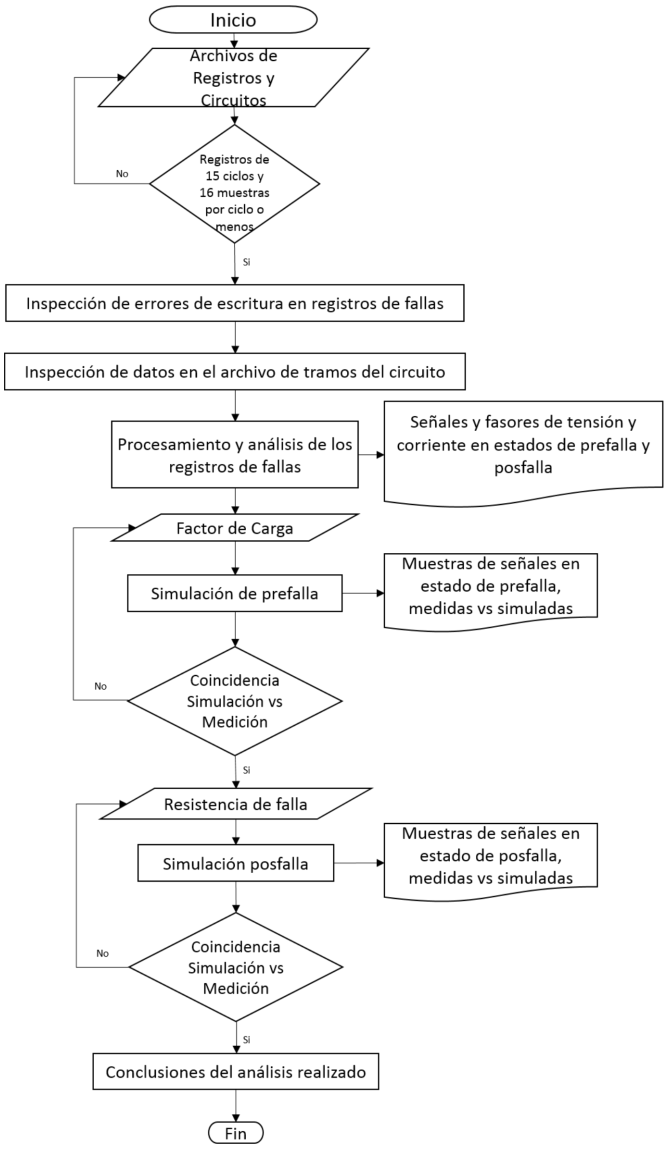


Los autores señalan que el procedimiento en sistemas aterrizados y no aterrizados para manejar los cálculos de falla en un nodo es el mismo. Este procedimiento consiste

en: obtener los voltajes de pre-falla en el nodo, determinar los límites impuestos por la falla dada, expresar los voltajes finales en términos de las impedancias de falla y corrientes de falla correspondientes a la conexión de falla dada y por último resolver. Esta metodología propuesta por (Dilek *et al.* (2004)) presenta una ventaja a la hora de implementarla por su sencillez ya que emplea datos sólo de estado estable (disponibles desde software para el cálculo de flujo de cargas).

En (Camargo León, 2012) se muestra el uso de la plataforma MATLAB junto a la plataforma de ATP con la que realiza simulaciones de los circuitos y sistemas, para la validación de la información obtenida de instrumentos de medición en un sistema de distribución con el fin de analizar, caracterizar y seleccionar los datos referentes a fallas presentes en el sistema y poder realizar tratamiento y ajuste en los datos de tensión y corriente obtenidos en la cabecera del sistema de distribución. Acorde con lo planteado dicho trabajo, se debe realizar primero una simulación del circuito en estado de pre-falla y una segunda simulación utilizando los datos proporcionados por los medidores. Estas dos simulaciones junto con los datos y comportamiento real del circuito son lo que conforman la metodología (Figura 1.7), para realizar la validación de la información obtenida en tiempo real.

Figura 1.7.: Metodología de Validación. (Camargo León, 2012)



Camargo León, 2012 presenta una herramienta software que en investigaciones futuras puede ser un complemento potencial a la herramienta presentada en este libro, ambas están enfocadas hacia las fallas en sistemas de distribución y su desarrollo busca encaminar futuras investigaciones de localización y determinación del tipo de fallas ocurridas en los sistemas de distribución.

CAPÍTULO 2

ESTRUCTURA DE LA HERRAMIENTA

Para el desarrollo de la herramienta se dispuso del lenguaje de programación de código abierto llamado Python en su versión 3.5.2, el cual es el encargado de hacer la gestión de archivos, lectura de las plantillas y manejo de datos. Por otra parte se utiliza el Alternative Transient Program (ATP), para simular los diferentes sistemas de distribución y generar el flujo de carga, obteniendo los datos necesarios corrientes y tensiones a graficar. Tanto Python como ATP son software sin costo alguno lo que ofrece un beneficio adicional.

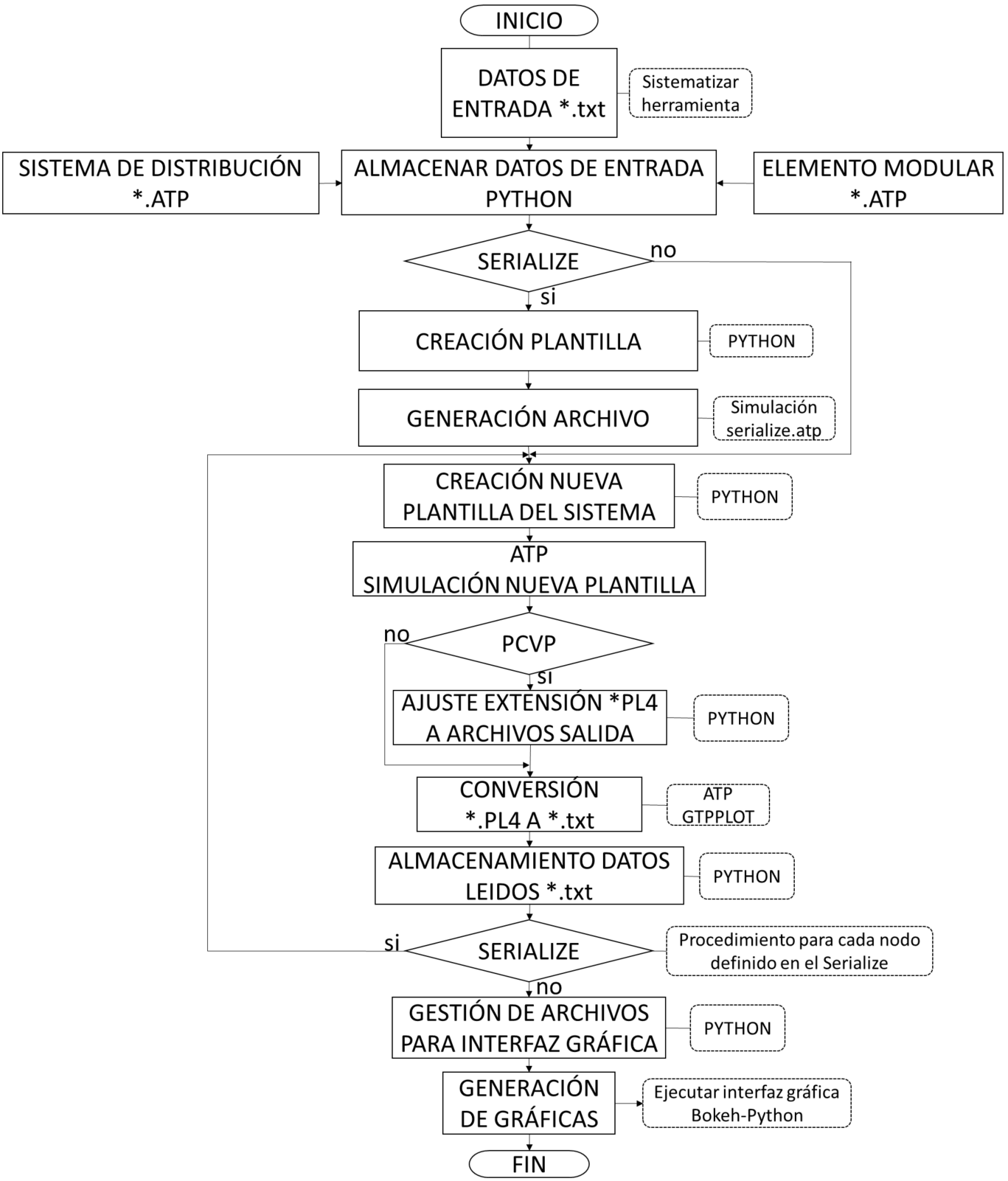
La combinación de estos dos elementos da como resultado una herramienta modular, capaz de simular sistemas de distribución agregando a este un elemento o modulo y permitiendo hacer variación en el valor del elemento insertado y nodo de conexión de dicho elemento.

En la figura 2.1 se ilustra el proceso de ejecución de la herramienta. En ella se puede observar las diferentes etapas de procesamiento de los archivos. El uso de este software requiere determinados requisitos, así mismo de configuraciones especificas para la entrada y procesamiento de los datos. Toda esta información se encuentra el la sección de anexos en Manual de usuario (Anexo B). Allí se especifica desde las versiones de los software a utilizar, tanto de las condiciones de funcionamiento de la herramienta y sus alcances.

A continuación se explica una a una las etapas de procesamiento de datos, de forma

sistematizada en la herramienta, donde se hace referencia al sistema Pre-evento, para los valores del sistema antes de insetar el elemento modular y el sistema Pos-evento una vez ya se ha incorporado dicho elemento.

Figura 2.1.: Proceso de simulación en software basado Python-ATP



2.1 ELEMENTOS DE ENTRADA

Para esta primera etapa el software implementado en Python, realiza un preprocesamiento con la configuración para sistematizar la simulación por parte de funciones de ATP, los parámetros de entrada del elemento modular y el archivo del sistema objeto del análisis, éstos dos últimos en formato *.atp. Esto se ejecuta mediante Python en la lectura de archivos plantilla *.atp, manejando estos como una cadena de caracteres facilitando el procesamiento de la información ahí contenida tanto en el sistema de análisis y el elemento modular. Por otra parte las opciones de sistematización y otras especificaciones se encuentran contenidas en un archivo de texto plano (*txt), en el cual se selecciona el tipo de sistematización a realizar. Cada una de estas opciones se explicarán a continuación.

Las funciones ATP «pocket calculator varies parameter» (PCVP) y el «Serialize», permiten sistematizar la herramienta se implementan por medio de la función «$Parameter», que se utiliza para declarar una variable o parámetro a un elemento especifico para de esta manera poder operarlo con las otras dos funciones.

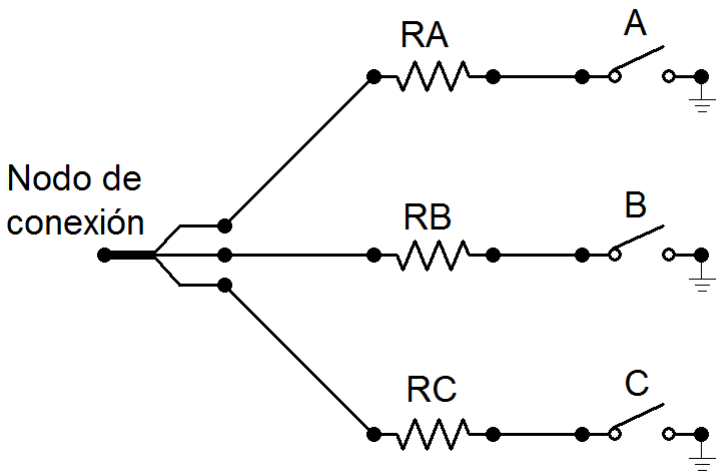
El PCVP proporciona la capacidad de asignar una cantidad determinada de valores a un parámetro del elemento modular mediante la asignación de valor del «KNT», de modo que éste aumenta proporcionalmente desde un valor inicial hasta un valor final estipulado por un rango establecido por la ecuación del «KNT» a realizar.

Por otra parte la función «serialize» posibilita cambiar el nombre del nodo de conexión del elemento modular, mediante la creación de una variable denifida en el manual de usuario (Anexo B), para renombrar el nodo de unión. La función solo permite cambio del parámetro con numeración consecutiva, es por eso que solo se varía entre nodos definidos en el sistema de distribución con numeración consecutiva. Para información detallada de la entrada de datos revisar el Manual de usuario (anexo B).

El elemento modular corresponde a una plantilla *atp que contiene el componente a

ser insertado al sistema de distribución, . Como ejemplo se presenta un modelo básico de falla (figura2.2), donde se configura hasta tres fallas fase-tierra, estableciendo los tiempos de apertura y cierre de los interruptores unidos a las resistencias de falla. En el ejemplo se tienen asignados los parametros RA, RB, RC para variar cada resistencia un valor especifico o un rango de valores.

Figura 2.2.: Modelo RF para el análisis de fallas



El código Python permite que el usuario pueda crear su propia plantilla *modelo.atp* en ATPDraw, para así insertar al sistema un elemento modular que sea de su interés, de este modo se podra visualizar los cambios en el sistema ante una gran cantidad de modelos posibles para implementar.

El sistema de distribución objeto del análisis se debe tener en formato *.atp, y renombrarse como sistema.atp con letras minúsculas. Debe contener características descritas en el Manual de usuario (anexo B).

Este archivo debe estar contenido en una carpeta dentro del ordenador, junto al archivo *modelo.atp* y todos los archivos complementarios de sistema.atp requiera en el caso de tener bases de datos incluidas. En dicha carpeta se consignará los resultados obtenidos del uso de la herramienta.

2.2 AJUSTE Y SIMULACIÓN NUEVAS PLANTILLAS

Como primera medida, el código Python realiza la lectura de las plantillas *.atp correspondientes, *sistema.atp* y *modelo.atp* almacenando los datos ahí contenidos, así mismo los datos consignados como parámetros de entrada de la simulación. Con esta información la herramienta se dispone a crear las plantillas modicadas para los sistemas pos-evento con el fin de generar mediante la simulación en atp los fujos de carga.

Antes de la modificación de dichas plantillas, si el estudio incluye usar la función «Serialize», la herramienta en Python crea una plantilla adicional generando una simulación previa,.Con esto se obtiene el archivo *dofile.dat* que contiene la platilla *modelo.atp* modificada n veces acorde al número de nodos definido. Posteriormente con Python se extrae la información del archivo de extensión *.dat.

Para la serialización de los nodos del sistema, como se creó y ejecuto previamente el archivo base *Serialize.atp* (el cual contiene la definición de la función «serialize» seguido del contenido del elemento modular llamado *modelo.atp*). Se ejecuta dicho archivo para generar un elemento llamado *Dofile.dat* que contiene un listado del contenido del fichero *modelo.atp* con el nodo de conexión serializado N de vecescomo cantidad de nodos estipulados. La gestión de este listado se realiza con Python mediante la modificación de una plantilla *sistemapos-evento.atp* para cada nodo de conexión.

Mediante la creación de un archivo nuevo llamado *Sistemapos-evento.atp*, en Python se modifica los elementos pertenecientes al *sistema.atp*, agregando a este lo contenido en la plantilla *modelo.atp*, siguiendo todas las directrices necesarias para que ATP no genere ningún error. Agregando adicionalmente las funciones de Atp a la simulación. Para el uso de «serialize» el código Python usa el archivho *Dofile.dat* que contiene el modelo serializado.

Una vez Python ha creado el archivo *sistemaPos-evento.atp*, se ejecuta de manera sistematizada generando así sus archivos salida o respuesta. La ejecución de las plantillas de ATP desde Python se realiza en simultaneidad con el archivo *runATP.exe* el cual es copiado por python directamente en la carpeta de análisis, Python se encarga de ejecutar uno a uno las plantillas ATP sobreescribiendo el mismo archivo pero por cada plantilla contenida en el archivo *Dofile.dat*, que corresponde a la cantidad de nodos definida con «Serialize», en caso de usar solo el PCVP se crea una sola plantilla que genera N simulaciones igual al valor del «KNT».

Para los análisis en donde se usa el PCVP, las salidas generadas por el ATP se encuentran sin formato y seguidas de un consecutivo que va desde 001 hasta el número total de «KNT» especificadas en los parámetros de entrada para la herramienta. Mediante funciones de Python se modifican estos archivos para asignarles la extensión *.PL4, necesaria para la siguiente etapa del código.

En caso de no usar el PCVP, la salida de la simulación del flujo de carga al tener un único valor la variable, el archivo de salida saldrá con la extensión incluida.

Dado que los datos almacenados en las salidas se encuentran en formato *.PL4, es necesario convertir a otro formato para facilitar la lectura y almacenamiento de la información ahí contenida. Por esto dentro de la instalación del paquete del ATPDraw, se encuentra una gran herramienta llamada «GTPPLOT». Esta aplicación me permite para el caso de estudio convertir los datos cifrados en la extensión *.PL4 a un archivo de texto plano que contiene toda la información de cada uno de los elementos de la simulación de los flujos de carga a la velocidas y propiedades de GNU FORTRAN. Estos se realiza mediante el comando «RELAY ALL» perteneciente al «GTPPLOT», ahí permite asignar un nombre especifico con la extensión requerida.

Para el uso del PCVP se genera tantas salidas en formato *.PL4 como «KNT» asignadas para un archivo sistemapos-evento.atp, es por esto que para estos casos, la herramienta en Python crea un archivo de texto plano llamado *ordenes.txt*, el cual contiene toda una serie de comandos a ser leídos por el «GTPPLOT» para realizar

toda la conversión de los archivos de manera sistematizada.

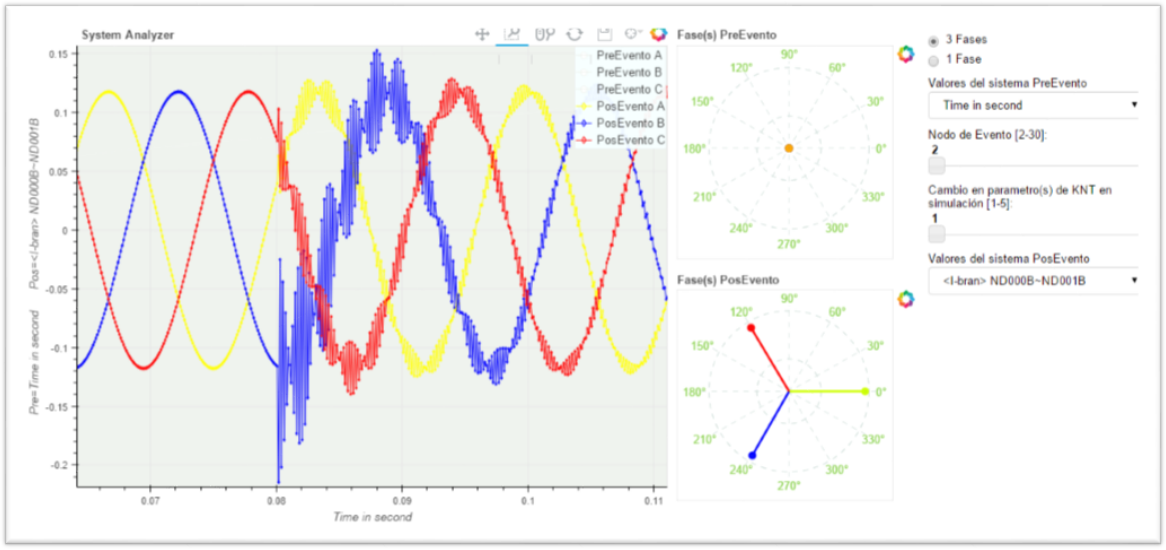
Una vez el «GTPPLOT» ha generado los archivos de texto plano que contienen toda la información de las simulaciones, estos datos son leídos y almacenados en Python de manera temporal mientras termina la ejecución de la herramienta. Para el caso de uso de la función «serialize», en Python ya se encuentra almacenada la información del archivo *Dofile.dat* que contiene una lista de N modelos acorde a M simulaciones realizadas de «KNT», donde Python repetirá todo el proceso anterior referente a la simulación previa, modificando un *sistemapos-evento.atp* para cada nodo de conexión y así Python almacena en una lista cada grupo de datos obtenido por todas las simulaciones de los archivos pos-evento.

2.3 VISUALIZACIÓN DE RESULTADOS

Una vez obtenida toda la información correspondiente para cada simulación realizada y los resultados de los cálculos de fallas (si habilito esta opción), toda esta información será depositada en un archivo codificación binaria datos.out, mediante la libreria de Python (_Pickle), el archivo tendrá como destino una carpeta *«PlotApp»*creada con la finalidad de contener todos los archivos e información necesaria para la ejecución de la herramienta gráfica, creada con la librería Bokeh de Python, la cual ofrece amplias opciones para la ilustración de datos.

Esta interfaz gráfica le permite al usuario interactuar con los las curvas de las diferentes simulaciones, pudiendo visualizar los valores de corriente y tensión para cada uno de los valores del elemento modular en los nodos insertados en el sistema, comparando los cambios derivados de dicha inserción mediantes el uso de mandos de selección(ver Figura 2.3) . El uso de esta interfaz y los controles de selección revisar Manual de usuario (anexo B) de la herramienta.

Figura 2.3.: Interfaz gráfica para análisis de resultados



Es posible guardar la imagen que se esta vizualizando junto a los comandos de desplazamiento en la parte superior de la imagen (Fig 2.4).

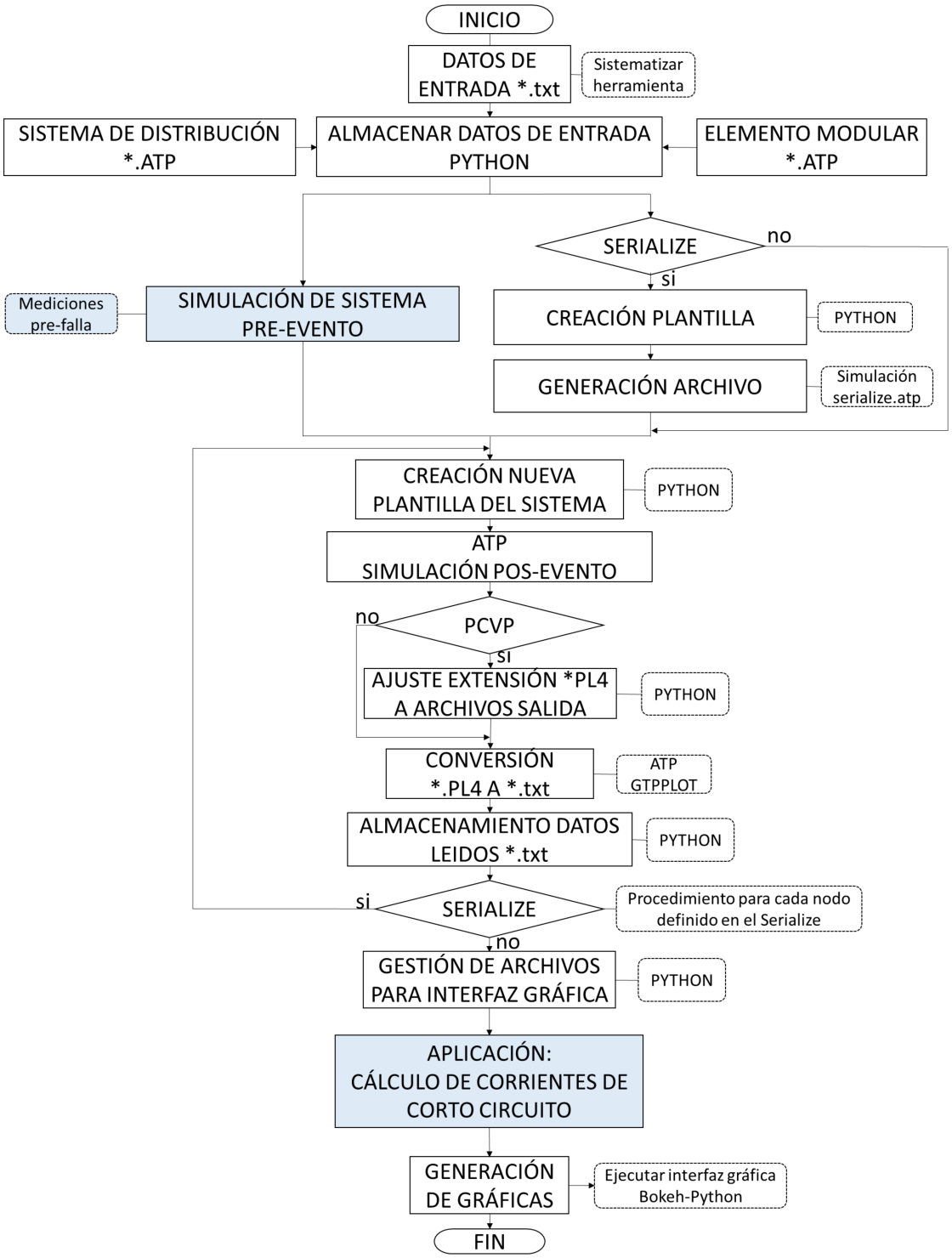
Figura 2.4.: Mandos de desplazamiento para la gráfica.

CAPÍTULO 3

APLICACIÓN

La herramienta tiene como finalidad brindar un entorno modular de simulación para el análisis de sistemas de distribución. Como aplicación adicional se implementó el modelo para el cálculo de corrientes de falla en sistemas de distribución (descrito en Dilek *et al.* (2004)), mediante la matriz de equivalentes de tensión de Thevenin en el nodo de falla. Finalmente se aplicó la herramienta a un sistema de distribución rural real (anexo A en [Suárez (2011)]. La figura 3.1 representa la secciones añadidas (cuadros sombreados) a la herramienta para el uso del cálculo de corrrientes de corto-circuito.

Figura 3.1.: Proceso de simulación con cálculos de corriente de corto



3.1 SIMULACIÓN CIRCUITO REAL

La modularidad que ofrece la herramienta permite insertar un elemento en diferentes nodos del sistema, y del mismo modo cambiar el valor del elemento N veces en cada uno de los nodos seleccionados, a continuación se emplea dichas funciones para insertar una resistencia de falla en tres nodos diferentes del sistema utilizado y así mismo se varía el valor de la resistencia en tres valores como lo ilustra el cuadro 3.1.

TABLA 3.1.: Parámetros de simulación aplicados

ELEMENTO	PARÁMETROS
Funciones ATP	PCVP, Serialize
Nodos	49 al 51
Resistencia [omhs]	10, 20, 30
Tiempo de Simulación [s]	0.0833
Tiempo falla [s]	0.0415

Por lo anterior la herramienta inserta en el nodo 49 una resistencia de 10 omhs y ejecuta el ATP para obtener los flujos de carga, posteriormente conecta en simulaciones separadas los siguientes dos valores de la resistencia de falla, ésto lo hace tanto para el nodo 49,50 y 51. Así se obtiene el valor de tres corriente de falla para tres nodos diferentes.

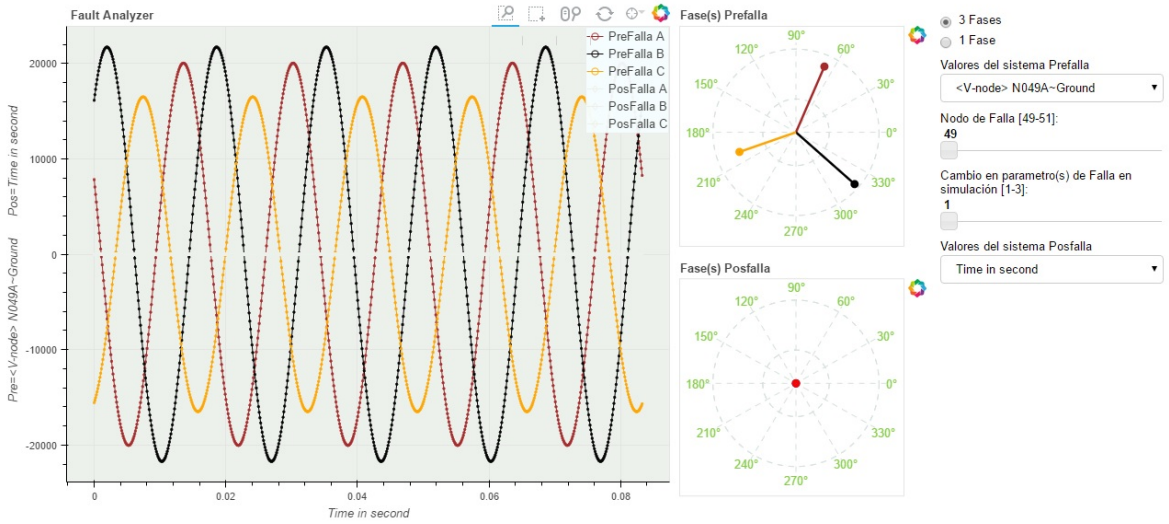
La cantidad de nodos y valores de resistencia se estipularon a manera de ejemplo, ya que no es necesario que la cantidad de nodos sea igual a la cantidad de valores del elemento insertado, la cantidad de nodos la define el usuario acorde a el rango de nodos a analizar y el valor del elemento es estipulado de igual manera en los parámetros de entrada de la herramienta con la asignación del valor del «KNT» para dichas simulaciones.

Para mas información revisar Manual del usuario (anexo B).Una vez ejecutada, se crea una carpeta llamada *PlotApp*, que contiene archivos necesarios para la visualización por medio de la aplicación creada en Bokeh de Python, entre esos se encuentra el datos.out, un archivo de codificación binaria que contiene toda la información correspondiente a las simulaciones realizadas por ATP.

Los resultados de la simulación permiten analizar los cambios ocurridos en el sis-

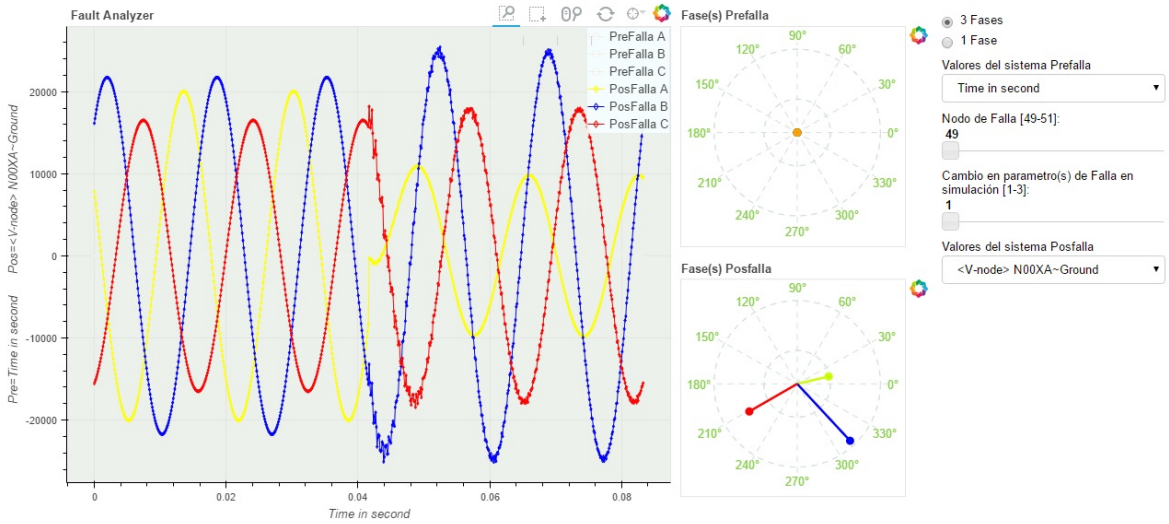
tema por el efecto de la inclusión de elemento modular, facilitando la opción de comparar mediante gráficas las diferencias existentes entre el sistema antes y después de la falla. La visualización de la herramienta se obtiene ejecutando el archivo *PlotApp.bat* contenido en la carpeta *PlotApp*, la cual se inicia en el explorador por defecto que utilice el equipo.

Figura 3.2.: Resultados herramienta Python/ATP. Tensión Nodo 49 sin falla



Como se ilustra en la Figura 3.2, se observan la tensión trifásica en el nodo 49 sin conectar la resistencia de falla, por medio de los controles mostrados a la derecha de la imagen, es posible cambiar la gráfica a visualizar, seleccionando la tensión o corriente a mostrar. Junto a la gráfica de curvas de tensión se encuentra en la parte superior central, los fasores de tensión del sistema prefalla. La fase A se identifica con el color café, la fase B con el color negro y la C con el color naranja.

Figura 3.3.: Resultados herramienta Python/ATP. Tensión Nodo 49 con inserción de falla



En la ventana se selecciona mostrar la gráfica de la tensión en el nodo 49 al insertar la resistencia de falla con un valor de 10 [omhs] en el instante de tiempo igual a 0.0415[s] (Fig 3.3), permitiendo observar como la fase A donde se introduce la falla, disminuye su valor pico de tensión considerablemente mientras la fase B y C se incrementan, además que las fases sufren un des-balance viéndose mas afectado la fase de falla. El diagrama fasorial de tensiones en el nodo 49 antes de la falla Fig3.5 y corrientes de falla (Fig 3.6). Usando los controles de mando (parte derecha de la imagen), se puede observar los cambios de amplitud y fase de cada una de las señales cambiando el nodo de falla y/o el valor del parámetro insertado.

Ubicando el cursor sobre las señales graficadas, se desplegará un cuadro mostrando el valor y tiempo que tiene el punto especifico señalado, de ésta manera es fácil observar el valor de cada amplitud de las señales graficadas.

La gráfica 3.4, se puede observar el cambio de amplitud que sufre la fase A una vez la resistencia de falla ha sido conectada.

Figura 3.4.: Corriente en la cabecera del circuito. (Pos-evento)

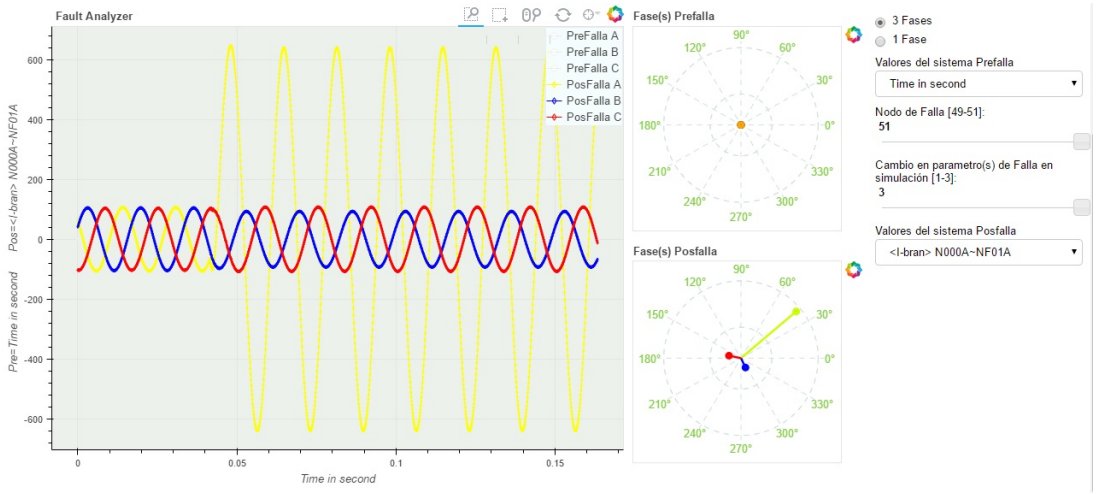
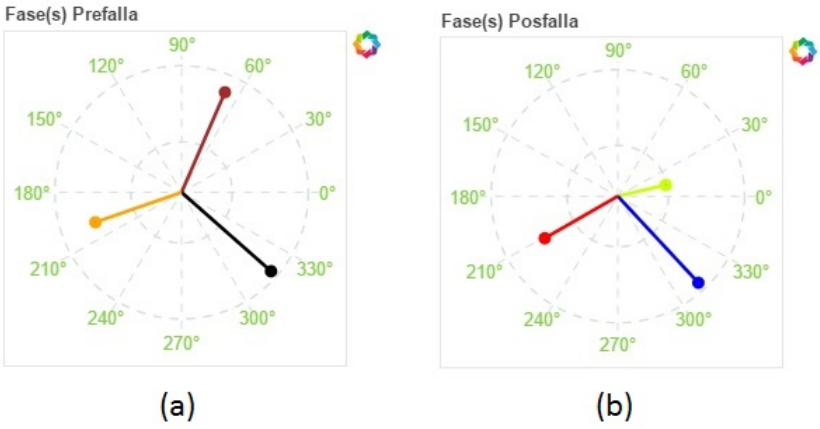


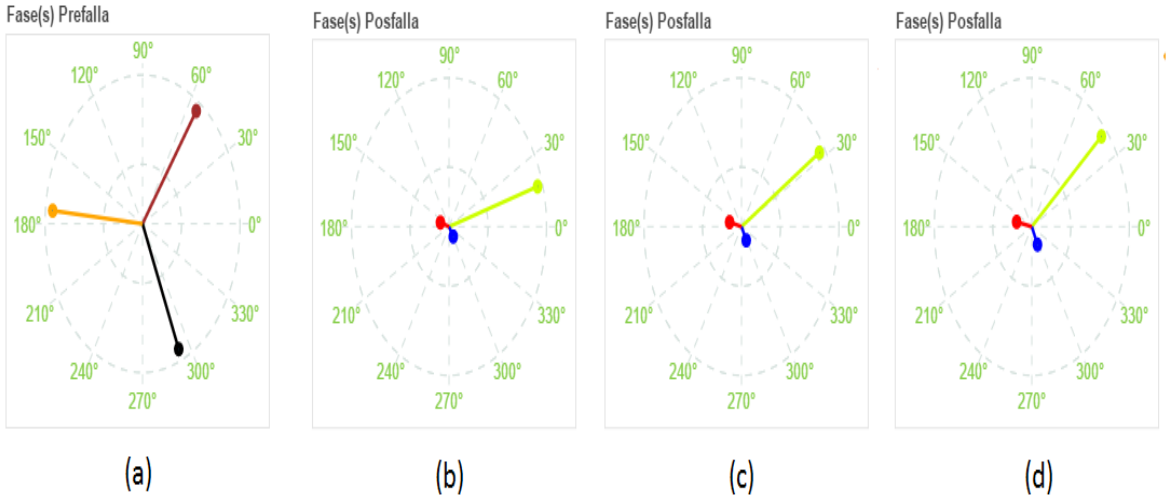
Figura 3.5.: Comparativa diagrama fasorial tensión nodo 49 pre-evento vs pos-evento



0.03

Nota: La imagen (a) tensión en el nodo 49 pre-evento, (b) tensión en el nodo 49 pos-evento

Figura 3.6.: Comparativa diagrama fasorial corrientes pre-evento vs pos-evento



Nota: imagen (a) fasores de corriente pre-evento nodo 49, (b) fasores de corriente pos-evento nodo 49 y RF=10, (c) fasores de corriente pos-evento nodo 49 y RF=20 y (d) fasores de corriente pos-evento nodo 49 y RF=30

Los valores obtenidos de la simulación para los nodos y valores de resistencia son consignados a continuación.

TABLA 3.2.: Resultados de simulación tensión nodo

Nodo	Resistencia de Falla [ohm]	Fase	Tensión Prefalla [V]	Tensión Posfalla [V]
49	10	A	20010	9820
49	10	B	21730	25537
49	10	C	16510	17920
49	20	A	20010	14900
49	20	B	21730	24710
49	20	C	16510	16760
49	30	A	20010	16700
49	30	B	21730	24170
49	30	C	16510	16320
50	10	A	20011	10940
50	10	B	21710	25480
50	10	C	16500	17960
50	20	A	20011	14860
50	20	B	21710	24730
50	20	C	16500	16770
50	30	A	20011	16690
50	30	B	21710	24160
50	30	C	16500	16380
51	10	A	20040	10920
51	10	B	21720	25540
51	10	C	16500	18070
51	20	A	20040	14830
51	20	B	21720	24700
51	20	C	16500	16740
51	30	A	20040	16670
51	30	B	21720	24160
51	30	C	16500	16380

Las corrientes de fallas en la fase A obtenidas de ATP se encuentran en el cuadro 3.3.

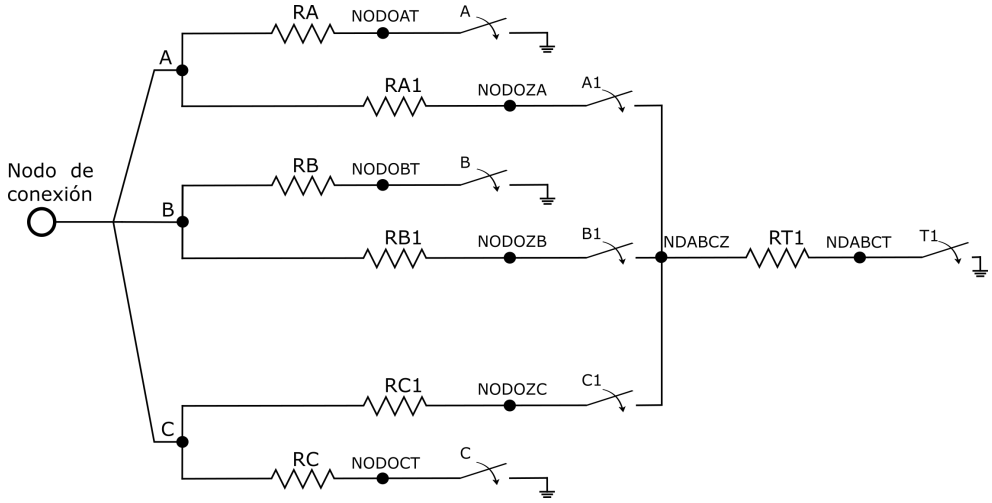
TABLA 3.3.: Corrientes de falla obtenidas ATP, falla monifásica en A

Nodo	Resistencia Falla [ohm]	Corriente de falla [A]
49	10	981.960
49	20	721.368
49	30	551.107
50	10	978.585
50	20	720.034
50	30	549.910
51	10	974.876
51	20	718.477
51	30	549.277

3.2 CÁLCULO DE CORRIENTES DE FALLA

La herramienta de simulación modular se le a dado una aplicación dentro del código Python para el cálculo de las corrientes de falla siguiendo los estándares utilizados en el articulo Dilek *et al.* (2004), donde especifica los métodos de computar las corrientes de falla acorde al tipo de falla y las características del sistema. Dentro de los parámetros de entrada el usuario selecciona si va a utilizar la aplicación añadida para el cálculo de fallas modificando en el archivo *«3-Datos de simulación.txt»* el parámetro «Falla», asignandole un valor entre 1 a 4 (acorde al tipo de falla a realizar revisar anexo manual de usuario) o por otro lado un valor de 0 (cero) para no realizar el cálculo de fallas. El valor por defecto que tiene configurado la herramienta es «Falla=0».

Figura 3.7.: Modelo RF para la aplicación del Cálculo de corrientes de corto circuito.



Para esto Python brinda herramientas computacionales para el cálculo y operación de matrices complejas gracias a la librería Numpy: Continuando con el circuito, para obtener utilizar y obtener la información de los flujos de carga de sistema se a creado una plantilla.atp (Figura 3.7), con la finalidad especifica (configurara en el codigo Python). El procedimiento de cálculo para una falla monofásica en A se describe a continuación:

- El primer paso consisten en insertar una carga de prueba en cada una de las fases en el nodo de estudio, para este caso se inserta la carga de prueba de 10 [omhs] en el nodo 49 y se obtiene la tensión del nodo y corriente de la carga en ese instante. La herramienta obtiene las tensiones Pre-evento del sistema en los nodos de estudio, información que será necesaria.

- Una vez se tenga la información anterior, se calcula la matriz de fases de Thevenin característica del nodo 49. Los valores de la diagonal (a11,a22,a33) son las características propias del nodo por fase, mientras los elementos fuera de la diagonal corresponden al acoplamiento entre fases.

$$\begin{bmatrix} \triangle V_{an} \\ \triangle V_{bn} \\ \triangle V_{cn} \end{bmatrix} = \begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ a31 & a32 & a33 \end{bmatrix} \begin{bmatrix} I_a \\ I_b \\ I_c \end{bmatrix}$$

$$a11 = \frac{V_{an} - V_{an}^{(1)}}{I_a^{(1)}}$$

- El valor V_{an} corresponde a la tensión en el nodo antes de insertar la carga de prueba, $V_{an}^{(1)}$ es la tensión en el nodo de estudio una vez se inserta la carga de prueba y $I_a^{(1)}$es la corriente que pasa por la carga de prueba insertada. La ecuación para calcular la corriente de falla monofásica para la fase A es:

$$I_a = \frac{V_{an}^i}{a11 + Z_f}$$

  En donde I_a es la corriente de corto circuito en el nodo, V_{an}^i es la tensión del nodo antes de conectar la caga de prueba(tensión en vacio del nodo), y Z_f es la resistencia de falla. Se procede a realizar dichos cálculos.

$$a11 = \frac{(-13772{,}6087 + 14564{,}2465j) - (4773{,}0969 + 8568{,}9536j)}{(477{,}3097 + 856{,}896j)}$$

$$a11 = (-3{,}8611 + 19{,}493j)$$

- Finalmente se calcula la corriente de corto en el nodo 49 para una resistencia de falla de 10 [omhs]:

$$I_a = \frac{(-13772{,}6087 + 14564{,}2465j)}{(-3{,}8611 + 19{,}493j)) + 10}$$

$$|I_a| = 980{,}864\,[A]$$

El cuadro3.4 , indica las magnitudes de corrientes calculadas realizando una comparativa con las corrientes obtenidas de ATP las cuales se toman como las corrientes teóricas.

TABLA 3.4.: Comparativa corriente calculadas vs teóricas(ATP)

Nodo	Resistencia de Falla [ohms]	Corriente Calculada [A]	Corriente Teórica [A]	% Error relativo
49	10	980.864	981.960	0.11 %
49	20	721.4930	721.368	0.13 %
49	30	550.4345	551.107	0.12 %
50	10	978.017	978.585	0.05 %
50	20	720.1541	720.034	0.01 %
50	30	549.7092	549.910	0.03 %
51	10	974.773	974.876	0.01 %
51	20	718.5857	718.477	0.01 %
51	30	548.8674	549.277	0.07 %

Los resultados del cálculo de las corrientes de corto será consignado en un archivo de extensión *.csv (Fig 3.8).

Figura 3.8.: Ilustración cálculos de corrientes de corto

GRÁFICAS OBTENIDAS EN ATPDRAW | GRÁFICA FALLAS CALCULADAS

Las gráficas corresponden a los cálculos de falla realizados acorde al tipo de falla seleccionada.

El color para cada fase es Verde=fase A, Azul=fase B y Rojo=fase C.

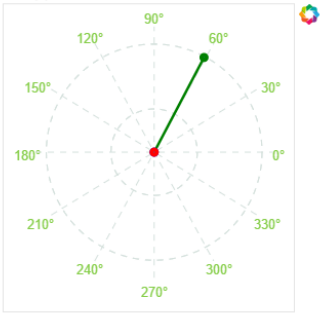
La tabla cambia con respecto al nodo de falla.

Nodo de Falla [49-51]: **49**

Cambio en parametro(s) de Falla en simulación [1-3]: **1**

#	NodoF	KNT	Corriente de Falla	Magnitud	Fase (grados)
0	49	1	[477.30968714+856.89535479j]	980.8639999999999	60.881224819799996
1	49	1	[0.+0.j]	0	0
2	49	1	[0.+0.j]	0	0
3	49	2	[109.58430824-713.12231886j]	721.4930091580001	-81.263791581
4	49	2	[0.+0.j]	0	0
5	49	2	[0.+0.j]	0	0
6	49	3	[96.77465843+541.86049681j]	550.434494298	79.8739085658
7	49	3	[0.+0.j]	0	0
8	49	3	[0.+0.j]	0	0

Fase(s) Posfalla



De la anterior imagen se puede apreciar los valores calculados acorde al método de la matriz de tensiones de Thevenin en un nodo, donde la tabla muestra las corrientes de corto para cada fase, pero en este caso particular como solo se obtienen para falla en la fase A, es por esto que pata KNT igual a 1 salen tres valores donde dos son valores igual a cero correspondientes la fase B y la fase C.

CAPÍTULO 4

CONCLUSIONES

El desarrollo de la herramienta permitió mostrar las capacidades de Python para el rápido cálculo de fallas en sistemas de distribución y desarrollo de aplicaciones, gracias a librerías como Numpy que proporcionan los mecanismos eficientes de manejo y operación de matrices complejas o Bokeh la cual brinda las facultades para la representación interactiva de los datos. Con esto dicho lenguaje de programación queda como una alternativa para reemplazar el uso de software pago tal como Matlab para aplicaciones similares a la aquí desarrollada.

La implementación combinada de las funciones de ATP para la sistematización del software como lo son PARAMETER, PCVP y el SERIALIZE (optimizada con la ayuda de Python), permiten realizar al usuario un estudio más amplio al admitir la posibilidad de cambiar el valor del elemento insertado y así mismo su nodo de conexión en cualquiera del sistema de distribución.

Gracias al manejo proporcionado por Python a las plantillas *.atp, la herramienta ofrece modularidad en el elemento a insertar, posibilitando al usuario crear un componente en ATPDraw y agregarlo al sistema para analizar los cambios que dicho elemento provoca, lo que favocere el estudio de un sistema de distribución en diferentes aspectos dependiendo de la plantilla del modelo creado.

La prueba de la herramienta en el circuito de distribución real se dio por medio de pruebas realizadas a cada uno de los nodos del circuito, teniendo este un total de 257 nodos, realizando pruebas de fallas y comparando dichos valores de corriente

de cortocircuito con las corrientes calculadas por medio del método de la matriz de tensiones de Thevenin.

Para trabajos futuros se sugiere la posibilidad de utilizar Python como lenguaje de programación debido a su gran capacidad, en la creación de herramientas enfocadas en diferentes análisis. Adicionalmente se tiene la posibilidad de mejorar ésta herramienta creando más plantillas, modificando el código Python agregando las opciones y cálculos referentes a las nuevas plantillas creadas, ampliando la capacidad de la herramienta.

BIBLIOGRAFÍA

DILEK, MURAT, BROADWATER, ROBERT, & SEQUIN, RICHARD. 2004. Computing Distribution System Fault Currents and Voltages via Numerically Computed Thevenin Equivalents and Sensitivity Matrices.

JURADO, FRANCISCO, ACERO, NATIVIDAD, CARPIO, JOSÉ, & CASTRO, MANUEL. 2000. Using Various Computer Tools in Electrical Transients Studies.

LEÓN, ANDRÉS FELIPE CAMARGO. 2012. *Localización de fallas: Gestión de información obtenida en equipos de protección instalados en cabeceras de circuitos de distribución.*

RONALD J. VILLAR, FELIX A. JAIMES. 2006. *Caracterización de circuitos de distribución para estudios de calidad en sistemas de energía eléctrica.*

SUÁREZ, IVÁN DAVID SERNA. 2011. *Estrategia para la validación de modelos empleados en sistemas de distribución.*

ZHANG, NAN, & KEZUNOVIC, M. 2005. Implementing an Advanced Simulation Tool for Comprehensive Fault Analysis. *In: Transmission and Distribution Conference and Exhibition: Asia and Pacific, 2005 IEEE/PES.*

ANEXOS

ANEXO A

MARCO TEÓRICO

A.1 FUNCIONES ATP

Se presentan las funciones de ATP para realizar simulaciones en los sistemas de distribución con las cuales se sistematiza el proceso de simulación. Estas funciones son «$Parameter», «Serialize» y «Pocket calculator varies parameter», inicialmente se plantea un caso monofásico y a medida que se avanza en el desarrollo de la herramienta, se implementapara el análisis de sistemas trifásicos.

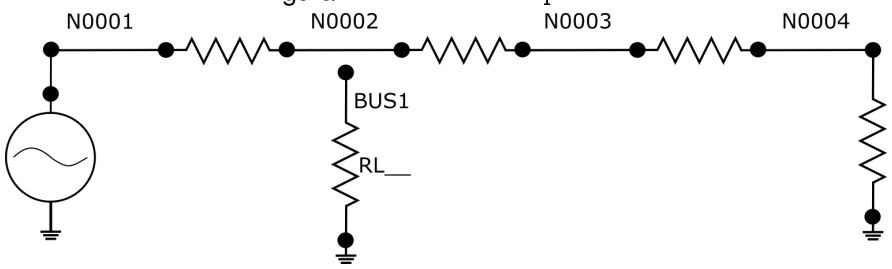
Es una declaración necesaria para definir los símbolos de datos para la implementación de las funciones PCVP y «Serialize» de ATP.

La función PCVP realiza una variación del valor tomado por un parámetro, dicha variación se controla con el valor KNT, con el fin de realziar un barrido de variables. La estructura para la implementación del PCVP en ATPDraw se define de la siguiente manera.

```
/REQUEST
POCKET CALCULATOR VARIES PARAMETERS     KNT
```

```
$PARAMETER
Definir parámetro y fórmula (KNT)
BLANK $PARAMETER
```

Figura A.1.: Circuito de prueba.



Usando ATPDraw se genera un circuito ejemplo monofásico (figuraA.1). La resistencia RL conectada a un nodo del circuito tendra un barrido de valores para este elemento, para esto en la creación del circuito, la resistencia se le asigna un valor RL en vez de un valor numérico para que el programa lo identifique como un parámetro. Ahora se define en la ventana settings de ATPDraw (figura A.3), en la pestaña variables la fórmula que define el valor del parámetro y el número de variaciones. Una vez creado el circuito con todas sus especificaciones se debe generar el archivo *.atp esto se hace en el menú ATP de la barra de herramientas, se elige la opción subprocces y se selecciona la opción Make ATP file (figura A.2).

Figura A.2.: Menu ATPDraw, generación plantilla *.atp

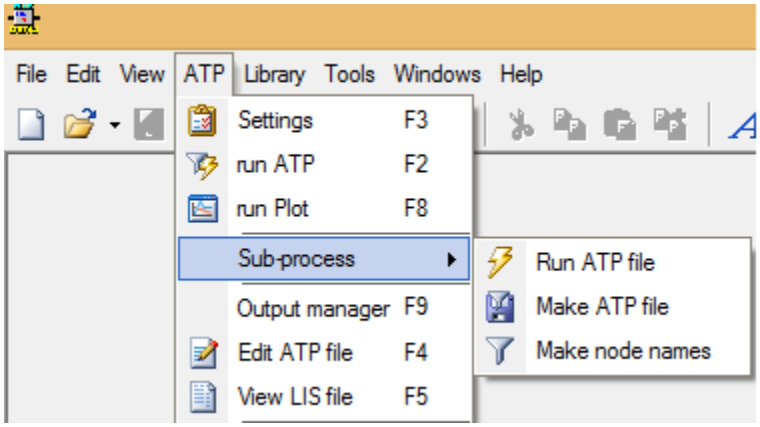
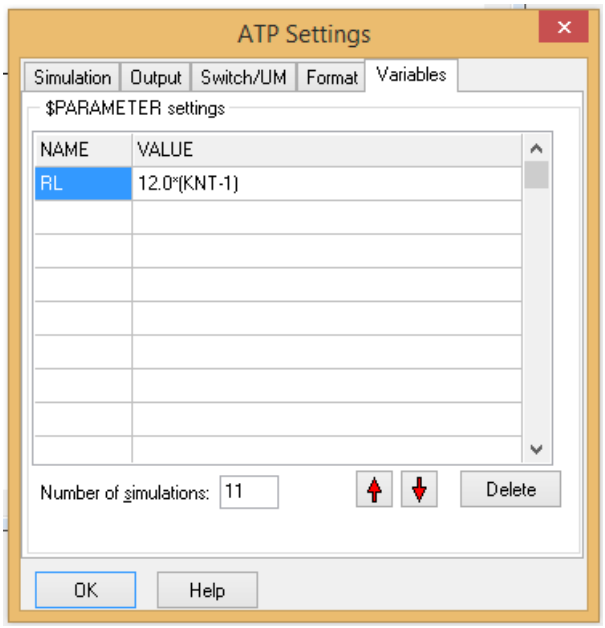
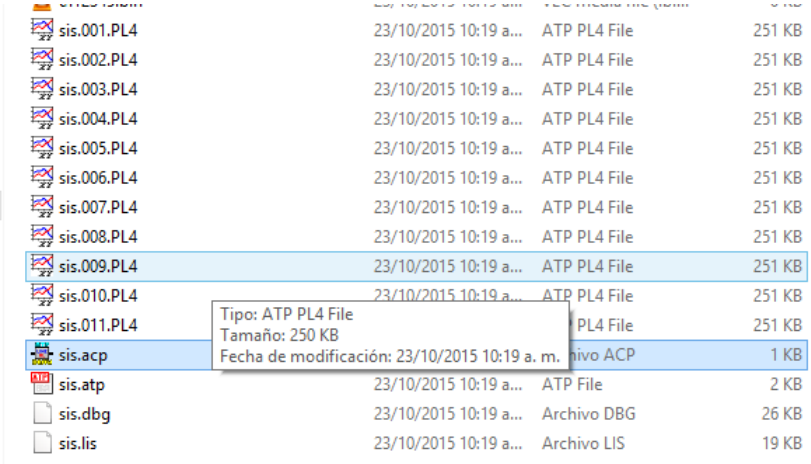


Figura A.3.: Menu configuración ATP



Con el archivo *.atp generado se procede a realizar la simulación mediante la opción run ATP del menú ATP de la barra de herramientas. Esta simulación genera las salidas en un archivo *.PL4 por cada valor definido en el barrido del parámetro (figura A.4).

Figura A.4.: Creación plantilla *.atp y generación de resultado *.PL4



Esta herramienta permite modificar la raíz de una cadena de caracteres, de esta manera se codifican los nombres de los nodos en un circuito. Antes de usar la de-

claración $PARAMETER para definir el símbolo de dato, se debe definir el bucle con el que se especifica el número de veces que se codificara la cadena de caracteres escribiendo el punto de inicio y el de final separados por una coma. Luego se define el símbolo de dato escribiendo la fórmula que incluye el contador KNT para especificar el número con el que se codificara la cadena de caracteres en cada paso del bucle, en la misma línea se llama la función SERIALIZE con la cadena de caracteres raíz encerrada dentro de comilla simple. El bucle se debe cerrar al final del código justo antes de las declaraciones BLANK.

```
DO KNT= inicio, fin /OUTPUT
$PARAMETER
Símbolo de dato = KNT SERIALIZE 'cadena de caracteres raíz'
BLANK $PARAMETER
Plantilla a ser serializada
ENDDO KNT BLANK BRANCH
```

A.2 BOKEH

Bokeh es una herramienta práctica que ayuda a crear de manera rápida y fácil gráficos interactivos, cuadros de mando y aplicaciones de datos mediante la creación de un documento bokeh procesado que se ejecuta en un navegador. Bokeh es también una librería de visualización interactiva de Python y ofrece una construcción elegante y concisa de gráficos con una interactividad de alto rendimiento sobre bases de datos muy grandes o en transmisión, de una manera rápida y sencilla. Para esto expone tres niveles de interfaz a los usuarios para ofrecerles tanto simplicidad como funciones potentes y flexibles para realizar personalizaciones, estas son:

Una interfaz bokeh.models de bajo nivel que proporciona la mayor flexibilidad a los desarrolladores de aplicaciones. Una interfaz de nivel intermedio bokeh.plotting centrada en torno a la composición de glifos visuales. Una interfaz bokeh.charts de alto nivel para crear gráficos estadísticos complejos de forma rápida y sencilla.

La interfaz bokeh.plotting es la usada en el desarrollo de la herramienta de este proyecto, también es muy útil si necesitamos personalizar la salida un poco más añadiendo más series de datos, glifos, eje logarítmico, etc. También es posible combinar fácilmente varios glifos juntos en un gráfico. En bokeh se manejan varios conceptos

---

Los gráficos son un concepto central en Bokeh. Son recipientes que contienen todos los objetos (renderizadores, guías, datos y herramientas) que comprenden la visualización final que se presenta a los usuarios. La interfaz bokeh.plotting proporciona una clase figura para ayudar con el montaje de todos los objetos necesarios, y una función de conveniencia figure() para crear objetos figura.

Las gráficas de Bokeh también pueden tener otros componentes visuales que asisten la presentación o ayudan al usuario a hacer comparaciones. Estos se dividen en dos categorías: -Las guías son asistencias visuales que ayudan a los usuarios a juzgar distancias, ángulos, etc. Estas incluyen líneas de cuadrícula o bandas, ejes (como lineal, de registro o de fecha) que también pueden tener etiquetas de marcas y marcas. -Las anotaciones son ayudas visuales que etiquetan o nombran partes de la trama. Estos incluyen títulos, leyendas, etc.

Para generar gráficos, la biblioteca del cliente BokehJS JavaScript y código CSS deben cargarse en el navegador. De forma predeterminada, la función output_file () cargará BokehJS desde http://cdn.pydata.org. Sin embargo, también puede configurar Bokeh para generar archivos HTML estáticos con recursos BokehJS incrustados directamente en el interior, pasando el argumento mode = "inline" a la función output_file ().

A.3 GTPPLOT

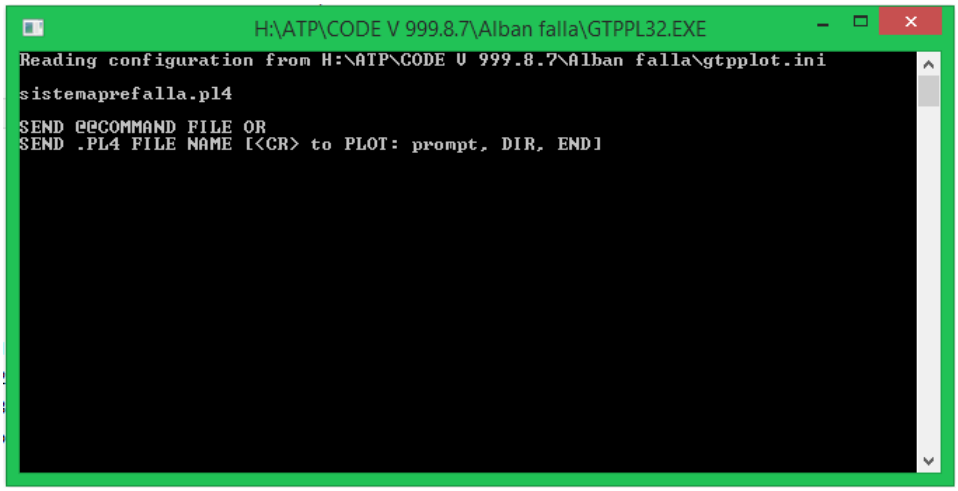
GTPPLOT es un programa de trazado para la salida de ATP, compilado con el GNU FORTRAN, y hace uso del paquete gráfico DISLIN. El programa está disponible

para DOS-djgpp extender, Windows 32 y Linux. Tiene las siguientes capacidades y limitaciones.

La versión actual de GTPPLOT puede leer archivos gráficos ampliados (FMTPL4 = 10Fnn.) Tanto con archivos binarios arbitrarios como en C, incluyendo el nuevo formato obtenido con NEWPL4 = 2 (llamado PISA) y FORTRAN (DOS-djgpp, Mingw32 , Linux y SALFORD) archivos no formateados. La extensión del archivo gráfico debe ser .PL4 por conveniencia, pero el usuario puede elegir otras extensiones. Si la extensión es .cfg, el programa considerará el archivo como un archivo COMTRADE y leerá el par .cfg y .dat, convirtiéndolos en un archivo .pl4. El programa detecta automáticamente el formato del archivo. Si la extensión es .pss, el programa considerará el archivo como un archivo PSSPLT ASCII. PSSPLT es el programa de trazabilidad de la estabilidad del paquete PSS / E 26 (Power Technologies Inc.)

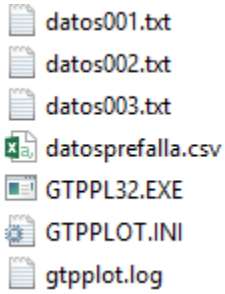
Para la conversión del archivo *.PL4 se ejecuta la aplicación GTPPL32.exe (Fig. A.5), teniendo los archivos necesarios para esto que se encuentran en la carpeta de la Herramienta desarrollada, para la ejecución son tres archivos necesarios los cuales se ilustran en la Fig A.6 que son el ejecutable GTPPL32.exe, gtpplot.log y el archivo GTPPLOT.INIque contiene la configuración establecida para la conversión en este proyecto. El programa se inicia en una ventana del CMD de windows a esperar los comandos para el proceso de conversión. El primer paso es darle el nombre del archivo *.PL4 para que el programa cargue los datos contenidos en él.

Figura A.5.: Ejecución de GTPPL32.exe en CMD de windows



Una vez cargado el archivo, por medio del comando RELAY ALL el programa realiza una conversión del archivo donde pedirá el nombre a dar al archivo permitiendo ahí elegir la extensión a la cual se desea dar. Para el caso de la herramienta se convierten en extension *.TXT para los datos Posfalla y *.CVS para los datos Prefalla como lo ilustra la figura A.6

Figura A.6.: Archivos resultantes del GTPPLOT



Todo este proceso se tiene sistematizado en la herramienta, siendo Python el encargado de crear el archivo de comandos de ordenes para el GTPPLOT y ejecutarlo en la aplicación para generar los *.txt y el datosprefalla.csv.

ANEXO B

MANUAL DE USUARIO.

El buen y correcto desempeño de la herramienta depende directamente que se cumplan a cabalidad cada uno de los requerimientos necesarios y configuraciones estipuladas. A continuación se especifica cada uno de los detalles a tener en cuenta para que la Herramienta de análisis de corto circuitos en sistemas de disstribución no presente inconvenientes al momento de ejecutarse.

La herramienta fue elaborada y probada en un equipo de las siguientes caracteristicas:

TABLA B.1.: Características Equipo de computo utilizado.

CARACTERISTICA	DESCRIPCIÓN
Sistema operativo	Windows 8.1 Pro
Procesador	AMD Quad core A10 7400P up to 3.4Ghz
Video	Tarjeta Radeon R6 dual Graphics
Memoria RAM	8 GB
Disco duro	1 TB

B.1 REQUERIMIENTOS DEL SISTEMA

Es de gran importancia tener todos los elementos previamente instalados para el buen funcionamiento de la herramienta. Estos elementos son las plataformas software necesarias para que corran los programas en los cuales está diseñada esta herramienta

software, como así mismo las librerias y demás caracteristicas que requieren.

- Instalar ATPDraw en la versión 6.0, incluyendo la carpeta de archivos del GTP-PLOT. Una vez instalado entrar a la ruta de instalación del ATP y ubicar la carpeta llamada tools (C:\ATP\tools), ahí encontrar el archivo ATPLnch.ini, abrirlo mediante un editor de texto plano, buscar donde dice "CloseWindow=0" y cambiar el valor de 0(cero) por 1(uno).
- Instalar Python en la versión 3.5 (selecionar al momento de instalar "Add Python 3.5 PATH").
- Instalar Anaconda3 versión 4.1.1, una vez instalado es necesario agregar las librerias de Bokeh a Python, para ellos se abre una consola CMD de windows y se ejecuta los siguientes comandos:

  - conda install bokeh
  - pip install bokeh=0.12.0
  - pip install –user numpy (éste solo en el caso que no reconozca la libreria ya que Anaconda la instala por defecto).
  - Configurar la ejecución de los archivos *.py para que sean lanzados por la "console Python for windows" y no por un programa editor.

Como primera media se definirá las caracteristicas que deben tener estos dos archivos en formato *.atp para le ejecucion de la herramienta como lo son el sistema de distribución objeto del análisis y el archivo del elemento modular el cual será insertado.

Es importante resaltar que los dos arhivos deben estan contenidos en una misma carpeta dentro del disco duro del equipo de computo utilizado, no es posible ejecutar la herramineta teniendo los archivos en un dispositivo de almacenamiento externo.

El sistema de distribución debe estar descrito en una plantilla de extensión *.atp, sin errores de ejecución y siguiendo las condiciones a coninuación presentadas:

- El nombre de este archivo debe ser "sistema.atp", en letras minúsculas y sin ningún otro caracter adicional.
- No puede estar activado el modo de alta precisión en la configuración del ATP, se admite el metodo por defecto de "normal" para el cual la plantilla de atp maneja una sensibilidad de los valores de entrada de 6(seis) caracteres.
- El sistema debe traer puesto un medidor de corriente (probe current) entre el nodo 0(cero) y el nodo 1(uno), de modo que el usuario tendra a partir del nodo 1(uno) en adelante disponible para la enumeración de los nodos. En el nodo 0(cero) debe ir conectado la linea de alimentación (fuente trifásica) con sus respectivas pérdidas de transformación y/o transmisión, dejando la tensión en el nodo 0(cero) como la referencia de entrada al circuito.
- Si se utiliza el modelo RF para el cálculo de fallas propuesto en éste trabajo, el sistema no puede tener ningun otro tipo de medidor adicional al ya mencionado (medidor de corriente nodo 0 al 1). En caso que no se realice un cálculo de corrientes de falla, el usuario tiene la posibilidad de colocar medidores de tensión y corriente en los nodos de interes y estos se podran visualizar en las graficas finales.
- El tiempo de simulación debe ser de mínimo 4 ciclos, para el caso más común a una frecuenta de 60Hz, se tendria:
  - Tmax= 0.0666 , correspondiente a un poco mas de 4 ciclos
  - dt= 1.1E-4, con este valor se tendrán 150 puntos de medición por ciclo, el usuario tiene la posibilidad de aumentar dicho valor pero haría lenta la

herramienta por la cantidad de datos. Aún que se realizaron pruebas con hasta 300 puntos por ciclo.

El elemento a insertar debe haber sido creado en ATPDraw, convertiendolo en una plantilla de extensión *.atp y debe llamarse "modelo.atp", teniendo dos posibilidades para dicho archivo:

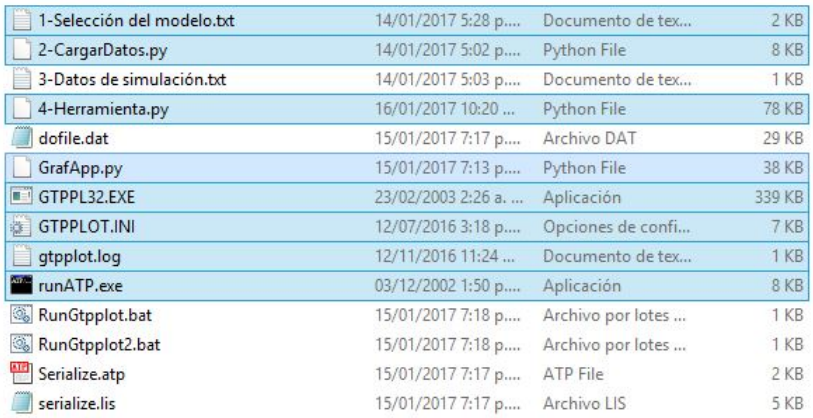
1. Usar el archivo que este trabajo provee al usuario de un modelo RF para el cálculo de corrientes de falla. Para eso se debe modificar los valores de las resistencias de falla y los tiempos de apertura/cierre directamente en ATPDraw mediante el archivo *ModeloRF.acp* acorde al tipo de falla a calcular, si el usuario asignó una variable ($PARAMETER) a un elemento del modelo RF, siempre debe verificar en ATPDraw pestaña ATP, luego seleciona "settings", donde se abre un recuadro y da click en la ultima pestaña llamada "variables", ahí encontrará un listado con el nombre asignado a un elemento seguido de la ecuación que describe el comportamiento de esa variable por medio del KNT(ver anexo A sección PCVP), donde se debe eliminar todos los nombres de las variables que no se fuesen a utilizar. Por ejemplo el modelo RF trae definidos las variables RA1, RB1, RC1 y RT1, en caso de que el usuario ya no utilice una de estas variables y le asigne un valor numérico, debera realizar el procedimiento anteior para eliminar dicha variables del listado, de lo contrario no simulará ATP.

2. Crear mediante ATPDraw un nuevo elemento para ser insertado, que de igual manera asignando variables/parametros se podran hacer cambios en sus valores. Para realizar la simulación se debe estipular la Falla=0 (cero).

B.2 FUNCIONAMIENTO DE LA HERRAMIENTA.

Como primera medida se debe verificar que todos los archivos necesarios para el funcionamiento de la herramienta se encuentren en la carpeta de ésta. Los elementos necesarios serán los que se ilustran seleccionados en azul en la figura B.1, por otra parte el resto de archivos que figuran en la imagen son creados durante la ejecución.

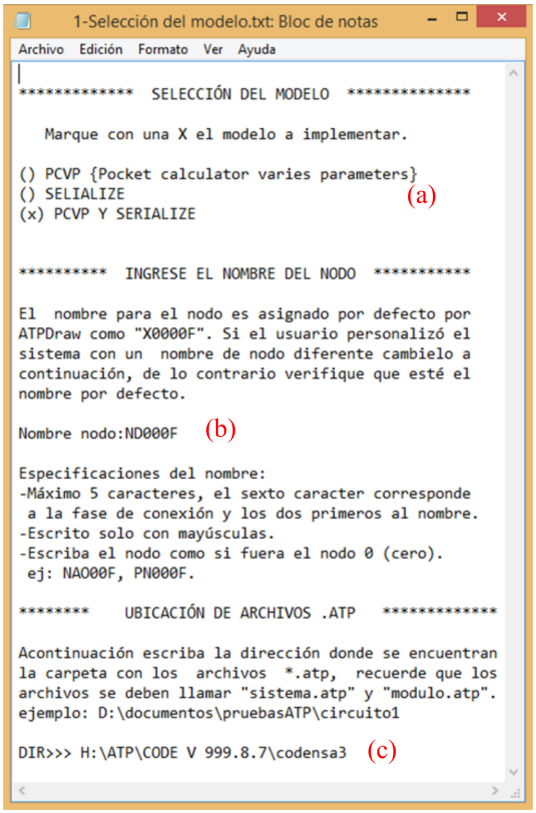
Figura B.1.: Archivos contenidos en la carpeta de la Herramienta desarrollada



La secuencia de ejecución de la herramienta son 4 pasos enumerados en los primeros archivos, que corresponden a dos archivo de texto plano (*.txt) y dos archivo ejecutables (*.py).

En el primer archivo el usuario deberá ingresar mediante un archivo de texto los siguientes datos:

Figura B.2.: Archivo Selección del modelo; (a) Selección tipo de sistematización, (b) Ingreso del Nombre del Nodo, (c) Ingreso de la ruta contenedora archivos *.atp



Como se ilustra en la Figura B.2 los datos a ser ingresados serán:

- (a): Se selecciona una sola opción de las tres disponibles para implementar las funciones de ATP en la herramienta, se debe digitar una X (equis) pudiendo ser mayúscula o minúscula en solo una de las opciones, la herramienta permite usar en cada una de sus opciones:0.03

  - PCVP: Opción por la cual permite variar el valor de un parámetro declarado previamente en ATPDraw, cambiándolo en N veces como simulaciones se hallan definido. La variación la realiza de manera ascendente, incrementando el valor por cada simulación.
  - SERIALIZE: Opción por la cual el programa permite variar el nodo de conexión del elemento modular insertado, este procedimiento lo realiza

mediante la creación y simulación de una plantilla *.atp la cual contiene el contenido de la plantilla modelo.atp dentro de la función serialize, esto genera un archivo dofile.dat que contiene N modelos como nodos de falla se han seleccionado, los cuales uno a uno serán implementados por la herramienta. El nombre de la variable que usa serialize esta definida en el código acorde al nombre del nodo ocupando los caracteres destinados a la numeración, se ilustra la manera correcta de llamar el nodo de conexión que usará el serialize utilizando el nombre del nodo referido al nodo cero en el siguiente cuadro B.2 :

TABLA B.2.: Asignación de la variable para serialize según nombre del nodo

Nombre del nodo	Cant números en el nombre	Asignación variable Serialize	Nombre del nodo Serialize
N00000	5	ZSLER	NZSLER
ND0000	4	ZSLR	NDZSLR
ND000A	3	ZSL	NDZSLA
NDX00A	2	ZS	NDXZSA

Nota: Se recomienda usar mínimo un nombre de nodo con 3 números y no se permite estos nombres para elementos del sistema ni el elemento modular, únicamente se usará como nombre del nodo a serializar.
El nombre del nodo ilustrado es a manera ilustrativa referido a que contiene la letra «N», de modo que este se modifica acorde al nombre del nodo del sistema ya sea por ejemplo XX000A siendo el nodo a serializar XXZSLA.

- PVCP Y SERIALIZE: Opción combinada por la cual el usuario tiene la posibilidad de realizar un estudio completo variando el valor del elemento en N simulaciones y del mismo modo cambiar el nodo de conexión.

- (b): Acá se ingresa el nombre del nodo con el cual fue creado el sistema de distribución contenido en la plantilla sistema.atp, el nombre debe ser referido

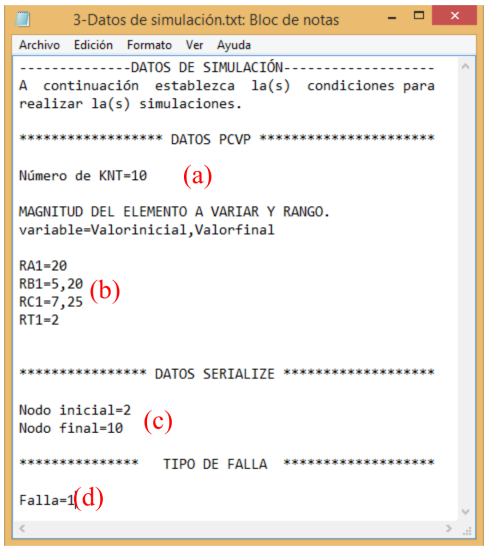
al nodo 0(cero) dejando siempre el carácter de la fase denotado con la letra F..

- (c): En esta sección de ingresa la ruta en la cual se encuentra la carpeta que contienen las dos plantillas *.atp las cuales son sistema.atp y modelo.atp.

El siguiente paso es ejecutar este código Python el cual se encargará de leer y extraer los parámetros definidos en el archivo modelo.atp y posteriormente crea el archivo "Datos de simulación" para el siguiente paso.

En este paso se ingresan los valores acorde a los parámetros extraídos del modelo.atp y se definen los datos necesarios acorde al tipo de sistematización seleccionada en el primer archivo, la figura B.3 ilustra el contenido del archivo *.txt.

Figura B.3.: Datos de simulación.txt



A continuación se explica cada uno de los valores necesarios para la Herramienta:

La parte (a) señalada en la Figura B.3 corresponde al valor designado a KNT para realizar las variaciones en el valor del parámetro, el código Python usa este valor de KNT para generar una ecuación representativa de la variación de dicho parámetro.

En (b) se ingresa el valor o rango de valores para cada una de las variables declaradas previamente en el archivo modelo.atp mediante ATDDraw, existe dos formas para la asignación de dichos valores que son:

- Dar un solo valor numérico al parámetro, de modo que la herramienta computa ese valor sobre el número de KNT. Por ejemplo el valor de RA1 es 20 para un KNT igual a 10, entonces RA1 toma 10 valores menores e igual a 20 así:

$$RA1 = [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]$$

- Dar un rango de valores para cada una de las variables, este rango debe ser escrito en dos valores numéricos separados por una coma (,). Para ejemplo se muestra los valores que usará la variable RB1 la cual fue declarada entre 5 y 20 para el KNT igual 10, donde se observa que el parámetro toma los valores límites como primera medida y el resto serán en intervalos dependiendo del valor de KNT.

$$RB1 = [5, 6{,}667, 8{,}334, 10, 11{,}667, 13{,}334, 15, 16{,}667, 18{,}334, 20]$$

Es importante que el usuario revise que estas variables sean las únicas declaradas en ATP/Settings/variables ya que se puede presentar la situación que el usuario cambie el nombre de una de estas variables y no la halla borrado en el ATPDraw, esto tendrá como consecuencia un error al ejecutar los archivos posfalla.atp.

En la parte (c) de la Figura B.3, se ingresa el nodo inicial para el análisis y el nodo final donde termina la inserción del elemento modular. La herramienta solo puede hacer el análisis entre nodos consecutivos, Serialize realiza el re-nombramiento de éstos de manera consecutiva (numéricamente) y no permite saltos en la numeración de los nodos a estudiar, de lo contrario si el usuario ingresa un rango de nodos en donde uno o mas nodos entre ese rango no fueron definidos en el sistema la Herramienta genera un error.

Para la anotación (d) de la Figura B.3 se debe asignar un valor a "Falla=", el valor por defecto que trae estipulado es 0(cero), que corresponde a no realizar dicho calculo, los valores de 1,2 y 3 corresponde a cálculos de fallas para sistemas trifásicos con conexión a tierra (Neutro) y 4 para sistemas de distribución sin neutro, cada una de las opciones serán detalladas a continuación:

- Falla=0; La herramienta realiza las simulaciones configuradas pero no realiza el calculo de corrientes de falla.

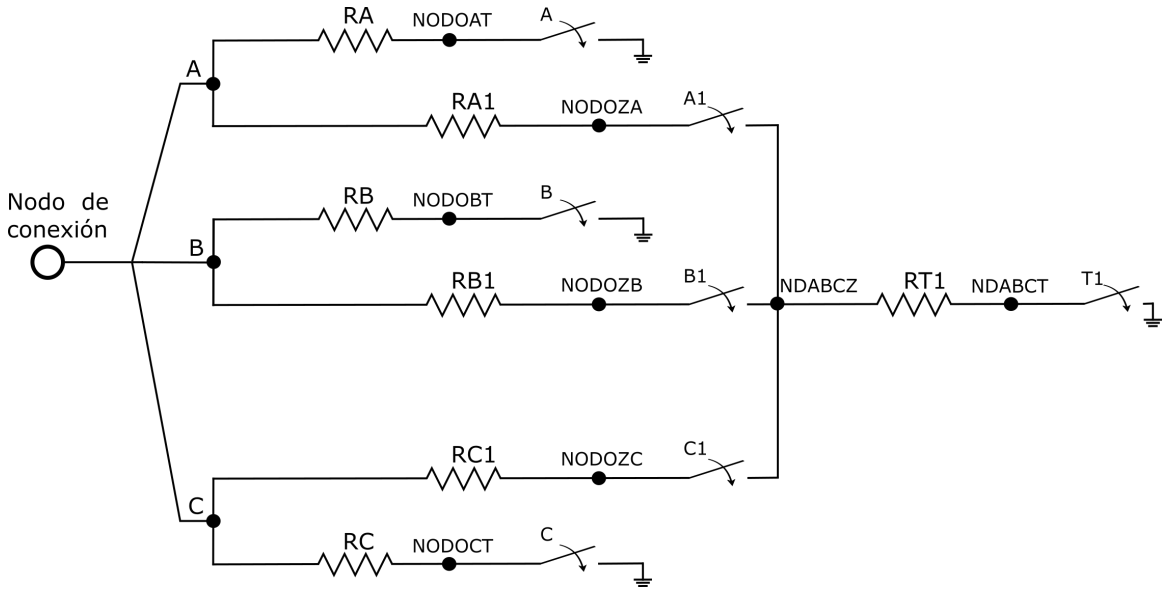
- Falla=1; Realiza cálculo de las corrientes de falla para fallas trifásicas con impedancia a tierra, bifásica con impedancia a tierra o monofásica con impedancia a tierra.

- Falla=2: Realiza el cálculo de la corriente de falla en una falla bifásica sin impedancia de tierra, para un sistema con con conexión a tierra(neutro).

- Falla=3: Realiza el cálculo para una falla monofásica a tierra.

- Falla=4; Realiza el cálculo para sistemas sin conexión tierra (neutro) , para fallas bifásicas y fallas trifásicas.

Figura B.4.: Modelo RF para cálculo de fallas.



Para cada tipo de falla se debe utilizar el modelo RF desarrollado, el cual se ilustra en la Figura B.4, cada falla tiene una configuración especifica en los interruptores[Intrp] del modelo, ya que las ramas con los elementos RA, RB y RC se utilizan para insertar la carga de prueba. Se necesitan mínimo cuatro ciclos para realizar estos cálculos con tiempo de simulación igual a 0.0665 [s], en la tabla B.3 se sugiere la configuración para cada falla.

- El cuadro tiene los tiempos para las siguientes fallas enumeradas así:

  - 1 Falla Trifásica A-B-C con impedancia a tierra [sistema aterrizado]
  - 2 Falla Bifásica A-B [sistemas aterrizados]
  - 3 Falla Monofásica en A [sistemas aterrizados]

— 4 Falla Bifásica en B-C [sistemas no aterrizados]

Para el caso de las fallas bifásicas hay que tener en cuenta que la resistencia de falla se obtiene de la suma de las resistencias de falla de cada fase, por ejemplo la falla Bifásica A-B, la resistencia de falla será RA1 + RB1.

Para las opciones 1 y 2, se debe insertar una carga de prueba por el ramal de RA, RB y RC, despues de eso se ingresa la resistencia de falla en lineas de RA1, RB1 y RC1 que se unen en la resistencia de tierra RT1. El tiempo de simulación para la carga de prueba debe ser de mínimo un ciclo y medio al igual que el tiempo para la resistencia de falla, es por esto que el tiempo mínimo para efectos de los cálculos y que dichas conexiones no se intercepten debe ser de 4 ciclos en adelante.

En el último paso se ejecuta el código python que contiene el desarrollo total de la herramienta, este es el encargado de sistematizar todo el procedimiento realizando todas las etapas que están estipuladas. Es importante que el usuario no este ejecutando aplicaciones externas al momento de iniciar la Herramienta, ya que dependiendo de la cantidad de nodos y variación del parámetro al momento de ejecutarse el código acumula procesos por periodos cortos, esto se debe a que atp maneja una velocidad alta de procesamiento en FORTRAN y windows retiene temporalmente el proceso ejecutado y al ejecutar algo externo limitaría las capacidades en memoria y procesador del equipo llegando a ocasionar la interrupción en la ejecución de la Herramienta.

Una vez finalizado el proceso de la Herramienta de simulación modular se cierran automáticamente todas las ventanas del Cmd de windows en las que se ejecuta la aplicación.

B.3 Interfaz Gráfica.

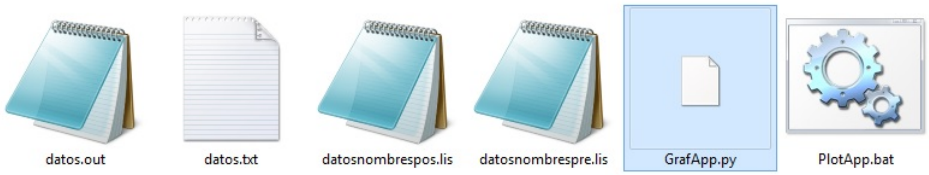
Para la visualización de los datos, hay que dirigirse a la carpeta donde se tienen los archivos de estudio (sistema.atp y modelo.atp), en esta carpeta se ha creado otra

TABLA B.3.: Tiempos sugeridos 4 ciclos con tiempo total de simulación 0.0665 [s]

Falla #	Intrpt A		Intrpt B		Intrp C		Intrp A1		Intrp B1		Intrp C1		Intrp T1	
	Time open [s]	Time close [s]	Time open [s]	Time close [s]	Time open [s]	Time close [s]	Time open [s]	Time close [s]	Time open [s]	Time close [s]	Time open [s]	Time close [s]	Time open [s]	Time close [s]
1	0.0166	-1.	0.0166	-1.	0.0166	-1.	1.	0.03	1.	0.03	1.	0.03	1.	-1.
2	0.0166	-1.	0.0166	-1.	0.0166	-1.	1.	-1.	1.	-1.	2.	1.	2.	1.
3	1.	-1.	2.	1.	2.	1.	2.	1.	2.	1.	2.	1.	2.	1.
4	2.	1.	2.	1.	2.	1.	2.	1.	1.	-1.	1.	-1.	2.	1.

carpeta llamada PlotApp en la cual esta consignada todos los archivos necesarios para las representaciones gráficas (Fig B.5).

Figura B.5.: Archivos carpeta PlotApp



El primer elemento es "datos.out" un archivo en codificación binaria realizada con Python (Pickle), que contiene toda la información obtenida de las simulaciones en ATP y adicionalmente los cálculos de falla. La información esta compactada en tres paquetes, es decir que para llamar esos datos desde Python es necesario declarar tres variables en el código. El primer paquete contiene todas la información de los valores prefalla (tensiones en los nodos de estudio y cabecera del circuito), el segundo paquete es los datos obtenidos de las simulaciones posfalla y finalmente el tercero es los cálculos de falla realizados con python, en caso de haber seleccionado Falla=0, el tercer paquete tendrá un único valor de 0(cero). Cada paquete está organizado en diccionarios de python, a manera de ejemplo se tomara como nombre de cada diccionario los nombres de Prefalla, Posfalla e Ifalla:

- El primer paquete esta organizado para llamarlo así:
  - $Prefalla[Elemento - lista - datosnombrespre.lis]$
- El segundo paquete esta de la siguiente manera:
  - $Posfalla[Nodo-de-falla][1-a-KNT][Elemento-lista-datosnombrespos.lis]$

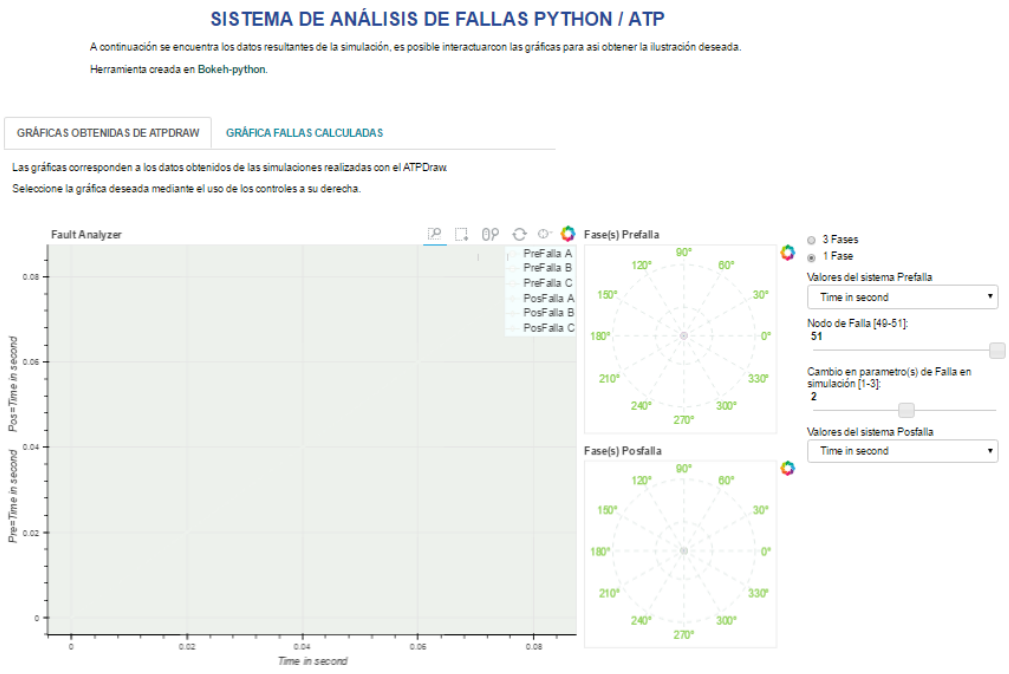
- El tercero es las corrientes calculadas:

  - $Ifalla[Nodo-de-falla][1-a-KNT][Corriente1-corriente2-corriente3]$

El archivo "datos.txt" contiene un resumen de los datos de la simulación y los archivos de extensión *.LIS, son un listado de los elementos medidos en las simulaciones. El código GrafApp.py contiene la programación con las librerías de Bokeh de la interfaz gráfica.

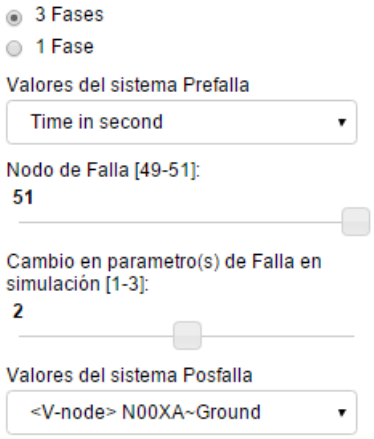
La visualización se inicia con el último archivo "PlotApp.bat", este ejecutable inicia en el CMD el bokeh serve junto al código GrafApp.py para establecer una conexión temporal con la ip http://localhost:5006/GrafApp, este enlace se abre en el explorador predeterminado que tenga el equipo de computo, por lo cual se recomienda tenerlo cerrado para el uso de la herramienta, en el desarrollo del herramienta fue probada con el Google Chrome. Una vez carga el código y se inicia la ventada en el explorador como lo indica Fig B.6.

Figura B.6.: Pantalla de visualización de resultados



La ventana tiene dos pestañas para visualizar en la primera las gráficas de Prefalla y Posfalla de la simulación, la segunda pestaña es para los cálculos de corrientes de corto. El usuario debe iniciar seleccionando una curva a representar, esto lo hace por medio de los controles de mando ubicados en la parte derecha de la ventana (Fig B.7).

Figura B.7.: Controles de selección para las gráficas.



En las opciones 'Valores del sistema Prefalla' y 'Cambio en parámetro', despliegan un listado para seleccionar la curva deseada. Como primera acción se debe seleccionar una curva en cualquiera de esas dos opciones y a continuación usando los 'deslizadores' se cambia la gráfica según el Nodo y Parámetro (KNT) ésto aplicable a las gráficas del sistema Posfalla. En la parte superior tiene la opción para visualizar las 3 fases del nodo seleccionado o solo la fase seleccionada ver Fig B.9.

Entre la gráfica de curvas y los controles de mando se encuentra dos gráficas para los Fasores prefalla y posfalla (Fig B.8) para cuando alguna de éstas sea seleccionada.

Figura B.8.: Selección visualizar Tensión nodo 51 con falla en KNT=2

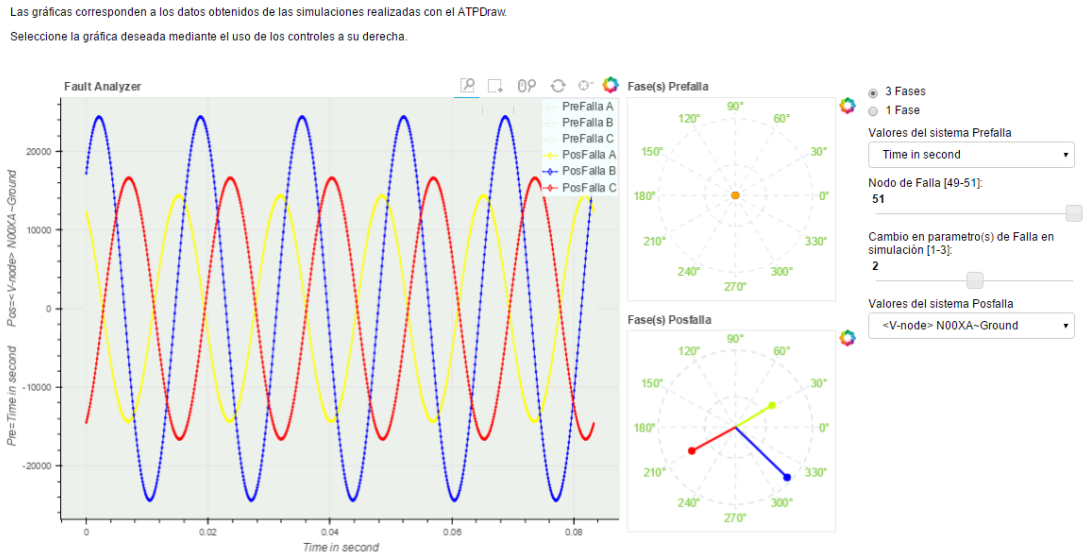
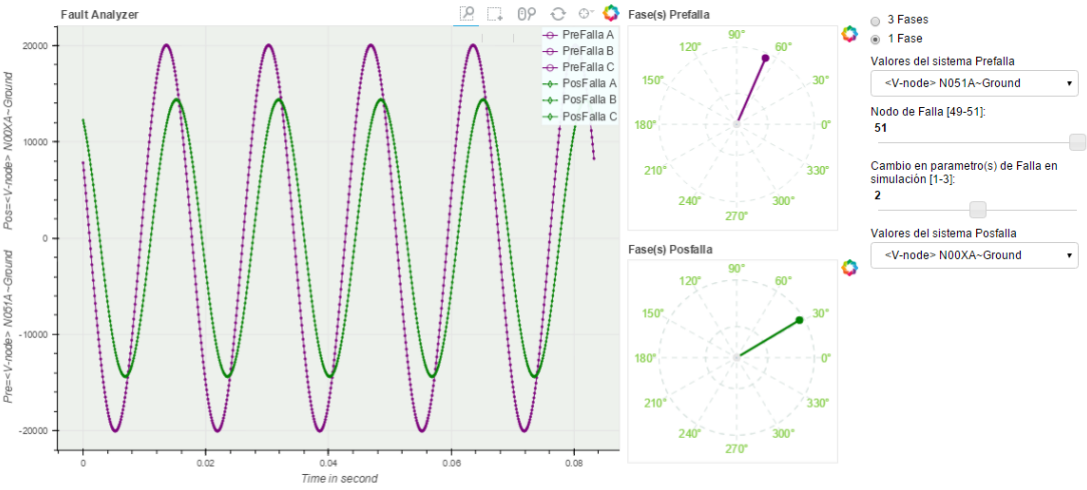


Figura B.9.: Gráfica Selección mostrar 1 Fase



La otra pestaña muestra la gráfica de las corrientes de falla calculadas. Los colores de representación de las gráficas se encuentran consignados en la tabla .

Tabla B.4.: Colores asignados para las gráficas.

Gráfica	Color [opción 3 Fases]	Color [opción 1 Fases]
fase A-PreEvento	Café	Morado
fase B-PreEvento	Negro	Morado
fase C-PreEvento	Naranja	Morado
fase A-PosEvento	Amarillo	Verde
fase B-PosEvento	Azul	Verde
fase C-PosEvento	Rojo	Verde

B.4 CAPACIDAD DE LA HERRAMIENTA Y MEDICIONES

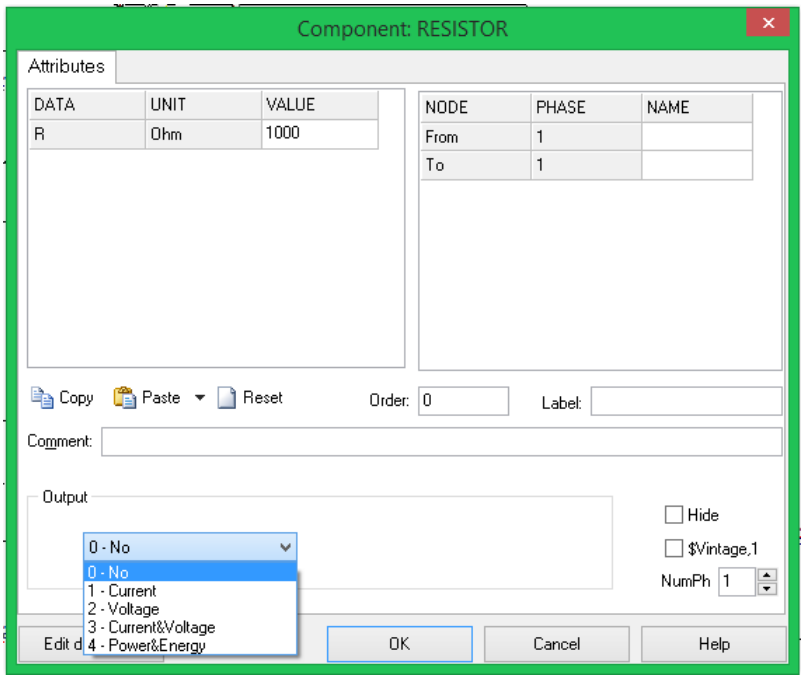
La herramienta admite plantillas creadas mediante el uso de ATPDraw para el modelo a insertar únicamente, en ella se leen por separado cada una de las secciones que contiene para definir sus elementos (Models,branch, source, switch, output y los blank para cada sección), de esa manera las inserta en su lugar respectivo.

La cantidad de nodos en los cuales es posible variar la conexión del modelo se ve estipulada por la capacidad del ATP, el cual tiene como límite poder construir circuitos de hasta 10000 elementos del Branch, 6000 Bus entre otros, es decir que la suma de los elementos de Sistema.atp y Modelo.atp no exceda estos valores. Pero estas simulaciones dependen directamente de la capacidad del equipo de computo para que soporte la velocidad con que se ejecutan éstas de manera simultanea, donde los errores mas comunes donde la herramienta falla es en la memoria que se usa mientras se esta ejecutando.

Por otro lado la cantidad de variaciones en el valor del parámetro por medio del KNT, depende de igual manera de la memoria y procesador del equipo de computo, donde se recomienda que el KNT se le asigne un valor máximo de 15 variaciones, aunque su capacidad sea mayor se hace con el fin de no generar congestión en el proceso de ejecución de la herramienta.

La herramienta permite insertar diferentes modelos al sistema, por defecto ella configura la salida de los elementos insertados pertenecientes a la sección de definición de los componentes del modelo (/BRANCH), de modo que todos los elementos definidos en ésta sección tendrán como salida la visualización de la corriente que pasa por ellos, además de la tensión en el nodo de conexión del serialize siempre y cuando el elemento (lineal o no-lineal), tenga la opción de OUTPUT en su ventana de configuración. Como se puede observar en la Figura , se configura el OUTPUT con una asignación 1 (uno), que corresponde a la medición de la corriente.

Figura B.10.: Menú de configuración del elemento ATPDraw de una Resistencia



De lo anterior se sugiere que el elemento a insertar no contenta mas de 30 componentes lineales o no-lineales (que estén en la definición del /BRANCH en la platilla *atp), ya que la información contenida será convertida en un archivo de texto plano *.txt por medio del GTPPLOT, en pruebas se comprobó que trabaja a una velocidad mayor que el Cmd de windows y generá saturación en la memoria temporal, lo que hace una interrupción en la conversión y así mismo en la ejecución de la herramienta. Cabe aclarar que este límite corresponde únicamente a la plantilla Modelo.atp, caso

contrario si el usuario agrega medidores adicionales en el archivo Sistema.atp lo podrá hacer siempre y cuando no use la opción del cálculo de corrientes de falla.