UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA

AMPLIACIÓN DE LAS CAPACIDADES DE CÁLCULO DEL ANALIZADOR SIMBÓLICO USANDO LA MATRIZ DE ADMITANCIA INDEFINDA

Br. Jodrick Xavier Colina Tromp

Mérida, agosto de 2022

UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA

AMPLIACIÓN DE LAS CAPACIDADES DE CÁLCULO DEL ANALIZADOR SIMBÓLICO USANDO LA MATRIZ DE ADMITANCIA INDEFINIDA

Trabajo de Grado presentado como requisito parcial para optar al título de Ingeniero Electricista

Br. Jodrick Xavier Colina Tromp
Tutor: Ms. Sc. Francisco J. Viloria

Mérida, agosto de 2022

UNIVERSIDAD DE LOS ANDES
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA ELÉCTRICA

AMPLIACIÓN DE LAS CAPACIDADES DE CÁLCULO DEL ANALIZADOR SIMBÓLICO USANDO LA MATRIZ DE ADMITANCIA INDEFINIDA

Br. Jodrick Xavier Colina Tromp

Trabajo de Grado, presentado en cumplimiento parcial de los requisitos exigidos para optar al título de Ingeniero Electricista, aprobado en nombre de la Universidad de Los Andes por el siguiente Jurado.

_____
Prof. Francisco J. Araujo R.
Jurado

_____
Prof. Arlex J. Cáliz M.
Jurado

_____
Prof. Francisco J. Viloria M.
Tutor

AGRADECIMIENTOS

En primera instancia, quiero agradecer al Prof. Francisco Viloria, por permitirme pese a las circunstancias del país realizar este proyecto bajo su tutoría.

A mis padres, quieres sin su apoyo incondicional no estaría acá.

A cada uno de mis familiares y amigos que me acompañaron e impulsaron durante todo este trayecto.

Br. Jodrick Xavier Colina Tromp. Ampliación de las Capacidades de Cálculo del Analizador Simbólico usando la Matriz de Admitancia Indefinida. Universidad de Los Andes. Tutor: Prof. Francisco J. Viloria. Agosto de 2022.

RESUMEN

El analizador simbólico de redes usando la matriz de admitancia indefinida (AnSiRE-MAI) es un programa de análisis de circuitos eléctricos que proporciona resultados tanto numéricos como simbólicos cuyo motor de cálculo está basado en la matriz de admitancia indefinida. El propósito de este proyecto fue ampliar las capacidades de cálculo del AnSiRE-MAI al agregar nuevos elementos (fuentes variables en el tiempo e interruptores), así como también modificar, agregar y corregir rutinas y algoritmos, para así, mejorar la efectividad del software e implementar de manera correcta los nuevos elementos agregados. A su vez, se desarrolló una interfaz de usuario con su instalador que permite la escritura, lectura y ejecución de las redes a analizar para que el usuario pueda instalar y operar de manera eficaz el software en cualquier ordenador. La programación de los mismo se hizo con los lenguajes de programación C, C++ y QML.

Descriptores: Análisis simbólico de circuitos, matriz de admitancia indefinida, transformada de Laplace, interruptores, interfaz de usuario.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El Analizador Simbólico de Redes (AnSiRE) es un programa de análisis de circuitos que un principio fue desarrollado por Omar Ruiz en el año 2017 [1]. El motor de cálculo de este software estaba basado en el método de Análisis Nodal Modificado (ANM). Posteriormente, se le hizo una actualización al software cambiando el motor de cálculo a uno más eficiente, el cual ahora hace uso de la matriz de admitancia indefinida, dando así origen al Analizador Simbólico de Redes usando la Matriz de Admitancia Indefinida (AnSiRE-MAI), esta actualización fue desarrollada por Jesús González en el año 2022 [2]. Esta última versión es capaz de calcular tensiones y corrientes en nodos y ramas, calcular funciones de transferencias y trabajar con el método de superposición con un costo computacional reducido si se compara con el software origen.

En el presente trabajo se ampliaron las capacidades de cálculo del AnSiRE-MAI así como también, se dotó de una interfaz de usuario que permite definir las redes eléctricas, realizar la ejecución y mostrar los resultados haciendo al software fácil y eficaz de utilizar.

El trabajo se divide en 4 capítulos correspondientes cada uno a lo siguiente:

En el capítulo 1, se presentan el planteamiento del problema, los objetivos del trabajo, la justificación, el alcance, la metodología y finalmente las limitaciones presentes.

El capítulo 2, se muestran los fundamentos teóricos para que se pudiese llevar a cabo el proyecto en cuestión.

El capítulo 3, contiene los algoritmos implementados, las modificaciones realizadas al software, depuración de errores encontrados y finalmente el desarrollo de la interfaz de usuario.

El capítulo 4, se realizan las pruebas de funcionamiento del software desarrollado.

CAPÍTULO 1 EXPOSICIÓN DE MOTIVOS

1.1 PLANTAMIENTO DEL PROBLEMA

El análisis simbólico de circuitos es un proceso que permite obtener una función de red de un circuito en particular, en la cual algunos o todos los elementos que los constituyen son representados por símbolos, junto con una variable independiente. Entre sus principales características, permite hallar la expresión de algún parámetro del circuito tal como la impedancia de entrada, ganancia de corriente y/o tensión, en función de los elementos del circuito representado como símbolos.

El difícil acceso a programas que hacen análisis simbólico a nivel comercial conlleva a que este tipo de análisis solo se realice de forma manual por el usuario, trayendo consigo que no se puedan validar los resultados obtenidos con un software. También, el realizar estos cálculos de forma manual implica una inversión de tiempo que puede variar de forma considerable acorde a la complejidad del o los circuitos a analizar.

Actualmente se tiene un software que permite realizar análisis simbólico llamado AnSiRE-MAI, pero está limitado solo a fuentes del tipo DC, por tales motivos, la expansión de las capacidades de análisis y cálculo es fundamental para convertir a esta en una herramienta que pueda satisfacer las necesidades de sus usuarios, siendo particularmente útil para estudiantes de ingeniería eléctrica o cualquier otra rama derivada de esta.

Con los componentes básicos iniciales cubiertas, añadir fuentes variables en el tiempo es el siguiente paso lógico en la expansión de las capacidades del programa. La inclusión de dichas fuentes expande también los requerimientos básicos del programa en cuanto a componentes básicos, además se introduce la necesidad de diseñar rutinas de análisis para circuitos con respuestas transitorias más exigentes a las implementadas hasta los momentos.

Dentro de este orden de ideas, introducir el interruptor como un nuevo elemento a la familia de componentes disponibles es natural. Este elemento es fundamental para la adición de elementos más complejos al programa y crea necesariamente rutinas aún más elaboradas debido a que los circuitos se convierten en sistemas variantes en el tiempo.

Por último, al programa en cuestión adicionarle una interfaz de usuario con su debido instalador es necesario para que así la manipulación del software sea más práctico, eficaz y que el mismo pueda correr en cualquier ordenador sin ningún inconveniente.

1.2 OBJETIVOS

1.2.1 Objetivos generales

- Ampliar las capacidades de cálculo del Analizador Simbólico de Redes -AnSiRE-MAI.

1.2.2 Objetivos específicos

- Introducir al programa elementos de fuentes de voltaje y corriente para señales variables en el tiempo, específicamente, señal seno, señal cuadrada, señal diente de sierra y señal triangular.
- Introducir al programa elementos pasivos de tipo interruptor
- Diseñar la interfaz de usuario y su instalador para el Analizador Simbólico de Redes -AnSiRE-MAI.

1.3 JUSTIFICACIÓN

El AnSiRE-MAI como herramienta de análisis simbólico de circuito, cuenta con diversos elementos y funciones para realizar así los cálculos simbólicos, si bien el software opera de manera correcta, aún faltan elementos, rutinas, así como nuevos algoritmos por agregar para así, volver al software más versátil y permitir al usuario tener una mejor experiencia al operar con el mismo. Dentro de este orden de ideas, las características más importantes necesarias por el programa son las propuestas en este trabajo de investigación, es decir: fuentes de voltaje y corriente variables en el tiempo y los interruptores. A su vez, dotar el software de un entorno para definir los circuitos, realizar la simulación y mostrar los resultados para que así los usuarios operen el mismo sin mayores inconvenientes.

1.4 ALCANCES

El alcance de este trabajo de investigación está definido bajo las siguientes premisas: La diversidad en los tipos de señales variables en el tiempo obliga a limitar las opciones presentadas al usuario, para los tipos de señales más comunes dentro del campo del diseño eléctrico, específicamente: señal sinusoidal, señal de pulsos cuadrados, señal diente de sierra y señal triangular.

El interruptor es un elemento que en aplicaciones prácticas puede ser activado por distintas condiciones (resalta notoriamente el control por voltaje), sin embargo, los interruptores implementados en este trabajo solo pueden ser controlador mediante el tiempo de activación.

Finalmente, la interfaz de usuario el cual permitirá un manejo más flexible y eficaz al introducir, guardar, abrir y/o ejecutar la simulación en cuestión al usuario.

1.5 METODOLOGÍA

El trabajo de investigación se enmarca dentro de la metodología de trabajo de un proyecto factible, la investigación dará como resultado un conjunto de algoritmos y propuestas que tienen utilidad inmediata.

1.6 LIMITACIONES

Las propuestas definidas en este trabajo de investigación están limitadas al trabajo al cual se le está haciendo las ampliaciones de capacidades de cálculo y es que los resultados se reflejarán en el dominio de la frecuencia a excepción de los circuitos que no sean declarados con elementos reactivos y fuentes variables en el tiempo.

CAPÍTULO 2 MARCO TEORICO

En el presente capítulo, se van a resaltar los aspectos teóricos relevantes que permiten llevar a cabo la ejecución del proyecto, iniciando con los antecedentes, seguido con el analizador simbólico de circuitos AnSiRE-MAI, en que se basa su motor de cálculo y las bases teóricas para poder desarrollar cada uno de los algoritmos que finalmente permiten ampliar las capacidades de este software.

2.1 ANTECEDENTES

Todo software ha de pasar por actualizaciones de codificación si se requiere agregar nuevas funciones, corrección de errores, ampliaciones en cuanto a sus capacidades, actualizaciones, fallos de seguridad, mejoramiento en velocidad, etc.

En un principio, el analizador simbólico de circuitos AnSiRE estaba basado en el método de análisis nodal modificado. Sin embargo, el costo computacional del mismo era bastante elevado si compara con el actual AnSiRE-MAI cuyo método está basado en la matriz de admitancia indefinida. El aplicar como nuevo motor de cálculo el método de matriz admitancia indefinida trajo consigo una mejora bastante notable en cuanto a su velocidad de procesamiento y muestreo de los resultados (hasta cinco veces más rápido), dato extraído de la tesis de (González Jesús, p. 146) [2]

Esto es una mejora bastante significativa pues el análisis simbólico de circuitos acarrea un costo computacional mucho mayor al cálculo numérico, por lo que el haber aplicado esta actualización al software inicial trajo consigo una optimización bastante favorable al AnSiRE-MAI.

2.2 ANALIZADOR SIMBÓLICO DE CIRCUITOS USANDO LA MATRIZ DE ADMITANCIA INDEFINIDA

Al igual que su antecesor, el actual AnSiRE-MAI está en la capacidad de resolver circuitos con elementos tales como resistencias, capacitores, inductores, transformadores, acoplamiento magnético, amplificadores operacionales, fuentes controladas, fuentes de voltaje y corriente continua.

2.3 MATRIZ DE ADMITANCIA INDEFINIDA

La matriz de admitancia indefinida se construye a partir del análisis por el método de nodos, salvo la excepción de que todos los nodos del circuito tienen un voltaje distinto de cero, es decir, no hay nodo de referencia [2]. La matriz de admitancia indefinida tiene diversas propiedades como lo son:

Primera propiedad: El orden de la matriz es igual a número de nodos que existen en el circuito eléctrico

Segunda propiedad: Esta misma es singular, por lo que su determinante es igual a cero

Tercera propiedad: En caso de que se asuma un nodo de referencia como tierra, es decir que el mismo valga cero voltios, todos los elementos de la columna correspondiente a este nodo serán igual a cero, por lo que se puede omitir de la matriz de admitancia indefinida.

Cuarta propiedad: La sumatoria de cada uno de los elementos de las casillas pertenecientes a una misma columna o fila, darán cero como resultado.

Quinta propiedad: Todos los cofactores de la matriz de admitancia indefinida proporcionaran el mismo resultado, y si llegase a pasar de que un cofactor sea igual a cero, es indicativo de que el circuito posee dos ramas independientes entre sí.

2.4 FUENTES VARIABLES EN EL TIEMPO

Para el presente proyecto, las fuentes con señales variables en el tiempo que se consideraron fueron las siguientes: fuente de señal seno, fuente de señal cuadrada, fuente de señal rampa y finalmente la fuente de señal triangular.

La secuencia de pasos para llevar a cabo la codificación de tales señales en el programa AnSiRE-MAI, contempló a las señales desde el dominio de la frecuencia, el cual, las expresiones consideradas son las siguientes:

2.4.1 Señal seno

En el dominio de la frecuencia, una señal seno con amplitud A, frecuencia ω y fase β se describe bajo la siguiente ecuación:

$$F(s) = \frac{A * s * \sin(\beta) + \mathrm{A} * \omega * \cos(\beta)}{s^2 + \omega^2} \tag{2.1}$$

Esta ecuación, es obtenida a partir de aplicar la transformada de Laplace a la función seno, con amplitud A, frecuencia ω y fase β en el dominio del tiempo, el cual tal ecuación está representada a continuación:

$$f(x) = A * \sin(\omega * x + \beta) \tag{2.2}$$

2.4.2 Señal cuadrada

Para definir la señal cuadrada en el dominio del tiempo, se necesita hacer una digresión y considerar algunos conceptos matemáticos, tales como son las funciones de singularidad. Las funciones de singularidad (también llamadas funciones de conmutación), son muy útiles en análisis de circuitos. Sirven como aproximaciones aceptables de las señales de conmutación que aparecen en circuitos con operación de conmutación. Las funciones de singularidad son discontinuas o tienen derivadas discontinuas [3].

Las tres funciones de singularidad de uso más común son las funciones de escalón unitario, impulso unitario y de rampa unitaria.

La función de escalón unitario u(t) es de 0 para valores negativos de t y de 1 para valores positivos de t.

En términos matemáticos:

$$\mu(\mathrm{t}) \begin{cases} 0, & t < 0 \\ 1, & t > 0 \end{cases} \tag{2.3}$$

cuya grafica es representa a continuación en la figura 2.1:

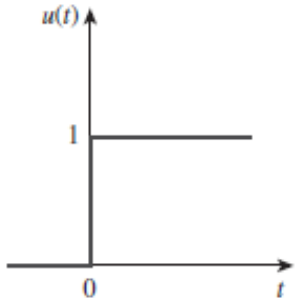

Figura 2-1. Función escalón unitario con cambio de estado en t = 0.

Si el cambio abrupto ocurre en un tiempo distinto de 0 simplemente se remplaza cada t por t = to, pudiendo ser to un valor positivo o negativo, quedando la expresión matemática de la siguiente manera:

$$\mu(\mathrm{t}) \begin{cases} 0, & t < \pm to \\ 1, & t > \pm to \end{cases} \qquad (2.4)$$

Cuyas gráficas, para t > 0 y para t < 0 se pueden observar en la figura 2.2 y 2.3:

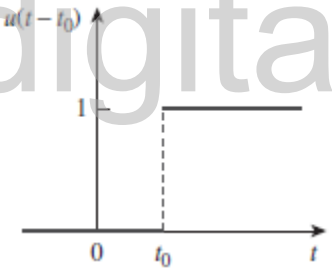

Figura 2-2. Función escalón unitario con cambio t > 0

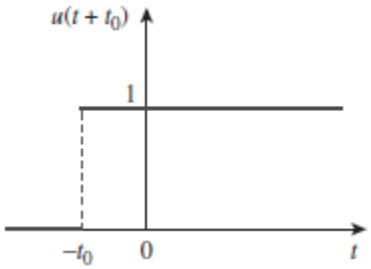

Figura 2-3. Función escalón unitario con cambio t < 0

Bajo esta premisa, se puede construir la señal cuadrada en el dominio del tiempo haciendo uso de la señal singular μ(t), quedando la misma de la siguiente manera:

$$sqr(t) = \sum_{n=1}^{\infty} A * (\mu(t) + \mu(t + nT) - 2\mu(t - \frac{nT}{2})) \quad (2.5)$$

Donde A representa la amplitud de la señal y T el periodo, el cual al aplicar la transformada de Laplace a la expresión, y por tratarse de una expresión periódica, la ecuación que se obtiene es la siguiente:

$$F(s) = \frac{A}{1 - e^{-sT}}\left(1 + e^{-sT} - 2e^{-\frac{sT}{2}}\right)\left(\frac{1}{s}\right) \quad (2.6)$$

2.4.3 Señal triangular

De igual manera que en la señal cuadrada, para definir la señal triangular en el dominio del tiempo se hace uso de la función de singularidad $\mu(t)$, técnicamente la expresión que se obtiene es similar a la ecuación sqr(t) bajo la salvedad de que la amplitud A será dividida entre el periodo T y juntos estarán en función de la variable independiente t. La ecuación que se obtiene es la siguiente:

$$tri(t) = \sum_{n=1}^{\infty} \frac{A}{T} t * (\mu(t) + \mu(t + nT) - 2\mu(t - \frac{nT}{2})) \quad (2.7)$$

El cual, al llevarlo al dominio de la frecuencia aplicando la transformada de Laplace se obtiene lo siguiente:

$$F(s) = \frac{A}{(1 - e^{-sT}) * T}\left(1 + e^{-sT} - 2e^{-\frac{sT}{2}}\right)\left(\frac{1}{s^2}\right) \quad (2.8)$$

2.4.4 Señal diente de sierra

De igual manera que la señal cuadrada y triangular, para definir la señal diente de sierra en el dominio del tiempo se utiliza la función de singularidad $\mu(t)$, obteniendo así la siguiente ecuación:

$$tri(t) = \sum_{n=1}^{\infty} \frac{A}{T} t * (\mu(t) + \mu(t + nT)) \quad (2.9)$$

El cual, al aplicar la transformada de Laplace para llevarla al dominio de la frecuencia se obtiene la siguiente expresión

$$F(s) = \frac{A}{(1 - e^{-sT}) * T}(1 + e^{-sT})\left(\frac{1}{s^2}\right) \quad (2.10)$$

2.5 INTERRUPTOR

Un interruptor es un dispositivo que se utiliza para abrir o cerrar un circuito eléctrico de forma manual o automática. En otras palabras, un interruptor eléctrico es un dispositivo de control que interrumpe el flujo de corriente eléctrica o cambia la dirección de la corriente en un circuito [4].

Se implementó dos modelos de interruptores: el interruptor de una vía y el interruptor de dos vías.

2.5.1 Interruptor de una vía (SPST)

El interruptor de una vía SPST (*single pole single throw*) también se le llama interruptor de un solo camino o sentido único. Controla una sola operación en un circuito, hace que se cierre o se abra el circuito [4].

El símbolo general para el interruptor implementado se muestra en la figura 2-4:

Figura 2-4. Modelo de interruptor de una vía SPST

2.5.2 Interruptor de dos vías (SPDT)

El interruptor de dos vías SPDT (*single pole double throw*) consta de tres pines, mediante este se pueden controlar dos circuitos al mismo tiempo. Debido a esta funcionabilidad también se les conoce a estos interruptores como interruptores selectores [4].

El símbolo general para este interruptor implementado se muestra en la figura 2.5

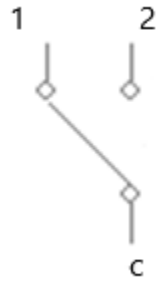

Figura 2-5. Modelo de interruptor de dos vías SPDT

La importancia de estos elementos en el análisis de circuitos eléctricos, el interruptor de una vía y el interruptor de dos vías, recae en que estos elementos producen cambios bruscos dentro de la naturaleza del circuito, y a su vez, agregando estos elementos con elementos reactivos tales como capacitores e inductores (el cual las ecuaciones que emulan la corriente y diferencia de potencial de estos elementos la rigen ecuaciones diferenciales) hace que dentro del circuito se produzca la respuesta natural y la respuesta forzada, ambas producto de aplicar un cambio brusco en la naturaleza del circuito originado por los interruptores.

CAPÍTULO 3 DISEÑO Y MODIFICACIÓN DE ALGORITMOS E INTERFAZ DE USUARIO

En el presente capítulo, se abordará los principales algoritmos implementados, las modificaciones hechas al software para la compatibilidad de los algoritmos, la interfaz gráfica de usuario elaborado, resolución de los errores más importantes encontrados en el software y finalmente los detalles de implementación.

3.1 ALGORITMO DE INTERRUPTORES

Abarca los interruptores de una vía y los interruptores de dos vías.

3.1.1 Interruptor de una vía

La implementación de los interruptores de una vía se hizo de tal manera que se pudiese trabajar de manera fluida con el resto de los algoritmos del programa, este mismo se emuló mediante una resistencia que el usuario puede especificar cuyo valor será de `rON` cuando el interruptor esté cerrado y de `rOFF` cuando este esté abierto en el circuito, los valores por defecto son 0 y $1 * 10^9$ Ω respectivamente.

Mediante este método de resolución se permite incluir el interruptor emulado como una resistencia a la matriz de transferencia del sistema.

La declaración del elemento se hace de la siguiente forma:

```
Nombre del elemento Nodo 1 Nodo 2 ON/OFF Tiempo en segundos
(rON) (rOFF)
```

En la tabla número 3.1, se observa los valores por defecto y obligatorios que se han de colocar al declarar el elemento:

Tabla 3.1. Valores por defecto al declarar el interruptor de una vía.

Parámetro	Valor por defecto	Definición/observaciones
Nombre del elemento	obligatorio	Todo interruptor de una vía debe iniciar con la letra clave "P" seguido de caracteres para declarar e identificar al elemento.
Nodo 1	obligatorio	Nodo donde se ubicará el terminal positivo del interruptor de una vía.
Nodo 2	obligatorio	Nodo donde se ubicará el terminal negativo del interruptor de una vía.
ON / OFF	obligatorio	Caracteres claves que indican el estado inicial del interruptor. ON si el interruptor está abierto. OFF si el interruptor está cerrado.
Tiempo en segundos	obligatorio	Tiempo de cambio de estado del interruptor en segundos (puede ser también simbólico).
rON	opcional	La declaración de este valor es opcional, si no se coloca ningún, el valor que tendrá la resistencia que emula al interruptor cuando está cerrado (ON) será de 0, caso contrario, si el usuario coloca un valor la resistencia en ON será dicho valor (puede ser también simbólico)
rOFF	opcional	La declaración de este valor es opcional, si no se coloca ningún valor, el valor que tendrá la resistencia que emula al interruptor cuando está abierto (OFF) será de $1 * 10^9$, caso contrario, si el usuario coloca un valor la resistencia en OFF será dicha valor (puede ser también simbólico).

Para dejar claro el cómo se declara el elemento, se parte del siguiente ejemplo:

```
P1 a b ON 5
```

Como se puede observar, el nombre del elemento es `P1`, se recuerda que cada uno de los elementos del circuito es representado por una letra y que con esta deberá comenzar el nombre del dispositivo a crear, para el caso del interruptor la letra con la que se inicia es P, a continuación le acompaña el carácter 1 que este mismo viene siendo el carácter que define el nombre del elemento, seguido de ello se encuentra el carácter `a` y luego el `b` que representa que el elemento se ubicará entre los nodos a y b, posteriormente se visualiza la palabra clave `ON`, este indica que el interruptor está cerrado, y por último, se encuentra el tiempo en

segundos, que para el ejemplo se tiene un valor de 5 segundos, pasado el tiempo de 5 segundos el interruptor cambiará de estado, es decir que pasará de `ON` a `OFF` (de cerrado a abierto). Como se puede observar, al elemento no se le colocó el valor de las resistencias rON y rOFF por lo que para efectos de este ejemplo el valor de la resistencia cuando el interruptor este cerrado será de 0 y cuando pase a abierto será de $1 * 10^9$. Si se quiere que la resistencia del interruptor cuando esté cerrado sea de 100 ohm y que cuando esté abierto sea de 1.000.000 ohm, por ejemplo, se declara al elemento agregando tales valores, tal como se ve a continuación:

```
P1 a b ON 5 100 1000000
```

También, el tiempo puede ser simbólico, así como las resistencias cuando el interruptor este abierto o cerrado, un ejemplo sería:

```
P1 a b ON Tsim Ron Roff
```

Para el correcto funcionamiento de los interruptores se tuvo que agregar nuevos algoritmos y rutinas que permiten el óptimo desempeño del mismo, en primera instancia, se tuvo que agregar una nueva lista enlazada representada a continuación:


```
struct interruptor {
    char nombre[15];
    char ro[25];
    char rf[25];
    char estado[5];
    double tiempo;
    struct interruptor *next;
};
```

Esta estructura de tipo registro con el que se hará una lista enlazada tiene como finalidad almacenar el nombre de los interruptores, el valor de la resistencia que tendrá el interruptor cuando esté cerrado (en caso de que el usuario lo haya declarado), el valor de la resistencia en caso de que el interruptor esté abierto (en caso de que el usuario lo haya declarado), el estado en el que se encuentra en un principio (ON u OFF) y finalmente el tiempo en el que el mismo realiza el cambio de abierto a cerrado o viceversa para luego ubicar cada uno de ellos en la lista enlazada llamada “elementos”, el cual esta última contiene los elementos resistivos y reactivos que se utilizan al momento de que el software hace los cálculos, esta

lista enlazada llamada “elementos” ya es parte del programa y toda la información referente a ella se encuentra en la tesis al que se está ampliando sus capacidades.

Teniendo estos parámetros almacenados en la lista enlazada llamada “interruptor”, se puede ubicar y saber las condiciones del elemento para hacer las respectivas modificaciones en el mismo, esto debido a que el valor de la resistencia que emula al interruptor cuando cambia de estado se hace durante la ejecución, básicamente se ha de acceder a la estructura elementos y realizar el cambio de estado del interruptor allí, aprovechando la disponibilidad de que el programa ya cuenta con una función para ubicar elementos dentro de la estructura únicamente pasando el nombre del elemento y como ya con la estructura llamada interruptor se tiene almacenada todos los nombres de los interruptores, se ubica al elemento, se cambia su estado y se prosigue con los cálculos que realiza el software. La función que ubica al elemento es llamada `busca_elementos` el cual su prototipo de función es:

```
Struct REGISTRO_ELEMENTO *busca_elemento (struct
REGISTRO_ELEMENTO *listaelemento, char *palabra)
```

busca_elemento devuelve la dirección de memoria de la estructura (nodo) donde está ubicado el elemento buscado, de existir, de lo contrario retorna NULL.

- **Parámetros:**

Listaelemento puntero que apunta al primer nodo de la lista enlazada donde se crearon y guardaron los elementos.

palabra nombre del elemento que se buscará en cada uno de los nodos de la lista enlazada.

Una vez ubicado el elemento, se procede a hacer el cambio de estado del interruptor.

Otro aspecto importante a tomar en consideración es que se creó una función cuya utilidad es ordenar la lista enlazada interruptor acorde al tiempo en que los que los interruptores cambian de estado, se recuerda que en el programa se pueden declarar varios interruptores a excepción si el tiempo de cambio de estado es simbólico, el cual esto último se explicará más adelante, sin embargo, en un principio el usuario puede introducir en la interfaz los interruptores de manera aleatoria (se hace referencia al tiempo en que cambia de estado), y la lista enlazada guarda los datos acorde a la posición en que los encontró en la interfaz, que

no necesariamente pueden estar ordenados, por ejemplo, el usuario pudo haber ingresado dos interruptores llamado p1 y p2, siendo el cambio de p1 en 10 segundos y el de p2 en 5 segundos, bajo esta lógica el interruptor p2 cambia de estado primero que el interruptor p1, pero si la lista enlazada no ordenara la posición de los interruptores acorde al tiempo de cambio de estado, como el primer elemento declarado fue p1, cambiaría de estado primero p1 y luego 5 segundos después cambiaria de estado p2, lo cual sería incorrecto, por lo que para evitar que el usuario tenga que introducir de manera ordenada los interruptores acorde al tiempo en que el interruptor cambia de estado se creó la función que ordena la posición de estos elementos con base a dicho tiempo, lo que hace al programa más eficiente y cómodo al momento de trabajar con estos elementos.

El prototipo de función que ordena acorde al tiempo de menor a mayor el cambio de los estados de los interruptores es:

```
void insertar_ordenar(char *nombre, char *tiempo, char *ro,
char *rf, char *estado)
```


El desarrollo de la función es:

```
void insertar_ordenar(char *nombre, char *tiempo, char *ro,
char *rf, char *estado)
{
    double x;
    int centinela = 0;
    struct interruptor *nuevo;
    nuevo=malloc(sizeof(struct interruptor));

    if(isalpha(tiempo[0]) == 0){
        x = atof(tiempo);
        nuevo->tiempo = x;
    }
    else{
        strcpy(nuevo->tiempo_sim,tiempo);
        centinela = 1;
    }

    strcpy(nuevo->nombre,nombre);
    strcpy(nuevo->ro,ro);
    strcpy(nuevo->rf,rf);
    strcpy(nuevo->estado,estado);
    nuevo->next=NULL;
```

```
    if (raiz == NULL)
        raiz = nuevo;

    else if(centinela == 1){
        //salir de la funcion si es simbolico
        struct interruptor *reco = raiz;
        reco->next = nuevo;
    }

    else{
        if (x<raiz->tiempo){
            nuevo->next = raiz;
            raiz = nuevo;
        }
        else{
            struct interruptor *reco = raiz;
            struct interruptor *atras = raiz;
            while (x >= reco->tiempo && reco->next != NULL){
                atras = reco;
                reco = reco->next;
            }
            if (x >= reco->tiempo)
                reco->next = nuevo;
            else
            {
                nuevo->next = reco;
                atras->next = nuevo;
            }
        }
    }
}
```


Siendo raíz un apuntador de tipo struct **interruptor** que en un principio apunta a NULL, la utilidad de este es tener referenciado la posición del primer elemento de la lista enlazada struct **interruptor** para su futura utilización.

Insertar_ordenar no retorna ningún valor ya que la finalidad de este mismo es ordenar la lista enlazada acorde al tiempo de cambio de estado del interruptor.

- **Parámetros:**

****nombre*** es el nombre que se le dio al interruptor.

****tiempo*** es el tiempo en el que el interruptor cambia de estado.

ro es el valor de la resistencia en ON (en caso de existir)

rf es el valor de la resistencia en OFF (en caso de existir)

estado contiene el estado inicial del interruptor (puede ser ON u OFF)

Para ordenar las estructuras que conforma la lista enlazada struct **interruptor** acorde al tiempo de cambio de estado de los interruptores se debió transformar la cadena de caracteres "tiempo" a punto flotante, para que así, se puede realizar la comparación y ordenar de menor a mayor, esto se hizo con la función atof propio de la librería stdlib.h

Los interruptores también pueden ser declarados con su tiempo de manera simbólica, bajo la excepción de que solamente se puede declarar un elemento de este tipo, si se declara más de un interruptor habiendo un interruptor con tiempo simbólico el programa arrojará un mensaje indicando que no es posible proseguir con la ejecución y finalizará el mismo. Un ejemplo de declaración de un interruptor de una vía simbólico es el siguiente:

```
P1 a b ON T_SIM
```


3.1.2 Interruptor de dos vías

A diferencia de los interruptores de una vía cuya utilidad es desprender o conectar una rama de un circuito, los interruptores de dos vías lo que hacen es desprender una rama de un circuito para conectar el elemento en otra rama del circuito y viceversa. la lógica tras este elemento es similar a la de los interruptores de una vía cuya única diferencia es que se crearan dos elementos resistivos, cuyo valor de la primera resistencia es 0 (al menos que el usuario indique lo contrario) y la de la segunda resistencia tendrá $1 * 10^9$ (al menos que el usuario indique lo contrario), pasado el tiempo en que el interruptor de dos vías cambia de estado, la primera resistencia tomará el valor de la resistencia que emula el camino abierto y la segunda tomará el valor de la resistencia que emula el camino cerrado, lo que se pudiese interpretar como dos interruptores de una vía que cambian de estado en el mismo tiempo pero con un nodo en común.

La declaración del elemento se hace de la siguiente forma:

```
Nombre del elemento   Nodo 1   Nodo común   Nodo 2   Tiempo en
segundos (rON) (rOFF)
```

En la tabla número 3.2, se observa los valores por defecto y obligatorios que se han de colocar al declarar el elemento

Tabla 3.2. Valores por defecto al declarar el interruptor de dos vías vía.

Parámetro	Valor por defecto	Definición/observaciones
Nombre del elemento	obligatorio	Todo interruptor de dos vías debe iniciar con la letra clave "S" seguido de caracteres para declarar e identificar al elemento.
Nodo 1	obligatorio	Nodo donde se ubicará el elemento. Siempre el nodo 1 y el nodo común emularan el camino en ON hasta que el interruptor cambie de estado pasado el tiempo, el cual pasará a OFF.
Nodo común	obligatorio	Nodo común que tendrá el elemento al cambiar de estado.
Nodo 2	obligatorio	Nodo donde se ubicará el elemento. Siempre el nodo 2 y el nodo común emularan el camino en OFF hasta que el interruptor cambie de estado pasado el tiempo, el cual pasará a ON
Tiempo en segundos	obligatorio	Tiempo de cambio de estado del interruptor en segundos (puede ser también simbólico)
rON	opcional	La declaración de este valor es opcional, si no se coloca ningún valor, el valor que tendrá la resistencia que emula al interruptor cuando está cerrado será de 0, caso contrario, si el usuario coloca un valor la resistencia en ON será dicho valor (puede ser también simbólico).
rOFF	opcional	La declaración de este valor es opcional, si no se coloca ningún valor, el valor que tendrá la resistencia que emula al interruptor cuando está abierto será de $1 * 10^9$, caso contrario, si el usuario coloca un valor la resistencia en OFF será dicha valor (puede ser también simbólico).

Para dejar claro el cómo declarar el elemento, se parte de la siguiente declaración:

```
S1 a b c 5
```

El nombre del elemento es `S1`, el tiempo de cambio de estado es en 5 segundos, en un principio se tiene una resistencia de valor 0 en el nodo `a` y `b` y una resistencia de valor $1 * 10^9$ en el nodo `b` y `c`, siendo `b` el nodo común, pasado los 5 segundos la resistencia entre los

nodos `a b` cambiará a $1 * 10^9$ y la resistencia entre los nodos `b c` cambiará a 0, ambos cambios ocurren simultáneamente, si se quiere que en el nodo b c en un principio el valor de la resistencia sea 0 y no $1 * 10^9$, es decir que emule el camino cerrado, y pasado el tiempo de cambio que para este ejemplo es de 5 segundos, pase a valer $1 * 10^9$, la declaración del elemento tendría que ser `S1 c b a 5`.

Al igual que los interruptores de una vía, este elemento también se puede declarar agregando los valores que tendrán las resistencias cuando el elemento esté conectado a una rama y desprendida de la otra, por ejemplo, si se parte del ejemplo anterior `S1 a b c 5`, si se quiere que la resistencia que hay entre los nodos `a b` sea de 100 y entre los nodos `b c` sea de 1.000.000, la declaración del elemento se haría de la siguiente forma:

```
S1 a b c 5 100 100000
```

Pasado el tiempo de 5 segundos, la resistencia contenida en el nodo a b pasará valer ahora 1.000.000 y la resistencia del nodo b c pasará a valer 100

Como se mencionó anteriormente, el funcionamiento del mismo es similar al de los interruptores de una vía, por consiguiente las funciones, rutinas y lógicas que se utilizan para emular al elemento son las mismas, salvo ciertas pequeñas excepciones al momento de almacenar el mismo en la lista enlazada struct **interruptor** y las de cambiar el valor resistivo, en vez de recorrer una sola posición de la lista enlazada para realizar el cambio del valor resistivo se tiene que recorrer dos posiciones consecutivas pues se tiene que cambiar el valor de dos resistencias que son las que emulan este elemento, de resto, todo es idéntico.

Al igual que los interruptores de una vía, el interruptor de dos vías puede ser declarado colocando su tiempo de cambio de estado de manera simbólica considerando las mismas limitaciones explicadas anteriormente, un ejemplo de la declaración sería:

```
S1 a b c T_SIM
```

También, la resistencia del camino cerrado y del camino abierto se puede declarar de manera simbólica, un ejemplo de la declaración seria:

```
S1 a b c T_SIM Ron Roff
```

3.2 FUENTES VARIABLES EN EL TIEMPO

Incluye las fuentes con señal sinusoidal, cuadrada, triangular y diente de sierra. La elaboración de las mismas se hizo de tal manera que las fuentes variables en el tiempo pudiesen agregarse a la matriz de transferencia del sistema, sin embargo, se tuvieron que hacer modificaciones a las rutinas existentes y agregar otras para su correcta implementación y funcionamiento. Dentro de las rutinas modificadas más importantes se tiene:

```
struct  REGISTRO_ELEMENTO{
    char elemento[15];
    char valor[100];
    char nodo_1[15];
    char nodo_2[15];
    char nodo_3[15];
    char nodo_4[15];
    char freq_1[15];
    char fase_1[15];
    char param8[15];

    struct MATRIZ **matriz;
    struct REGISTRO_NODO *listaNodosAsociada;
    int numeroNodosAsociados;

    char cofactor1Asociado[MAX_LONG_LINEA];
    char cofactor2Asociado[MAX_LONG_LINEA];

    struct REGISTRO_ELEMENTO *sig;
};
```


En esta estructura con la que se emula una lista enlazada en cual se registran los elementos que se van agregando a la matriz de transferencia del sistema, se agregaron las variables de tipo char "`freq_1[15]`" "`fase_1[15]`" puesto que el AnSiRE-MAI no se registraban elementos con más de 6 parámetros. Agregar estos dos parámetros conllevo a modificar cada una de las funciones y registros de tipo estructura en él que se hacían uso de estos parámetros.

Otra modificación importante fue en la función *crear_elemento cuyo prototipo de función es:

```
Struct REGISTRO_ELEMENTO *crear_elemento(char *nombre, char *nodo1, char *nodo2, char *nodo3, char *nodo4, char *valor, char *freq, char *fase, struct REGISTRO_ELEMENTO *listaelemento, FILE *archivo);
```

Dentro de esta función que retorna la dirección de memoria de un registro de tipo estructura fue donde se agregó los algoritmos para cada una de las señales variables en tiempo.

Como bien se mencionó, se hicieron múltiples modificaciones y se agregaron diversas rutinas, se buscó la manera en que tales modificaciones hechas se hicieran únicamente de ser estrictamente necesario para no alterar considerablemente la esencia del programa original. La declaración de los elementos agregados se visualiza a continuación:

3.2.1 Fuente de señal sinusoidal

La función de señal sinusoidal se implementó a partir de la ecuación (2.1).

La declaración del elemento se hace de la siguiente forma y sus valores por defectos se muestran en la tabla 3.3:

```
Nombre del elemento  Nodo 1  Nodo 2  Tipo de fuente  valor offset  amplitud  frecuencia  (fase)
```

Tabla 3.3. Valores por defecto al declarar una fuente de señal sinusoidal.

Parámetro	Valor por defecto	Definición/observaciones
Nombre del elemento	obligatorio	Debe iniciar con su letra clave “V” o “I” (dependiendo de si es una fuente de tensión independiente o fuente de corriente de independiente) seguido de caracteres para declarar e identificar al elemento
Nodo 1	obligatorio	Nodo donde se ubicará el terminal positivo de la fuente de señal seno
Nodo 2	obligatorio	Nodo donde se ubicará el terminal negativo de la fuente de señal seno
Tipo de fuente	obligatorio	Caracteres claves que indican el tipo de fuente. Para señal seno sus caracteres clave son SIN.
Valor offset	obligatorio	Valor offset agregada a la señal
Amplitud	obligatorio	Amplitud de la señal
Frecuencia	obligatorio	Frecuencia de la señal
Fase	opcional	Fase de la señal, valor por defecto 0

3.2.2 Fuente de señal cuadrada

Para la implementación de esta señal variable en el tiempo al igual que la señal triangular y diente de sierra, se tuvo que buscar otra alternativa en el que software pudiese operar con las expresiones de estas señales discontinuas puesto que las mismas no presentan una solución directa mediante el analizador de expresiones usado en el programa Mathomatic, esto se debe a que Mathomatic está severamente limitado en cuanto a evaluación de funciones en el dominio de la frecuencia se refiere [5]. Por consiguiente, se ameritó analizar dichas expresiones desde otro punto de vista.

En Primera instancia, se analizó un periodo de la señal y se llevó al dominio de la frecuencia, obteniendo así para la señal de onda cuadrada la siguiente expresión:

$$F(s) = \alpha\frac{A}{S} \tag{3.1}$$

Donde A representa la amplitud de la señal, S es la variable en el dominio de la frecuencia y α es un factor variable que depende de la ubicación del punto de evaluación dentro de la ecuación fundamental, valiendo esta (1,-1) indicando así el segmento positivo y negativo de la función onda cuadrada.

Al tratarse de una señal periódica, al resultado final se le multiplica por la siguiente expresión:

$$Periodicidad(s) = \frac{1}{1 - e^{-sT}} \tag{3.2}$$

Siendo T el periodo de la señal.

La declaración del elemento se hace de la siguiente forma y sus valores por defecto se muestran en la tabla 3.4:

```
Nombre del elemento  Nodo 1  Nodo 2  Tipo de fuente  valor
offset  amplitud  frecuencia  (fase)
```

Tabla 3.4. Valores por defecto al declarar una fuente de señal cuadrada.

Parámetro	Valor por defecto	Definición/observaciones
Nombre del elemento	obligatorio	Debe iniciar con su letra clave "V" o "I" (dependiendo de si es una fuente de tensión independiente o fuente de corriente de independiente) seguido de caracteres para declarar e identificar al elemento
Nodo 1	obligatorio	Nodo donde se ubicará el terminal positivo de la fuente de señal cuadrada
Nodo 2	obligatorio	Nodo donde se ubicará el terminal negativo de la fuente de señal cuadrada
Tipo de fuente	obligatorio	Caracteres claves que indican el tipo de fuente. Para señal cuadrada sus caracteres clave son SQR.
Valor offset	obligatorio	Valor offset agregada a la señal
Amplitud	obligatorio	Amplitud de la señal
Frecuencia	obligatorio	Frecuencia de la señal
Fase	opcional	Fase de la señal. Valor por defecto 0

3.2.3 Fuente de señal triangular

Al igual que la señal de onda cuadrada, se tuvo que abordar desde otra perspectiva el cómo implementar esta señal, la lógica empleada es idéntica a lo explicado anteriormente siendo la expresión hallada la siguiente:

$$F(s) = \alpha\frac{A}{s} + \frac{4\beta A}{Ts^2} \tag{3.3}$$

Al resultado final se le multiplica con la expresión (3.2)

La declaración del elemento se hace de la siguiente forma y sus valores por defecto se muestran en la tabla 3.5:

```
Nombre del elemento    Nodo 1    Nodo 2    Tipo de fuente
valor offset   amplitud   frecuencia   (fase)
```

Tabla 3.5. Valores por defecto al declarar una fuente de señal triangular.

Parámetro	Valor por defecto	Definición/observaciones
Nombre del elemento	obligatorio	Debe iniciar con su letra clave "V" o "I" (dependiendo de si es una fuente de tensión independiente o fuente de corriente de independiente) seguido de caracteres para declarar e identificar al elemento
Nodo 1	obligatorio	Nodo donde se ubicará el terminal positivo de la fuente de señal triangular
Nodo 2	obligatorio	Nodo donde se ubicará el terminal negativo de la fuente de señal triangular
Tipo de fuente	obligatorio	Caracteres claves que indican el tipo de fuente. Para señal seno sus caracteres clave son TRI.
Valor offset	obligatorio	Valor offset agregada a la señal
Amplitud	obligatorio	Amplitud de la señal
Frecuencia	obligatorio	Frecuencia de la señal
Fase	opcional	Fase de la señal. Valor por defecto 0

3.2.4 Fuente de señal diente de sierra

Para la señal diente de sierra la función hallada fue la siguiente:

$$F(s) = \frac{A}{T} * \frac{1}{s^2} \tag{3.4}$$

Al resultado final se le multiplica por la expresión (3.2)

La declaración del elemento se hace de la siguiente forma y sus valores por defecto se muestran en la tabla 3.6:

```
Nombre del elemento  Nodo 1  Nodo 2  Tipo de fuente  valor offset  amplitud  frecuencia  (fase)
```

Tabla 3.6. Valores por defecto al declarar una fuente de señal diente de sierra.

Parámetro	Valor por defecto	Definición/observaciones
Nombre del elemento	obligatorio	Debe iniciar con su letra clave “V” o “I” (dependiendo de si es una fuente de tensión independiente o fuente de corriente de independiente) seguido de caracteres para declarar e identificar al elemento
Nodo 1	obligatorio	Nodo donde se ubicará el terminal positivo de la fuente de señal diente de sierra
Nodo 2	obligatorio	Nodo donde se ubicará el terminal negativo de la fuente de señal diente de sierra
Tipo de fuente	obligatorio	Caracteres claves que indican el tipo de fuente. Para señal seno sus caracteres clave son SAW.
Valor offset	obligatorio	Valor offset agregada a la señal
Amplitud	obligatorio	Amplitud de la señal
Frecuencia	obligatorio	Frecuencia de la señal
Fase	opcional	Fase de la señal. Valor por defecto 0

Cada tipo de fuente puede ser declarado con sus valores obligatorios y opcionales de manera simbólica.

3.3 INTERFAZ DE USUARIO

Para la elaboración de la interfaz usuario, se utilizó el entorno de desarrollo QT, en cual se creó un proyecto en el módulo Qt Widgets. El módulo de Qt Widgets proporciona un conjunto de elementos para crear interfaces de usuario clásicos de estilo de escritorio. Los Widgets son los elementos principales para crear interfaces de usuario en Qt. Los Widgets pueden mostrar datos e información de estado, recibir información del usuario y proporcionar contenidos para otros Widgets [6].

La programación de la interfaz de usuario se hizo mediante el lenguaje de programación QML, lenguaje propio de QT, el cual el mismo integra funciones de JavaScript y C++, a su vez, se utilizó el lenguaje de programación C y C++ para complementar parte de la codificación.

La interfaz del AnSiRE-MAI y el software AnSiRE-MAI funcionan como dos programas independientes, es decir, al momento de que se escriben los datos en la interfaz y se presiona la pestaña ejecutar (la misma se explicará más adelante), el entorno llamará al ejecutable .exe que en este caso es el software AnSiRE-MAI, se realizan los respectivos cálculos en el mismo y luego se retorna nuevamente al entorno para visualizar los resultados.

El proyecto de la interfaz de usuario, cuenta con cuatro carpetas en los cuales se encuentran los archivos de cabecera, los códigos fuentes, los recursos (donde se encuentran las imágenes usadas en el interfaz) y finalmente los formularios (donde se construye la interfaz). En conjunto dan origen a la interfaz mostrada en la figura 3.1.

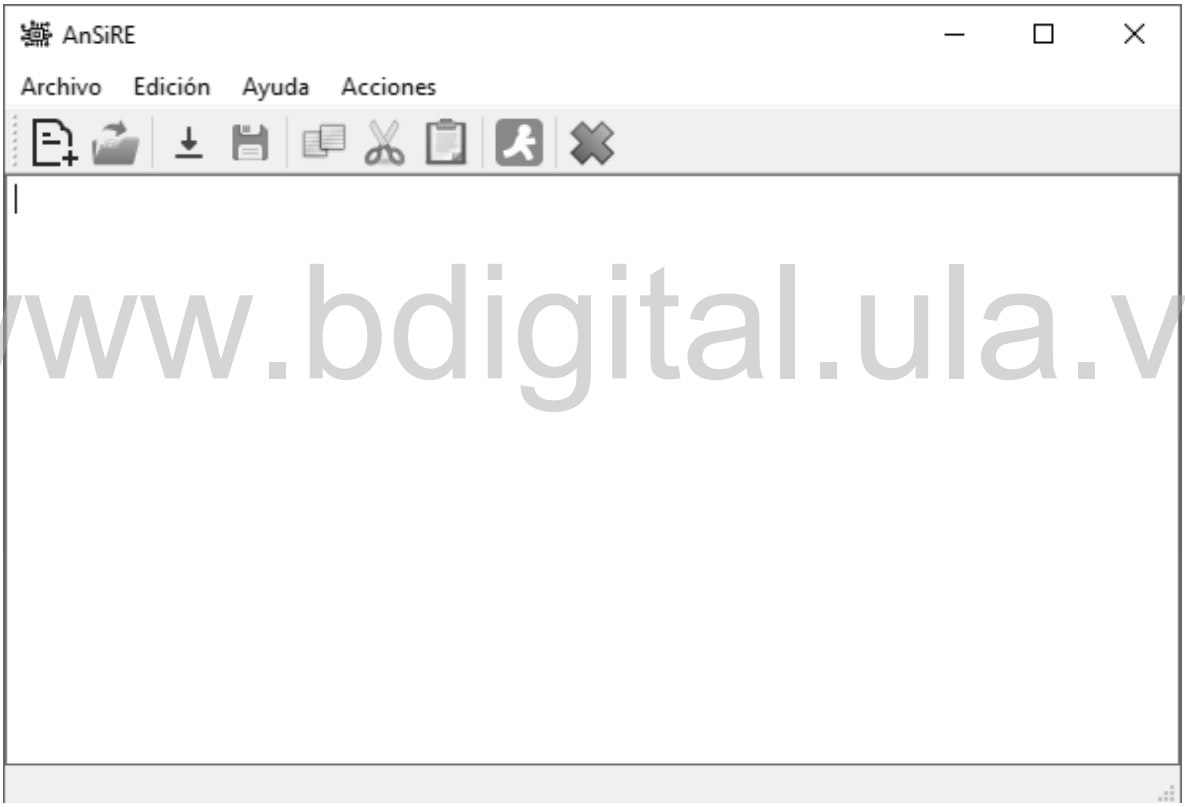

Figura 3.1. Interfaz de Usuario

En el AnSiRE al igual que en el AnSiRE-MAI, se debía crear un fichero de manera manual, colocar los datos de simulación en el fichero, copiar la ruta de ubicación del fichero y pasarlo por la línea de comando de argumentos al ejecutar el programa o en su defecto, colocar de manera manual la ruta del fichero en plena ejecución del programa. Habiendo hecho esto, el analizador realizaba los cálculos y para visualizar los datos se había que dirigir a la ruta donde

se creó el fichero de salida con los resultados, esto no es nada amigable para el usuario puesto que, cada vez que se iba a correr una simulación distinta, se debía crear el fichero de manera manual, y hacer los pasos mencionados anteriormente.

Ahora, se dispone de una interfaz en el que el usuario puede colocar los datos de simulación, ejecutar el programa para que realice la simulación sin necesidad de pasar la ruta de los ficheros de manera manual, visualizar los resultados, abrir archivos existentes para realizar modificaciones o volver a simular, guardar los resultados en una ruta deseada, un menú de ayuda de cómo funciona el programa y otras funciones que serán explicadas en breve.

3.3.1 Menú Archivo

El menú archivo se desglosa en cinco pestañas llamadas: abrir, nuevo, guardar, guardar como y cerrar, tal como se visualiza en la figura 3.2

La pestaña "*abrir*" permite abrir un documento de texto existente para su visualización, modificación y/o ejecución, también se puede acceder mediante el atajo de teclado ctrl+q. El prototipo de función codificado que se encarga de realizar esta acción es la siguiente:


```
void on_actionAbrir_triggered();
```

La pestaña "*nuevo*" permite crear un nuevo documento de texto, ya sea para limpiar el área de escritura, realizar otra simulación que requiera guardarse con otro nombre, etc. también se puede acceder mediante el atajo de teclado ctrl+n. El prototipo de función codificado que se encarga de realizar esta acción es la siguiente:

```
void on_actionNuevo_triggered();
```

La pestaña "*guardar*" permite guardar el documento en su ubicación actual, de no existir pedirá al usuario que introduzca el nombre y la ubicación para finalmente guardarlo. Su atajo de teclado es ctrl+s. El prototipo de función codificado que se encarga de realizar esta acción es la siguiente:

```
void on_actionGuardar_triggered();
```

La pestaña "*guardar como*" permite guardar el documento por primera vez, en una ubicación distinta, o para crear una copia de este en la misma ubicación o una ubicación distinta. Su

atajo de teclado es ctrl+g. El prototipo de función codificado que se encarga de realizar esta acción es la siguiente:

```
void on_actionGuardar_como_triggered();
```

La función "*cerrar*" termina el proceso y cierra el programa. Su atajo de teclado es la tecla esc. El prototipo de función codificada que se encarga de realizar esta acción es la siguiente:

```
void on_actionCerrar_triggered();
```

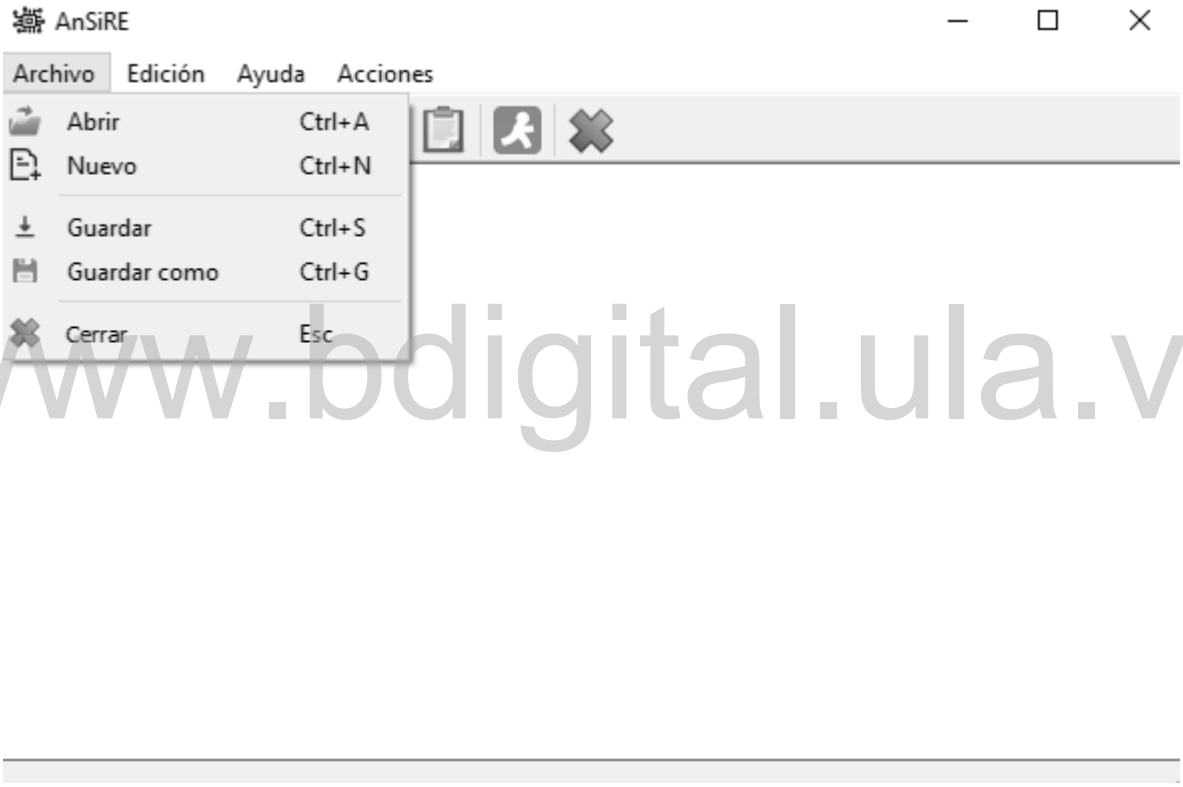

Figura 3.2. Menú Archivo

3.3.2 Menú edición

El menú edición cuenta con tres pestañas llamadas: copiar, cortar y pegar. Tal como se ve en la figura 3.3.

La función "*copiar*" permite duplicar el texto seleccionado para ser usado en otra parte mediante la función "pegar". Su atajo de teclado es ctrl+c. El prototipo de función codificado que se encarga de realizar esta acción es la siguiente:

```
void on_actionCopiar_triggered();
```

La función "*cortar*" elimina el texto seleccionado pero el mismo puede ser usado en otra parte haciendo uso de la función "pegar". Su atajo de teclado es crtl+x. El prototipo de función codificada que se encarga de realizar esta acción es la siguiente:

```
void on_actionCortar_triggered();
```

La función "pegar" permite colocar el texto copiado o cortado en el lugar seleccionado. Su atajo de teclado es *ctrl+v*. El prototipo de función codificada que se encarga de realizar esta acción es la siguiente:

```
void on_actionPegar_triggered();
```


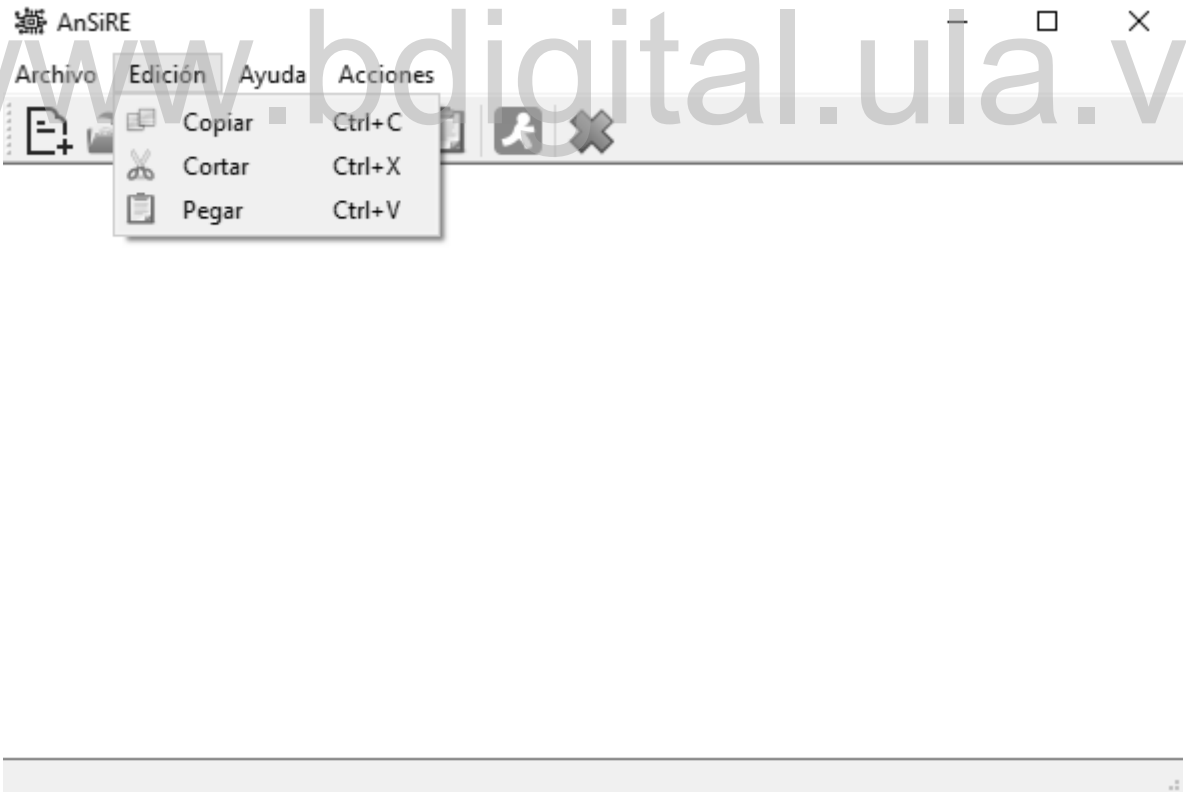

Figura 3.3. Menú Edición

3.3.3 Menú ayuda

Este menú ayuda cuenta con tres pestañas: Acerca de, acerca de QT y acerca del programa. Tal como se ve en la figura 3.4.

La pestaña "*acerca de*" abre una ventana con los datos del autor y tutor de la tesis. El prototipo función de codificada que se encarga de realizar esta acción es la siguiente:

```
void on_actionAcerca_de_triggered();
```

La pestaña "*acerca de QT*" abre una ventana con los datos, versión y características del entorno de desarrollo Qt. El prototipo de función codificada que se encarga de realizar esta acción es la siguiente:

```
void on_actionAcerca_de_QT_triggered();
```

La pestaña "*acerca del programa*" muestra en la interfaz información acerca de cómo usar el analizador simbólico de circuitos, consideraciones, etc. El prototipo de función codificada que se encarga de realizar esta acción es la siguiente:

```
void on_actionAcerca_del_Programa_triggered();
```

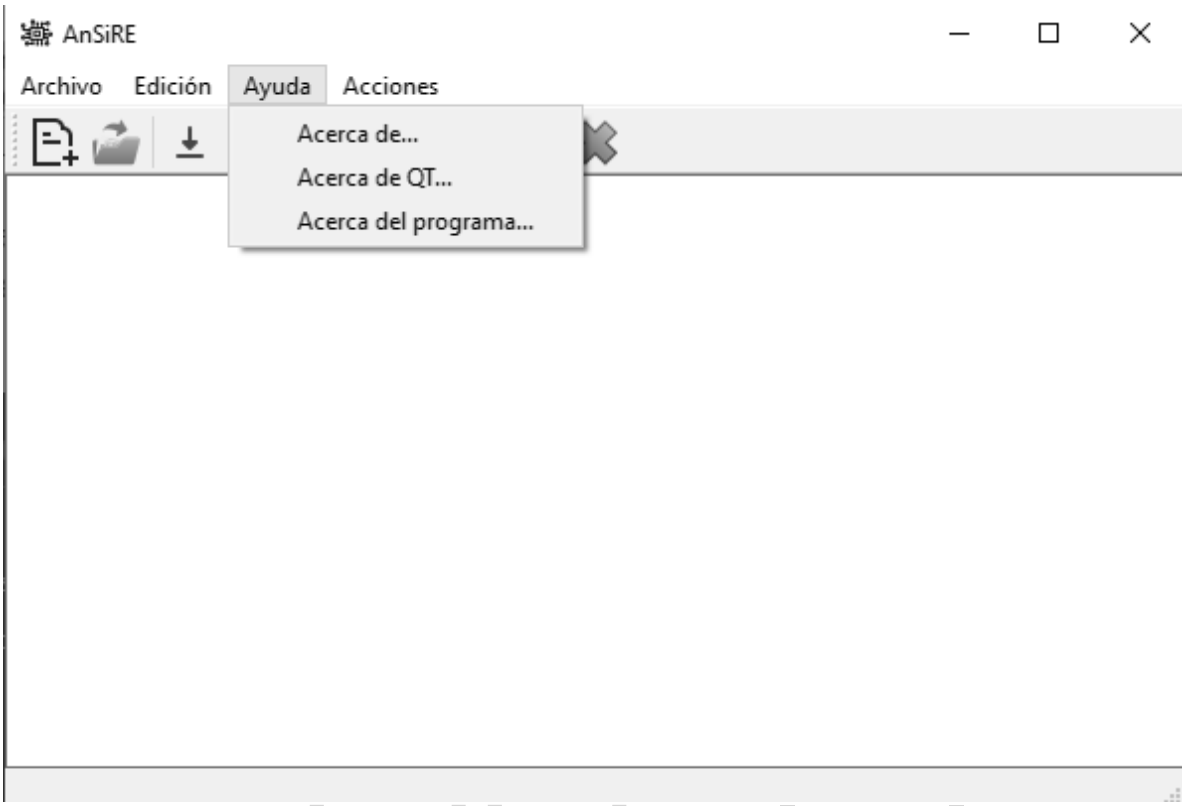

Figura 3.4. Menú Ayuda

3.3.4 Menú Acciones

Cuenta con una sola pestaña llamada ejecutar, tal como se ve en la figura 3.5

La pestaña "ejecutar" cuyo atajo rápido de teclado es F10, llama a las funciones correspondientes para ejecutar el programa AnSiRE-MAI. En un principio, si se trata de un archivo nuevo, al momento de que se llame la función ejecutar, se le pedirá al usuario que coloque un nombre al archivo, este debe guardarse con formato .cir, automáticamente se llama al .exe que contiene el software AnSiRE-MAI mediante el comando system(), se realiza los respectivos cálculos y finalmente se carga en la interfaz los resultados obtenidos. En caso de que el archivo ya haya sido guardado, o que el mismo se haya abierto haciendo uso de la función abrir, al presionar ejecutar no se le pedirá al usuario que coloque un nombre al fichero pues este ya está establecido por lo que ocurrirán los pasos mencionados con anterioridad sin necesidad de guardar el fichero. El prototipo de función codificada que se encarga de realizar esta acción es la siguiente:

```
void on_actionEjecutar_triggered();
```

Figura 3.5. Menú Acciones

3.3.5 Barra de herramientas

El entorno cuenta también con una barra de acceso rápido en el que se puede realizar alguna de las acciones mencionadas en los menús, ubicando el cursor del ratón y haciendo clic izquierdo sobre alguna de las opciones, tal como se ve en la figura 3.6, tal barra de herramientas se puede seleccionar y mover a cualquiera parte de la pantalla si así se desea

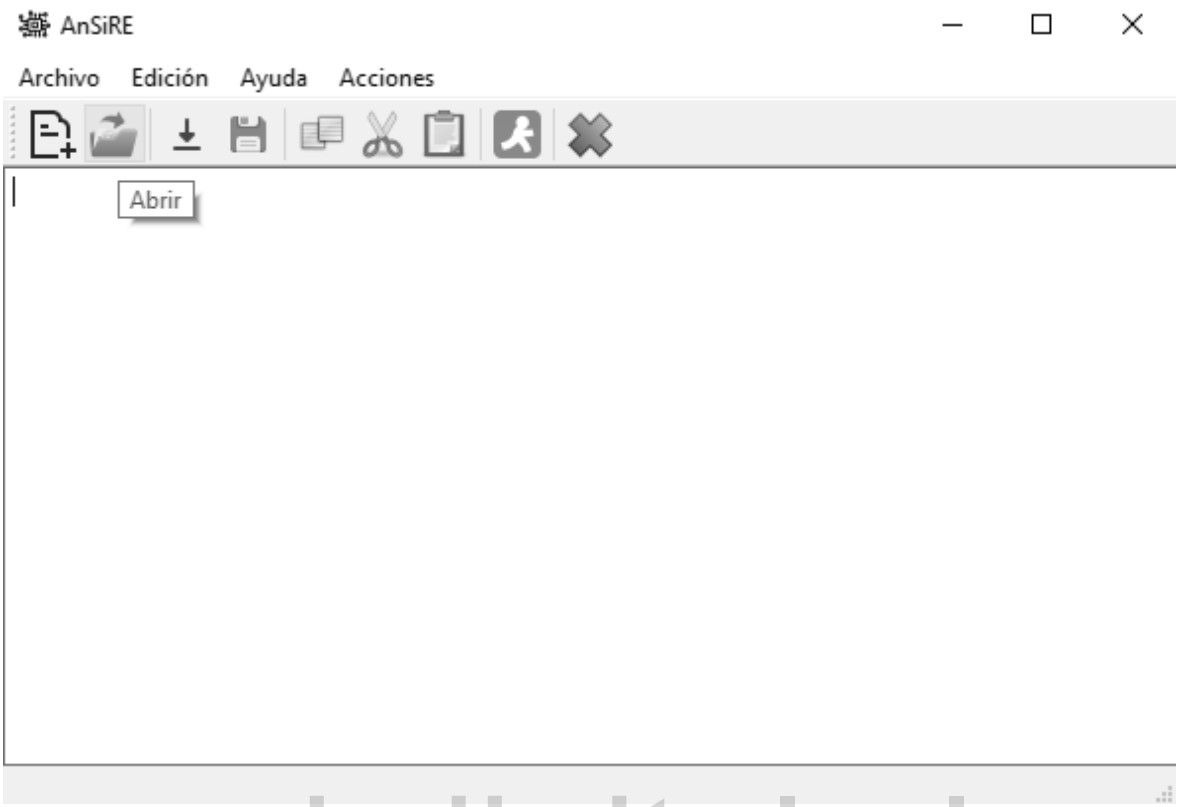

Figura 3.6. Barra de herramientas

3.4 NUEVO FORMATO DE DECLARACIÓN Y CORRECCIÓN DE ERRORES

A lo largo del desarrollo del software se hicieron múltiples correcciones a las rutinas existentes, así como también se agregaron nuevas rutinas para su optimo desempeño, he aquí alguna de ellas.

3.4.1 Nuevo formato de declaración de las fuentes independientes DC

Anteriormente, la declaración de las fuentes independientes tanto de voltaje como de corriente en DC se hacían de la siguiente manera:

```
Nombre del elemento   Nodo 1   Nodo 2   Valor
```

A lo que se agregaron nuevas fuentes con señales variables en el tiempo, se colocó una condición extra para indicar el tipo de señal con la que se va a operar, y poder así declarar

cualquier tipo de fuente de tensión o corriente con su carácter clave “V” o “I”, su declaración ahora es de la siguiente manera:

```
Nombre del elemento   Nodo 1   Nodo 2   Tipo de fuente   valor
```

El cual sus valores por defectos se muestran en la tabla 3.7

Tabla 3.7. Valores por defecto al declarar una fuente DC.

Parámetro	Valor por defecto	Definición/observaciones
Nombre del elemento	obligatorio	Debe iniciar con su letra clave “V” o “I” (dependiendo de si es una fuente de tensión independiente o fuente de corriente de independiente) seguido de caracteres para declarar e identificar al elemento
Nodo 1	obligatorio	Nodo donde se ubicará el terminal positivo de la fuente de señal DC
Nodo 2	obligatorio	Nodo donde se ubicará el terminal negativo de la fuente de señal DC
Tipo de fuente	obligatorio	Caracteres claves que indican el tipo de fuente. Para señal DC sus caracteres clave son DC.
Valor	obligatorio	Amplitud de la señal

Para dejar claro el cómo declarar el elemento, se parte de la siguiente declaración:

```
V1 1 2 DC 5
```

El nombre de la fuente es V1, la misma está ubicada entre los nodos 1 y 2, es de tipo DC y tiene una amplitud de 5V.

3.4.2 Corrección a fuentes DC

Un detalle que se tuvo que corregir respecto al algoritmo de las fuentes DC es que cuando se trabajaban con circuitos con elementos reactivos o circuitos que se operaran en el dominio de la frecuencia, la señal DC se emulaba como una señal impulso en vez de una señal constante, trayendo consigo un error de cálculo al operar con circuitos bajo las condiciones anteriores.

La manera más eficiente de corregir este detalle fue que justo antes de que se pasaran los datos a Mathomatic, se analizará si el circuito se trabajará en el dominio de la frecuencia para dividir el valor de cada una de las fuentes de tensión DC entre S. La lógica empleada para conocer si se trabajará en el dominio de la frecuencia es mediante una variable centinela cuyo valor por defecto es 0, indicando tal número que no se trabajará en el dominio de la frecuencia, y cambiando a 1 si se trabajará en el dominio de la frecuencia

Tal variable centinela cambia a 1 si se crea un capacitor, inductor o cualquier fuente variable en el tiempo. Esta variable es pasado por valor a la función inicia_simbolico, contenido la misma en la biblioteca guardar.c, el cual la misma es la que se encarga de llevar el valor de cada uno de los elementos a punto flotante (en caso de que los mismos no sean simbólicos), a la función se le tuvo que agregar un nuevo parámetro de tipo entero el cual es por donde se envía la variable centinela mencionada, quedando de la siguiente manera:

```
void        inicia_simbolico(struct        REGISTRO_ELEMENTO
*lista_elemento,  struct  REGISTRO_COLECCION  *baul,  struct
eeSTATE *EST, FILE *archivo, int centinela_s);
```


Ahora a tal función se le agregó la siguiente estructura de decisión para llevar las fuentes DC al dominio de la frecuencia en caso de requerirlo:

```
if(string2[0] == 'V' && centinela_s == 1&& strcmp(aux-
>nodo_3,"DC") == 0)
        strcat(aux->valor,"/s");
```

Se observa que a la cadena de caracteres aux->valor se le concatena los caracteres /S bajo la siguiente premisa:

Si la primera letra del nombre del elemento es ‘V’ (indicando la misma que una fuente de tensión), si la variable centinela_s == 1 (indicando este que el circuito se trabajará en el dominio de la frecuencia) y si en la variable aux->nodo_3 está contenido la cadena de caracteres “DC” (indicando que la fuente de tensión es de corriente directa), si estas tres condiciones se cumplen se procede a concatenar la cadena en cuestión.

Esta misma corrección se hizo también para las fuentes independientes de corriente.

3.4.3 Corrección a la cantidad máxima de caracteres en parámetros

Al momento de que se declara un elemento con sus respectivos parámetros, cada parámetro puede tener hasta un máximo de 15 caracteres, si se sobrepasa tal valor al declarar algún parámetro del elemento, a lo que es un registro dinámico y los espacios de memoria que se reservan son consecutivas, se genera un desborde ocasionando que los caracteres restantes se almacenen y ocupen memoria de otro parámetro, esto ocasiona al momento de la compilación un error semántico. Por ejemplo, si se quiere declarar una fuente de tensión DC se sabe que la sintaxis es la siguiente:

```
Vnombre   nodo1   nodo2   tipo de fuente   valor
```

Se ha de agregar cinco parámetros para la creación de la misma

Cada uno de esos parámetros puede contener hasta 15 caracteres a excepción del parámetro (tipo de fuente) el cual para el caso de una fuente DC los datos escritos allí ha de ser estrictamente DC, de lo contrario se arrojará un mensaje indicando que la fuente no se declaró correctamente pues no se conocería con que fuente se va a trabajar.

Sabiendo esto, si la fuente DC se declara de la siguiente manera:

```
Vmasdequincecaracteres nodo1 nodo2 DC 5
```

A lo que el parámetro uno el cual es el nombre del elemento tiene más de quince caracteres, solo los primeros quince caracteres se almacenaran en sus espacios de memoria reservados, el resto se almacenara en espacios de memoria que no le pertenecen generándose así el error semántico, lo mismo ocurre con el resto de los parámetros.

Esto es una limitante no significativa para parámetros como, nombre del elemento, nodos, etc. debido a que no es nada usual declarar tales parámetros con tal magnitud de caracteres, pero si es perjudicial al momento de que se declare el valor de los elementos en cuestión, puesto que, por ejemplo, si se declara una resistencia de 10.000.000 ohm, el software en su diseño considera hasta cinco decimales y los separadores de miles, el punto, y el separador de decimal, la coma, también se cuenta como carácter en la cadena, por lo que para tal resistencia al ser leído por el software quedaría su valor de la siguiente manera: 10.000.000,00000, se observa que hay 17 caracteres, por lo que se generará el error mencionado más arriba, trayendo consigo que no se pudiese declarar para este ejemplo

valores de resistencia mayores a 9.999.999. Aparte, este detalle también toma fuerza al declarar fuentes variables en el tiempo puesto que su valor al llevarlo al dominio de la frecuencia es más que seguro que habrán más de 15 caracteres. Por lo que para este parámetro que es el que almacena el valor se incrementó a 100 caracteres.

Si bien es una leve modificación, haber entendido el problema del mismo tomo un tiempo considerable pues siendo este un error semántico se tuvo que analizar con detalle la causa del mismo.

Tal modificación se hizo en la biblioteca elementos.h, quedando de la siguiente manera:

```
struct  REGISTRO_ELEMENTO{
    char elemento[15];
    char valor[100];
    char nodo_1[15];
    char nodo_2[15];
    char nodo_3[15];
    char nodo_4[15];
    char freq_1[15];
    char fase_1[15];
    char param8[15];

    struct MATRIZ **matriz
    struct REGISTRO_NODO *listaNodosAsociada;
    int numeroNodosAsociados;

    char cofactor1Asociado[MAX_LONG_LINEA];
    char cofactor2Asociado[MAX_LONG_LINEA];

    struct REGISTRO_ELEMENTO *sig;
};
```


En este mismo registro de tipo estructura se puede modificar la cantidad de caracteres de todos los parámetros.

3.4.4 Formato de mensajes de errores

Se agregaron nuevos mensajes de errores en caso de que el usuario declare un elemento de manera errónea o se generé algún fallo durante la ejecución, esto con la finalidad de que no se generen cierres bruscos en el software y que el usuario conozca el motivo por el cual la simulación no se pudo llevar a cabo.

3.5 DETALLES DE IMPLEMENTACIÓN

Los algoritmos diseñados fueron implementados usando el entorno de desarrollo Qt 4.9.1, bajo el lenguaje de programación C, cuyo compilador fue el MinGW 5.13.0 en su versión de 32 bits.

La generación del .exe del software AnSiRE-MAI se realizó mediante el entorno de desarrollo Dev-C++, bajo el lenguaje de programación C, cuyo compilador fue el TDM-GCC 4.9.2 32-bit. Esto, debido a que el peso del ejecutable que se generaba en comparativa con Qt era mucho menor, apenas alcanzando los 1mb, por lo que la programación de los algoritmos se hizo en Qt, pero él .exe se extrajo de Dev-C++ pasa así hacer el programa más ligero.

La construcción de la interfaz gráfica de usuario se hizo mediante el entorno de desarrollo Qt 4.9.1, haciendo usos de las herramientas "*Widget*" de Qt bajo los lenguajes de programación C, C++ y QML.

La interfaz gráfica de usuario y el software AnSiRE-MAI son dos programas el cual se entrelazan haciendo uso de la función system().

El instalador del software se creó mediante una de las librerías de desarrollo de Qt llamada *Qt Installer Framework* versión 4.3.0.

CAPÍTULO 4 COMPROBACIÓN DE FUNCIONAMIENTO

Para la comprobación del funcionamiento del software, se trabajaron algunos circuitos de prueba tanto simbólicos como numéricos y los resultados numéricos se compararon el simulador Multisim.

A su vez, se instaló el software en diversos equipos con el sistema operativo Windows de diferentes versiones (Windows 7, Windows 10) tanto de 32 como de 64 bits, operando el mismo de manera correcta.

Si bien, el software es robusto, puede que exista la posibilidad de que un circuito propuesto no arroje los resultados correctos, esto se debe a que, como todo software, su iniciación pasa por una fase beta en el que los usuarios que utilicen el mismo detecten posibles errores, los reporten y con base a ellos realizar las respectivas depuraciones y actualizaciones.

4.1 CIRCUITO RESISTIVO CON INTERRUPTOR DE UNA VÍA Y DE DOS VÍAS

Se parte del circuito representado en la figura 4.1

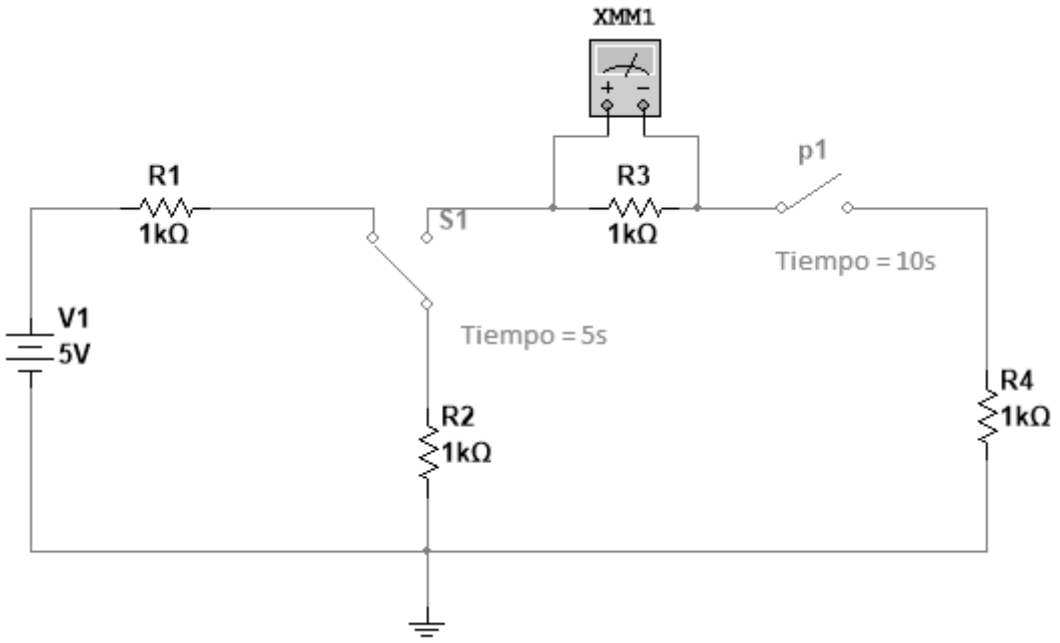

Figura 4.1. Circuito de prueba "circuitoPS.cir"

Para el siguiente ejemplo, se busca conocer el voltaje en la resistencia R3 bajo tres condiciones: la primera cuando se tiene el circuito tal cual como se refleja en la figura 4.1, la segunda cuando el interruptor de dos vías S1 cambia de posición pasado los 5 segundos y la tercera cuando el interruptor de una vía P1 se abre pasado los 10 segundos, todo esto considerando que:

La resistencia encendido (Ron) para S1 y P1 es igual a 100Ω

La resistencia apagado (Roff) para S1 es P1 igual a 10.000.000Ω

Los resultados se muestran a continuación:

```
*CIRCUITOPS.CIR

V1 1 7 DC 5
R1 1 2 1000
S1 2 3 4 5 100 10000000
R2 3 7 1000
R3 4 5 1000
P1 5 6 ON 10 100 10000000
R4 6 7 1000

.PLOT V(R3)
```

```
*******************************************************************************
*                            UNIVERSIDAD DE LOS ANDES                         *
*                              FACULTAD DE INGENIERIA                         *
*                          ESCUELA DE INGENIERIA ELECTRICA                    *
*                                                                             *
*                                                                             *
*        Ampliación de las Capacidades de Cálculo para el Analizador          *
*      Simbólico de Circuitos usando la matriz de Admitancia Indefinida       *
*  Trabajo de grado presentado como requisito parcial para optar al título    *
*                           de Ingeniero Electricista                         *
*                                                                             *
*                                                                             *
*                                                                             *
*  Autor: Jodrick Xavier Colina Tromp                                         *
*                                                                             *
*  Tutor: M. Sc. Francisco J. Viloria M.                                      *
*                                                                             *
*                                                                             *
*  Mérida 2022                                                                *
*******************************************************************************

*CIRCUITOPS.CIR

V1 1 7 DC 5
R1 1 2 1000
S1 2 3 4 5 100 10000000
R2 3 7 1000
R3 4 5 1000
P1 5 6 ON 10 100 10000000
R4 6 7 1000

.PLOT V(R3)

0 < t < 5.000000

-----------------------
Nodos del circuito:
         Nombre          # Nodo
              1               1
              2               2
              3               3
              4               4
              5               5
              6               6
              7               7

-----------------------
Lista de elementos:
       elemento           nodo+           nodo-          estado          tiempo
             P1               5               6        CERRADO              10

       elemento           nodo+           nodo-           valor
             R1               1               2     1000.000000
             R2               3               7     1000.000000
             R3               4               5     1000.000000
             R4               6               7     1000.000000

       elemento           nodo+    nodo-(comun)          estado          tiempo
             S1               2               3        CERRADO               5
            S1C               4               3        ABIERTO               5

       elemento           nodo+           nodo-           valor
             V1               1               7        5.000000
```

```
---------------------------------------------------------------------

---> COMANDO .PLOT V(R3)

Para V(R3) el resultado es:  0.00023803278282698

              .PLOT V(R3) =  0.00023803278282698

---------------------------------

5.000000 < t < 10.000000

-----------------------
Nodos del circuito:
         Nombre          # Nodo
              1               1
              2               2
              3               3
              4               4
              5               5
              6               6
              7               7

-----------------------
Lista de elementos:
       elemento           nodo+           nodo-          estado          tiempo
             P1               5               6         CERRADO              10

       elemento           nodo+           nodo-           valor
             R1               1               2    1000.000000
             R2               3               7    1000.000000
             R3               4               5    1000.000000
             R4               6               7    1000.000000

       elemento           nodo+    nodo-(comun)          estado          tiempo
             S1               2               3         ABIERTO               5
            S1C               4               3         CERRADO               5

       elemento           nodo+           nodo-           valor
             V1               1               7        5.000000


---------------------------------------------------------------------

---> COMANDO .PLOT V(R3)

Para V(R3) el resultado es:  0.00015622363726121

              .PLOT V(R3) =  0.00015622363726121

---------------------------------

t >= 10.000000

-----------------------
Nodos del circuito:
         Nombre          # Nodo
              1               1
              2               2
              3               3
              4               4
              5               5
```

```
              6               6
              7               7

-----------------------
Lista de elementos:
       elemento           nodo+           nodo-          estado          tiempo
             P1               5               6         ABIERTO              10

       elemento           nodo+           nodo-           valor
             R1               1               2     1000.000000
             R2               3               7     1000.000000
             R3               4               5     1000.000000
             R4               6               7     1000.000000

       elemento           nodo+    nodo-(comun)          estado          tiempo
             S1               2               3         ABIERTO               5
            S1C               4               3         CERRADO               5

       elemento           nodo+           nodo-           valor
             V1               1               7        5.000000

---------------------------------------------------------------------

---> COMANDO .PLOT V(R3)

Para V(R3) el resultado es:  4.9974510401021E-08

              .PLOT V(R3) =  4.9974510401021E-08

--------------------------------
```


Los resultados obtenidos se pueden verificar de manera manual o mediante algún simulador.

Al instalar el software, en la ruta donde el usuario guardó al mismo, habrá una carpeta llamada “ejemplos” donde se ubica los archivos .cir, .out utilizados para los ejemplos mostrados.

4.2 CIRCUITO CON SEÑAL SINUSOIDAL

Para el siguiente ejemplo, se realizará el análisis de un circuito resistivo sencillo con una fuente de tensión sinusoidal, tal como se ve en la figura 4.2, a la fuente se le agregará un valor offset y se utilizará el método de superposición para ver el aporte tanto de la señal variable como del offset.

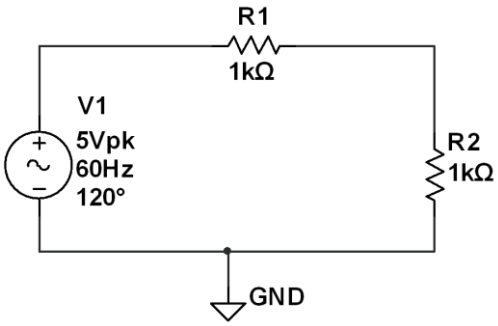

Figura 4.2. Circuito de prueba "circuitoSIN.cir"

Se describe el circuito y se definen los comandos para realizar los cálculos:

```
*CIRCUITOSIN.CIR

V1 1 3 SIN 5 5 60 120
R1 1 2 1000
R2 2 3 1000

.PLOT V(R1)
.SP

*******************************************************************************
*                          UNIVERSIDAD DE LOS ANDES                           *
*                            FACULTAD DE INGENIERIA                           *
*                     ESCUELA DE INGENIERIA ELECTRICA                         *
*                                                                             *
*                                                                             *
*        Ampliación de las Capacidades de Cálculo para el Analizador          *
*      Simbólico de Circuitos usando la matriz de Admitancia Indefinida       *
*  Trabajo de grado presentado como requisito parcial para optar al título    *
*                          de Ingeniero Electricista                          *
*                                                                             *
*                                                                             *
*                                                                             *
*  Autor: Jodrick Xavier Colina Tromp                                         *
*                                                                             *
*  Tutor: M. Sc. Francisco J. Viloria M.                                      *
*                                                                             *
*                                                                             *
*  Mérida 2022                                                                *
*******************************************************************************

*CIRCUITOSIN.CIR

V1 1 3 SIN 5 5 60 120
R1 1 2 1000
R2 2 3 1000

.PLOT V(R1)
.SP
-----------------------
Nodos del circuito:
         Nombre          # Nodo
              1               1
              2               2
```

```
             3               3
        offset1              4

-----------------------
Lista de elementos:
elemento           nodo+      nodo-           valor
    R1               1          2          1000.000000
    R2               2          3          1000.000000

elemento           nodo+     nodo-             valor
   V1             offset1     3       (5*S*0.58061+5*60*0.81418)/(S^2+60*60)
  VOS1              1       offset1         5.000000

---------------------------------------------------------------------

---> COMANDO .PLOT V(R1)

APORTE DE LA FUENTE  V1

                             (((1.451525*S) + 122.127))
             Aporte de V1 = --------------------------
                                   ((3600 + S^2))


APORTE DE LA FUENTE  VOS1

             Aporte de VOS1 = 2.5

Para V(R1) el resultado es: ((9122.127 + (2.5*S^2) + (1.451525*S)))/((3600 +
S^2))
                            ((9122.127 + (2.5*S^2) + (1.451525*S)))
             .PLOT V(R1) = ---------------------------------------
                                      ((3600 + S^2))

---------------------------------
```


En los resultados, se puede ver el aporte de la fuente variable en el tiempo, como del offset agregado a la señal y finalmente el resultado total esperado.

4.3 CIRCUITO CON AMPLIFICADOR OPERACIONAL Y SEÑAL CUADRADA

El siguiente ejemplo consta de un amplificador operacional tipo inversor el cual cuya señal a amplificar es una señal de onda cuadrada, el esquema de montaje se puede visualizar en la figura 4.3.

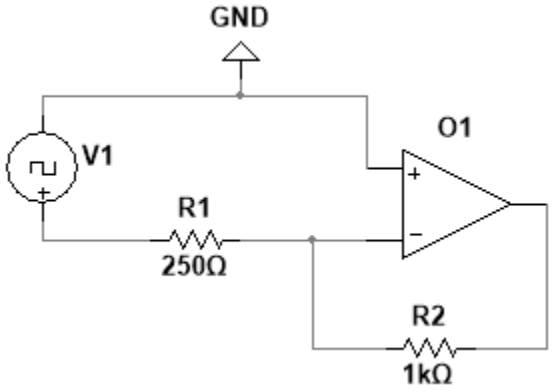

Figura 4.3. Circuito de prueba "circuitoAO.cir"

Se procede a describir el circuito y sus respectivos comandos, obsérvese que se tiene expresiones simbólicas por lo que su resultado se expresará de dicha forma:

```
*CIRCUITOAO.CIR

V1 1 2 SQR 0 A T
R1 1 MIN 250
R2 MIN VO 1000
O1 2 MIN VO 2

.PLOT V(VO,2)
```



```
********************************************************************************
*                          UNIVERSIDAD DE LOS ANDES                            *
*                           FACULTAD DE INGENIERIA                             *
*                       ESCUELA DE INGENIERIA ELECTRICA                        *
*                                                                              *
*                                                                              *
*        Ampliación de las Capacidades de Cálculo para el Analizador           *
*      Simbólico de Circuitos usando la matriz de Admitancia Indefinida        *
*  Trabajo de grado presentado como requisito parcial para optar al título     *
*                         de Ingeniero Electricista                            *
*                                                                              *
*                                                                              *
*                                                                              *
*  Autor: Jodrick Xavier Colina Tromp                                          *
*                                                                              *
*  Tutor: M. Sc. Francisco J. Viloria M.                                       *
*                                                                              *
*                                                                              *
*  Mérida 2022                                                                 *
********************************************************************************

*CIRCUITOAO.CIR

V1 1 2 SQR 0 A T
R1 1 MIN 250
R2 MIN VO 1000
O1 2 MIN VO 2

.PLOT V(VO,2)
-----------------------
```

```
Nodos del circuito:
         Nombre          # Nodo
              1               1
              2               2
            MIN               3
             VO               4
          x__O1               5

-----------------------
Lista de elementos:
       elemento           nodo+           nodo-           valor
             R1               1             MIN      250.000000
             R2             MIN              VO     1000.000000
          Ri__O1              2             MIN          Ri__O1
       Rmas__O1           x__O1               2        1.000000
       Rmen__O1           x__O1              VO       -1.000000

       elemento           nodo+           nodo-           valor
             V1               1               2        A*alfa/S

elemento        nodo+           nodo-           cont+      cont-     valor
   O1             2             MIN             VO         2     InfinitoO1

-----------------------------------------------------------------

---> COMANDO .PLOT V(VO,2)

Para V(VO,2) el resultado es: (1*(1-e^-ST)^-1)*((-4*A*ALFA)/(S))

                              (1*(1-e^-ST)^-1)*((-4*A*ALFA)
               .PLOT V(VO,2) = ----------------------------
                                          (S))
--------------------------------
```


La expresión final obtenida se observa que es multiplicada por -4, el cual es la ganancia por la cual la señal cuadrada se amplificará, de igual manera ocurre con las diversas señales variables en el tiempo, como es de esperarse.

4.4 CIRCUITO CON ELEMENTO REACTIVO

El siguiente ejemplo consta de una fuente variable en el tiempo en serie con una resistencia y un capacitor, tal cual como se ve en la figura 4.4

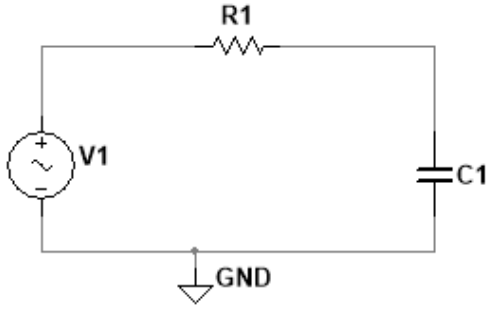

Figura 4.4 Circuito de prueba "circuitoER.cir"

Se procede a describir el circuito y sus comandos, para efectos de este ejemplo todos los parámetros serán simbólicos:

```
*CIRCUITOER.CIR

V1 1 3 SIN OFFSET AMP FRE FASE
R1 1 2 RES
C1 2 3 C

.PLOT V(R1) V(C1)

********************************************************************************
*                            UNIVERSIDAD DE LOS ANDES                          *
*                             FACULTAD DE INGENIERIA                           *
*                         ESCUELA DE INGENIERIA ELECTRICA                      *
*                                                                              *
*                                                                              *
*        Ampliación de las Capacidades de Cálculo para el Analizador           *
*      Simbólico de Circuitos usando la matriz de Admitancia Indefinida        *
*  Trabajo de grado presentado como requisito parcial para optar al título     *
*                           de Ingeniero Electricista                          *
*                                                                              *
*                                                                              *
*                                                                              *
*  Autor: Jodrick Xavier Colina Tromp                                          *
*                                                                              *
*  Tutor: M. Sc. Francisco J. Viloria M.                                       *
*                                                                              *
*                                                                              *
*  Mérida 2022                                                                 *
********************************************************************************
```



```
*CIRCUITOER.CIR

V1 1 3 SIN OFFSET AMP FRE FASE
R1 1 2 RES
C1 2 3 C

.PLOT V(R1) V(C1)
-----------------------
Nodos del circuito:
         Nombre          # Nodo
              1               1
              2               2
              3               3
        offset1               4
-----------------------
Nodos del circuito:
         Nombre          # Nodo
              1               1
              2               2
              3               3
        offset1               4

-----------------------
Lista de elementos:
elemento           nodo+           nodo-           valor
   C1               2               3               C
   R1               1               2             RES

elemento   nodo+     nodo-                          valor
```

```
   V1      offset1       3          (AMP*S*sinFASE+AMP*FRE*cosFASE)/(S^2+FRE*FRE)
  VOS1        1       offset1                        OFFSET/s
-----------------------------------------------------------------------

---> COMANDO .PLOT V(R1)

Para V(R1) el resultado es: (RES*C*((S^2*((AMP*SINFASE) + OFFSET)) +
(S*AMP*FRE*COSFASE) + (OFFSET*FRE^2)))/(((1 + (RES*C*S))*(S^2 + FRE^2)))



         (RES*C*((S^2*((AMP*SINFASE)+OFFSET))+(S*AMP*FRE*COSFASE)+(OFFSET*FRE^2)))
.PLOT V(R1) = ------------------------------------------------------------------
                             (((1 + (RES*C*S))*(S^2 + FRE^2)))

---------------------------------
---------------------------------

---> Comando .PLOT V(C1)

Para V(C1) el resultado es: (((S^2*((AMP*SINFASE) + OFFSET)) +
(S*AMP*FRE*COSFASE) + (OFFSET*FRE^2)))/(((S^3 + (FRE^2*S))*((S*C*RES) + 1)))


             (((S^2*((AMP*SINFASE) + OFFSET))+(S*AMP*FRE*COSFASE)+(OFFSET*FRE^2)))
.PLOT V(C1) = ------------------------------------------------------------------
-                              (((S^3 + (FRE^2*S))*((S*C*RES) + 1)))

---------------------------------
```


Las expresiones que se obtienen son bastante abstractas si se trabaja con valores completamente simbólicos, sin embargo, con esto se comprueba la capacidad del software.

4.5 CIRCUITO CON INTERRUPTOR CON TIEMPO SIMBÓLICO

El siguiente ejemplo consta de un circuito RC tal como se ve en la figura 4.5, en el que tanto el tiempo del interruptor como las resistencias de ON y OFF serán simbólicos.

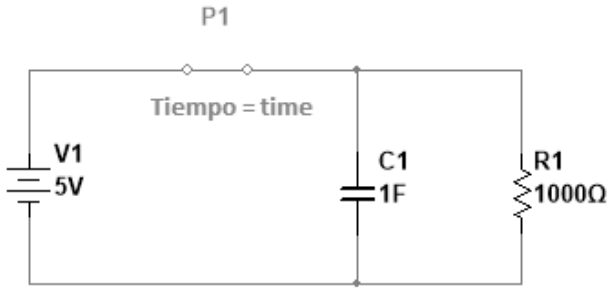

Figura 4.5 Circuito de prueba "circuitoISR.cir"

Se procede a describir el circuito y definir los comandos:

```
*CIRCUITOISR.CIR

V1 1 T DC 5
P1 1 2 ON TIME RON ROFF
C1 2 T 1
R2 2 T 1000

.PLOT V(R2)

********************************************************************************
*                           UNIVERSIDAD DE LOS ANDES                           *
*                            FACULTAD DE INGENIERIA                            *
*                        ESCUELA DE INGENIERIA ELECTRICA                       *
*                                                                              *
*                                                                              *
*        Ampliación de las Capacidades de Cálculo para el Analizador           *
*      Simbólico de Circuitos usando la matriz de Admitancia Indefinida        *
*  Trabajo de grado presentado como requisito parcial para optar al título     *
*                          de Ingeniero Electricista                           *
*                                                                              *
*                                                                              *
*                                                                              *
*  Autor: Jodrick Xavier Colina Tromp                                          *
*                                                                              *
*  Tutor: M. Sc. Francisco J. Viloria M.                                       *
*                                                                              *
*                                                                              *
*  Merida 2022                                                                 *
********************************************************************************
```



```
*CIRCUITOISR.CIR

V1 1 T DC 5
P1 1 2 ON TIME RON ROFF
C1 2 T 1
R1 2 T 1000

.PLOT V(R1)


0 < t < TIME

-----------------------
Nodos del circuito:
         Nombre          # Nodo
              1              1
              2              2
              T              3

-----------------------
Lista de elementos:
       elemento           nodo+           nodo-           valor
             C1               2               T        1.000000
             P1               1               2         CERRADO            TIME
             R1               2               T     1000.000000

       elemento           nodo+           nodo-           valor
             V1               1               T             5/s

---------------------------------------------------------------------
```

```
---> COMANDO .PLOT V(R1)

Para V(R2) el resultado es: (5000)/(((S*(1000 + RON)) + (1000*S^2*RON)))

                                          (5000)
              .PLOT V(R1) = ---------------------------------------
                            (((S*(1000 + RON)) + (1000*S^2*RON)))

---------------------------------

t >= TIME

-----------------------
Nodos del circuito:
         Nombre          # Nodo
              1              1
              2              2
              T              3

-----------------------
Lista de elementos:
       elemento           nodo+           nodo-           valor
             C1               2               T        1.000000
             P1               1               2         ABIERTO            TIME
             R1               2               T     1000.000000

       elemento           nodo+           nodo-           valor
             V1               1               T             5/s

------------------------------------------------------------
---> COMANDO .PLOT V(R1)

Para V(R1) el resultado es: (5000)/(((S*(1000 + ROFF)) + (1000*S^2*ROFF)))

                                           (5000)
              .PLOT V(R1) = ----------------------------------------
                            (((S*(1000 + ROFF)) + (1000*S^2*ROFF)))

---------------------------------
```

Se observa que para valores menores que t < TIME la tensión que se refleja en R1 producto del capacitor en paralelo es descrito en función de la resistencia Ron del interruptor, tal como se ve en la expresión obtenida. Ahora, una vez que cambie el interruptor a OFF en t>= TIME, el capacitor se descargará en función de la resistencia Roff, y tal tensión se reflejará en la resistencia R1, esto asumiendo que Ron < Roff

4.6 INSTALACIÓN Y VISUALIZACIÓN DEL SOFTWARE

El instalador tiene un peso de 165 MB el cual es un ejecutable .exe, al momento de ejecutar el instalador, el cual lleva por nombre *AnSiREinstaller*, se abrirá una ventana en el que se

encuentra los pasos a seguir para llevar a cabo su correcta instalación, tal como se ve en la figura 4.6.

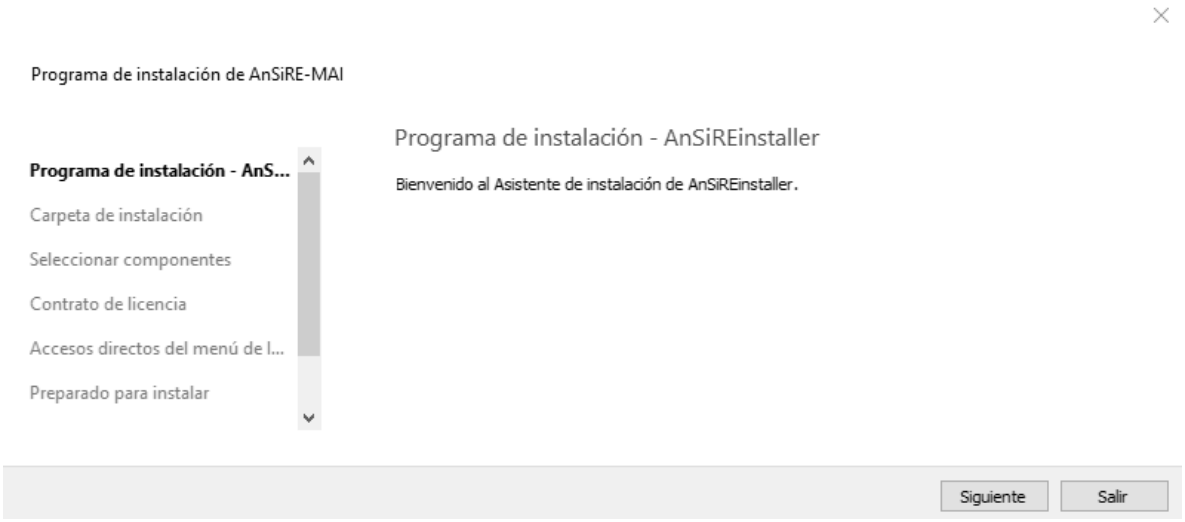

Figura 4.6 Menú de instalación

Una vez se haya seleccionado la ruta de la carpeta de instalación y se haya completado cada uno de los pasos que el menú de instalación solicita, el programa se habrá instalado correctamente. El programa ocupará aproximadamente 883.69 MB en el disco duro.

Finalizado la instalación, se habrá creado en el menú inicio su respectivo acceso directo para acceder al software, tal como se observa en la figura 4.7.

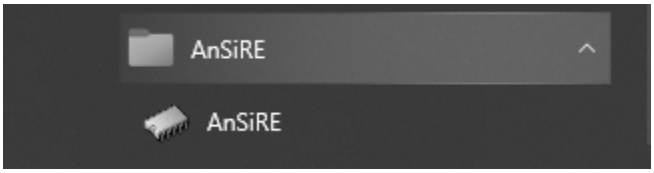

Figura 4.7 Carpeta y acceso directo del software en el menú de inicio

Al ejecutar el mismo se abrirá el software por el cual ya se puede hacer uso del mismo, se declaran los elementos, se definen los comandos y se ejecuta la simulación, un ejemplo de cómo se visualiza los datos agregados se visualiza en la figura 4.8

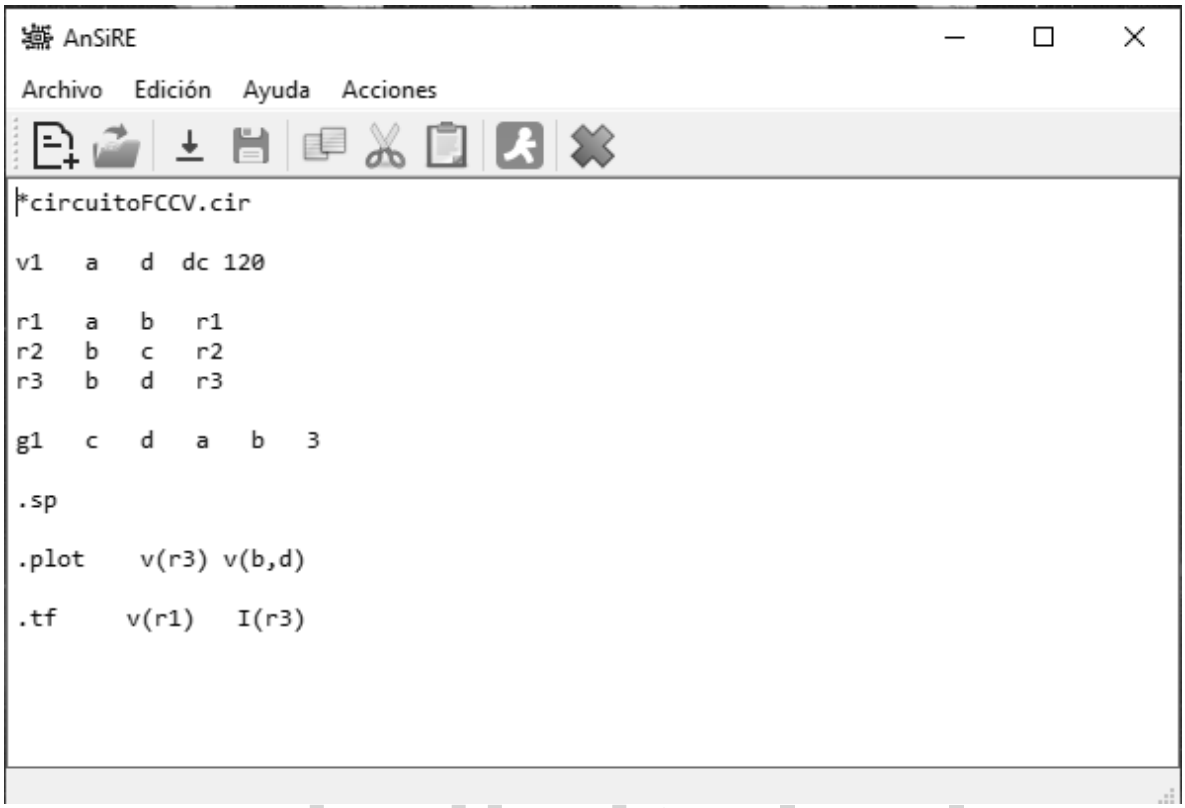

Figura 4.8 Visualización de los datos de entrada

Al ejecutar la simulación, se pedirá la ruta en donde se guardará el archivo .cir y .out, al finalizar la ejecución, automáticamente se reflejará los resultados en la interfaz tal como se visualiza en la figura 4.9

Todos los comandos presentes en la interfaz de usuario se explicaron a detalle en el capítulo 3 del presente trabajo.

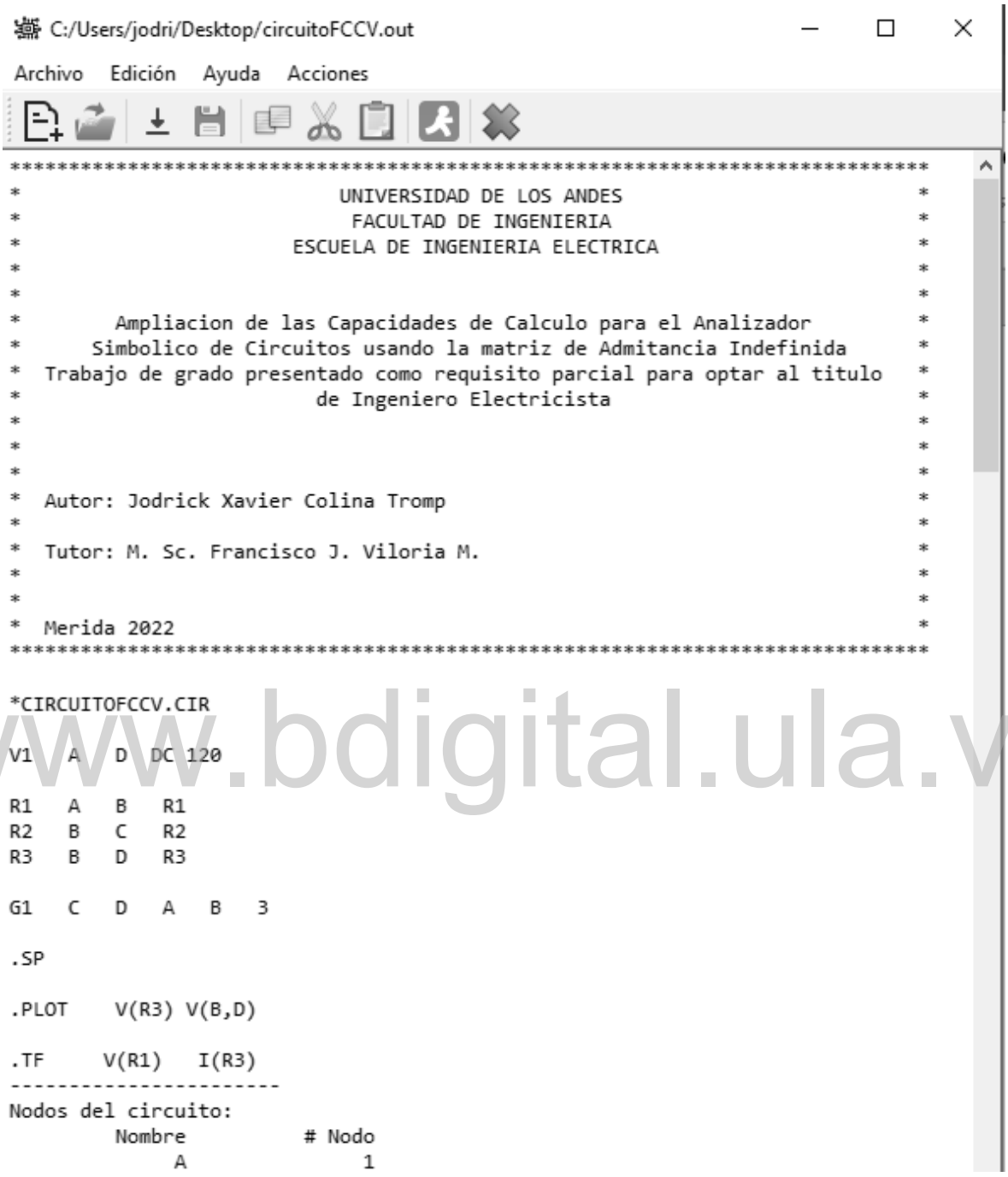

Figura 4.9 Visualización de los resultados

CONCLUSIONES

La ampliación de las capacidades de cálculo del AnSiRE-MAI se llevó a cabo de manera exitosa, cumpliendo así el objetivo general y cada uno de los objetivos específicos propuestos. Se agregaron al programa nuevos algoritmos, rutinas, así como mejoras a algunas funciones existentes para que así el software operé de manera exitosa.

El programa con las nuevas actualizaciones es capaz de realizar múltiples cálculos tanto numéricos, como simbólicos y mixto, conservando la velocidad de procesamiento y muestreo de los resultados que caracteriza al AnSiRE-MAI en comparativa a la versión anterior.

El algoritmo de los interruptores se implementó de manera satisfactoria, agregando funciones y condiciones para que así sus parámetros se puedan declarar de forma mixta, simbólica o numérica. A su vez, los interruptores están en la capacidad de considerar la resistencia intrínseca del elemento si el usuario así lo desea.

Para las fuentes variables en el tiempo, en algunos de los casos su implementación se tuvo que abordar desde otra perspectiva a como se había considerado en un principio puesto que los subprogramas de análisis de expresiones usados en el AnSiRE-MAI no están en la capacidad de abordar expresiones complejas producto de algunas de las nuevas funciones agregadas.

El nuevo entorno de escritura, lectura y ejecución del AnSiRE-MAI junto su instalador se realizó de manera satisfactoria, agregando así una mejora significativa no solo en la parte de escritura y visualización de los datos, si no también, en el uso del software puesto que el mismo se puede instalar en cualquier ordenador y comenzar a usarlo inmediatamente sin ningún tipo de complicaciones.

Por último, cada uno de los algoritmos, rutinas y funciones implementadas se hizo de tal manera que sus futuras actualizaciones y mejoras se puedan realizar sin mayores complicaciones puesto que se programó en lo posible de manera estructurada y secuencial.

RECOMENDACIONES

Para las distintas actualizaciones futuras, se sugiere un nuevo procesador de cálculo simbólico distinto a Mathomatic puesto que, el mismo presenta diversas limitantes que se hacen evidentes al momento que la complejidad de las expresiones a analizar incrementa, tal como fue en el caso de las funciones variables en el tiempo.

A su vez, se recomienda implementar funciones o agregar algún subprograma que permitan llevar los resultados obtenidos en el dominio de la frecuencia al dominio del tiempo.

El software AnSiRE-MAI, si bien está completamente funcional y cumple su función, la versión del mismo es una fase beta, por lo que se hace necesario hacer una gran cantidad de pruebas con diversos circuitos, para que, por medio de ello, los posibles errores que surjan en algunos casos particulares se puedan corregir, depurando así el programa y permitiendo al mismo mejorar en cuanto sus capacidades.

El nuevo motor de cálculo es prometedor, por lo que se insta a seguir ampliando las capacidades del software, agregando nuevas señales, condiciones, así como nuevos métodos de análisis de circuitos como son los circuitos con fasores.

REFERENCIAS

[1] O. Ruiz, “Análisis Simbólico de Circuitos. Parte 1: Motor de Cálculo”, Mérida, Venezuela: ULA, 2017.

[2] J. Gonzalez, “Análisis Simbólico de Circuitos usando la Matriz de Admitancia Indefinida”, Merida, Venezuela: ULA, 2022.

[3] C. Alexander y M. Sadiku. “Fundamentos de circuitos eléctricos”. Mexico. McGraw-Hill. 2001.

[4] Electrical Technology, “Switch & Types of Switches – Construction, Working, Characteritics & Applications” 2020. [En linea]. Available: https://www.electricaltechnology.org/2014/11/types-of-switches-electrical.html. [Último acceso: 28 de mayo 2022]

[5] J. Escalante, “Ampliación de las capacidades de cálculo del analizador simbólico de redes AnSiRE.”, Mérida, Venezuela: ULA, 2019.

[6] The Qt Company, “Qt Widgets” 2022. [En linea]. Available: https://doc.qt.io/qt-5/qtwidgets-index.html. [Último acceso: 29 de Junio 2022]