BUAP

BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA

FACULTAD DE CIENCIAS DE LA ELECTRÓNICA

MAESTRÍA EN CIENCIAS DE LA ELECTRÓNICA
OPCIÓN EN AUTOMATIZACIÓN

“SCAPy, herramienta EDA para el análisis simbólico de circuitos eléctricos”

T E S I S

Presentada para obtener el título de:

Maestro en Ciencias de la Electrónica Opción en Automatización

Presenta:

Ing. Luis Cortés Ramírez

Directores:

Dr. Luis Abraham Sánchez Gaspariano (FCE-BUAP)

Dr. Carlos Leopoldo Pando Lambruschini (IFUAP-BUAP)

*Becario CONAHCYT 08/24

Benemérita Universidad Autónoma de Puebla

Facultad de Ciencias de la Electrónica

Maestría en Ciencias de la Electrónica
Opción en Automatización

SCAPy,
herramienta EDA para el análisis simbólico de circuitos eléctricos

Alumno: **Ing. Luis Cortés Ramírez**

Directores de Tesis: **Dr. Luis Abraham Sánchez Gaspariano (FCE-BUAP)**
Dr. Carlos Leopoldo Pando Lambruschini (IFUAP-BUAP)

Puebla, Puebla. 08/24.

Agradecimientos

Quiero agradecer a mis padres Marina Ramírez Sánchez y Carlos Cortés Medina por apoyarme de manera incondicional en todas mis decisiones, sin mis padres no podría haber conseguido toda esta trayectoria, y todo este proyecto de vida los amo mucho papás.

También quiero agradecer a Diana Laura Corona Cervantes por todo el amor y apoyo que he recibido como compañera y pareja en mi vida, gracias también por esa maravillosa confianza y motivación que me das en todas nuestras bonitas aventuras gracias por este hermoso proyecto de vida juntos te amo mucho.

Agradezco a Fermín Ramírez Sánchez y Caridad Ramírez Sánchez por todo el apoyo y cariño incondicional a lo largo de mi vida y en todo momento desde que tengo memoria los quiero mucho, agradezco a mi asesor de tesis el Dr. Luis Abraham Sánchez Gaspariano por toda esta trayectoria, preparación, paciencia y amistad. Quiero agradecer a todos mis profesores durante mi preparación en maestría, a la Dra. Olga, al Dr. Moisés, a la Dra. Aurora, Al Dr. Alexandre y al Dr. Jorge, muchas gracias por compartir su tiempo, y maravillosas enseñanzas que estoy seguro me servirán para toda la vida.

Agradezco a mis hermanas Ana y Yose por todo lo bueno que han hecho mi. Quiero agradecer a mis compañeros de maestría que en todo este tiempo nos hemos vuelto grandes amigos y estoy seguro de que grandes colegas. Compartí momentos increíbles muchas gracias a Juan, Fernanda, Ángel, Valentín, Steven, Michelle y Alfredo, Gracias a todos mis maestros que he tenido a lo largo de mi vida, agradezco a Josué Meléndez Lima y Orlando Aguilar Figueroa por toda su paciencia motivación y amistad cuando estuve en General Cable, Gracias a Santiago Juan Juárez Moreno por todas sus grandes enseñanzas cuando estuve en SIMEC.

Quiero agradecer a CONAHCyT por la beca recibida para poder realizar mis estudios de Maestría, ya que sin su apoyo este estudio no podría haberse llevado a cabo.

Finalmente, agradezco a dios por la salud y virtud que me da para hacer lo que más me gusta en este mundo.

Resumen

A continuación se presenta $SCAPy$ herramienta EDA para el análisis simbólico de circuitos eléctricos, esta herramienta es una librería escrita en Python [1] y disponible en PyPi, es de licencia libre y podrá ser usada dentro del entorno anaconda, colab o alguna otra distribución. Es importante expresar que esta librería trabaja en conjunto con otras de las muchas librerías disponibles para Python y escritas por la amplia comunidad. Como lo son NumPy, Sympy, Pandas, Symengine, Multiprocessing y Memory-profiler.

También es importante hacer notar que el desarrollo de esta librería, está encaminada al análisis simbólico, en donde se obtienen expresiones algebraicas que describen el comportamiento del modelo, que para fines de la tesis se refiere a modelos en los circuitos eléctricos, pero también se hace notar que esta librería puede devolver resultados solo de sistemas algebraicos lineales que el usuario introduzca.

Así mismo, el análisis simbólico está encaminado a soluciones obtenidas por análisis nodal modificado, y matrices de parseo por el método de solución DDD (Diagrama de Decisión de Determinantes) en donde ha demostrado reducir el tiempo de cómputo, que para análisis numérico no es una característica tan costosa computacionalmente como lo es en cálculos simbólicos, en donde se busca un tiempo de cómputo reducido.

Un análisis confiable, y una representación gráfica de lo que se está analizando, por último la forma de introducir un circuito dentro del entorno de Python es por una netlist.

Definiciones

- SCAPY –Análisis simbólico de circuitos en Python.
- netlist –Lista descriptiva del circuito eléctrico.
- MNA –Análisis nodal modificado.
- EDA –Automatización de diseño electrónico
- NA –Análisis nodal
- IDLE –Ambiente de desarrollo integrado.
- Sympy –Librería de cálculo simbólico para Python.
- Lcapy –Librería de análisis simbólico para Python
- TINA –Software de análisis simbólico.
- SCAM –Herramienta simbólica de solución de circuitos en MATLAB.
- Matplotlib –Librería de gráficos en Python.
- VCVS –Fuente de voltaje controlada por voltaje.
- CCVS –Fuente de voltaje controlada por corriente.
- CCCS –Fuente de corriente controlada por corriente.
- VCCS –Fuente de corriente controlada por voltaje.
- Numpy –Librería de análisis numérico pera Python.
- LTSpice –Programa de simulación con énfasis en circuitos integrados.

Índice general

Introducción

ABC fue desarrollado como alternativa a Basic a principios del año 1980, posteriormente Python nace como evolución de ABC de los años ochenta del siglo pasado, desde la versión 1.0 en 1994, 2.0 en el año 2000 y la versión 3.0 en 2008, siendo la 2.7.x y Python 3.x.x las más usadas por su gran número de librerías y de personas interesadas en seguir desarrollando, en los últimos años las versiones 3.x.x son las que tienen mayor soporte, por la comunidad de programadores que continúan desarrollando librerías y nuevos métodos de programación como ahora los conocidos entrenamientos de redes neuronales, que a principios de siglo XX, parecía una gran meta a conseguir tanto por hardware de cómputo como por flexibilidad en software, y técnicas de como hacerlo, hoy en día a dos décadas del siglo XXI, es un placer poder observar como es posible hacer entrenamientos de aprendizaje automático, una maravilla y una fusión de diferentes disciplinas de ciencia e ingeniería.

Esto ha hecho que Python hoy por hoy sea un lenguaje flexible, con tanta información, soporte y gran número de programadores, matemáticos, físicos por mencionar algunas áreas de la investigación, ciencia y tecnología que se han inclinado hacia Python y hacen posible este gran soporte con el cual se cuenta, también Python cuenta con su extensa variedad de librerías como lo son Sympy lanzada en el 2007 y que es una biblioteca o librería para matemáticas simbólicas y que permiten resolver un gran número de cálculos. O como NumPy que desde el año 2006, ha sido de gran ayuda para cálculos numéricos. O Matplotlib que desde el año 2003 ha sido de gran utilidad para plotear gráficos, en Python, algo que en décadas anteriores hubiera sido un sueño, hoy por hoy Python sigue dando mucho.

Por otro lado, en MATLAB mediante un paquete llamado SCAM y EI-SCAM, son librerías para análisis simbólico en circuitos eléctricos bastante bien desarrolladas con resultados muy buenos y de uso académico, por la forma en la que este paquete fue escrito para converger en las soluciones simbólicas, Roldán [4] explica que la primera generación de los analizadores simbólicos apareció a finales de los años 60, pero que estos simuladores no tuvieron mucha difusión por el elevado costo de cómputo y que en los años 70 estos tuvieron una edad de invierno por la aparición de los simuladores numéricos y fue hasta la siguiente década de los 80 que el análisis simbólico fue reconocido por facilitar el conocimiento de circuitos analógicos y proporcionar los modelos matemáticos a circuitos eléctricos.

SCAM para MATLAB es un paquete que se descarga y se corre tomando como carpeta raíz la dirección en donde se encuentre alojado la paquetería de SCAM. Trabaja introduciendo un netlist, en donde se ingresan los componentes electrónicos, con algún valor en resistencia, capacitancia, etc. con extensión *.cir* compatible con herramientas de simulación tipo SPICE, el análisis de SCAM devuelve las soluciones para los elementos introducidos.

Capítulo 1

Estado del arte

El presente capítulo se aborda el estado del arte para el análisis simbólico de circuitos eléctricos destacando diferentes tipos de simuladores (simbólico, semi simbólico, numérico y simbólico simplificado), posteriormente se presentan las diferentes herramientas existentes para el diseño simbólico en la categoría de soluciones comerciales y soluciones académicas, finalmente este capítulo borda el planteamiento del problema, justificación, y objetivos a cubrir.

1.1. Análisis simbólico de circuitos eléctricos

Este tipo de análisis requiere de un mayor costo computacional, y por muchos años el análisis simbólico fue descartado dando paso a los simuladores numéricos, tiempo después los avances tecnológicos en computación y decremento en los costos de las computadoras ha hecho posible retomar el análisis simbólico de circuitos eléctricos, además los cálculos hechos por simuladores simbólicos permiten evaluar análisis Gielen [5]:

- *Simbólico*, donde todos los parámetros del componente y de la frecuencia compleja se mantienen en símbolos.
- *Semi simbólico*, donde algunos de los parámetros del componente son valores numéricos.
- *Numérico*, donde todos los parámetros del elemento son valores numéricos.
- *Análisis simbólico simplificado*, donde solo los términos significativos aparecen al final de la expresión.

Existen diferentes técnicas para formular y resolver los sistemas de ecuaciones, así como también es posible aproximar soluciones simbólicas, dada la complejidad de circuitos, *Mourad* [6] explica que en algunos circuitos en donde la cantidad de ecuaciones, y términos resultantes son numerosos, estos pueden aproximarse reduciendo el número de términos y ecuaciones resultantes simbólicas.
Los circuitos formulados por las diferentes técnicas incluyen usualmente los siguientes elementos:

- Elementos pasivos (resitor, inductor, capacitor)

- Fuentes independientes (voltaje y corriente)
- Amplificadores operacionales

Nuevas versiones de herramientas *EDA* para el cálculo simbólico incluyen fuentes dependientes tales como: VCVS (fuentes de voltaje controladas por voltaje), CCVS (fuentes de voltaje controladas por corriente), VCCS (fuentes de corriente controladas por voltaje) y CCCS (fuentes de corriente controladas por corriente), así como la implementación de transformadores y giradores, algunos dispositivos no lineales como el diodo y el transistor, pueden ser modelados mediante estas fuentes dependientes explica *Mourad* [6], actualmente, existen diferentes herramientas computacionales para análisis simbólico.

Las herramientas simbólicas y numéricas deben verse como herramientas complementarias, no como herramientas de competencia, es importante resaltar que los simuladores numéricos sirven para verificar el rendimiento de circuitos dimensionados previamente, mientras que los simuladores simbólicos sirven para la asistencia en la predicción de circuitos no dimensionados.

Algunas aplicaciones asignadas a los simuladores numéricos, están asociadas con la generación de conocimiento en la operación de circuitos eléctricos. En evaluaciones repetitivas de la fórmula que describe el funcionamiento del circuito, como procedimiento a optimización iterativa o análisis estadístico.

1.2. Herramientas de diseño (EDA) para el análisis simbólico

Las herramientas de diseño asistido por computadora (EDA) para el análisis simbólico, refiere al software que lee un circuito eléctrico, y converge en la solución de tenciones y corrientes como incógnitas del circuito. Existe software de tipo comercial, y de tipo académico.

1.2.1. Soluciones comerciales

Analog INSYDES

Es un toolbox para usarse dentro de *Wolfram Mathematica* para circuitos electrónicos personalizados para aplicaciones industriales, trabaja mediante una especificación por medio de una netlist, **Analog INSYDES** permite construir circuitos lineales y no lineales, modelar sistemas simbólicos, y obtiene fórmulas aproximadas e incluye todos los tipos de fuentes controladas, se puede utilizar para generar modelos de comportamiento reducido de circuitos microelectrónicos [7].

Tina V14

Permite realizar análisis simbólico, de ruido, series de Fourier, diagramas de Nyquist, análisis CA, análisis CD, simulación HDL, simulación de microcontroladores [8]

1.2.2. Soluciones académicas

SapWin

Este programa proporciona varias herramientas para crear el esquemático de un circuito lineal que permite encontrar la función de red en el dominio de Laplace con parámetros simbólicos, de forma descriptiva SapWin permite una captura descriptiva basada en una hoja en blanco donde se dibuja un circuito a través de las herramientas básicas como lo son: copiar, cortar, pegar mover y editar un componente, además de acuerdo con Luchetta [9] tiene todos los componentes lineales activos y pasivos bipolares de dos puertos, incluidos los elementos LRC, fuentes controladas, amplificadores operacionales y modelos equivalentes de pequeña señal de transistores BJT y MOSFET, el cálculo simbólico es ajustable esto quiere decir que cada componente puede aparecer en función de red con su nombre simbólico o con un valor numérico.

También el programa contiene una rutina de aproximación controlada por el usuario, esta rutina permite reducir la expresión simbólica a sus términos más significativos, otra característica es que permite trazar la ganancia de fase, el retardo, la posición de polos y ceros, el escalón y la respuesta al impulso. SapWin opera con tres módulos base, explica Arturo [8]:

- El módulo de entrada con un entorno gráfico **SAPWIN** para dibujar el circuito.
- El módulo que opera mediante la expansión simbólica de Laplace llamado **SAPEC**, el cual es capaz de preservar los términos más significativos mediante una rutina de aproximación que trabaja basándose en la tolerancia del error, algo bastante útil en circuitos de tamaño medio y alto, de modo que los términos menos significativos, son eliminados.
- El módulo gráfico para mostrar la función de transferencia, ganancia de fase, diagrama de polos y ceros, etc.

SLiCAP

Es open source para Python y des continuado en 2021 por sus siglas en inglés (Symbolic Linear Circuit Analysis Program) trabaja por medio de una netlist.
Es flexible en cuanto a que permite una conversión de netlist SPICE jerárquica en una matriz simbólica o numérica explica Sánchez [10], ofrece un análisis de ruido simbólico y numérico, integración de ruido numérico sobre frecuencia, determinación simbólica y numérica de funciones de transferencia y coeficientes polinómicos, análisis numérico de polos y ceros. Lugar geométrico de las raíces.

ahkab

Es una librería que trabaja con NumPy SciPy sympy y tabulate, como librerías de cálculo como librerías de ploteo trabaja con Matplotlib y permite ingresar circuitos a través de una netlist, permite análisis transitorio, AC, periódico, polos y ceros, simbólico, también permite plotear resultados explica Arturo [8].

Circuit Magic

Es un programa de simulación de pago diseñado específicamente para estudiantes, tal como se describe en la documentación oficial de Circuit Magic Inc. [11]. Este software proporciona una plataforma versátil y poderosa para el análisis de circuitos eléctricos que contienen dispositivos de corriente continua y alterna. Con Circuit Magic, los estudiantes tienen la capacidad de explorar y comprender el comportamiento de componentes electrónicos clave, como capacitores, inductores, impedancias, fuentes de voltaje y corriente en corriente continua (CC).

El programa ofrece una interfaz intuitiva y amigable, lo que facilita el proceso de diseño y simulación de circuitos. Los estudiantes pueden construir circuitos virtuales utilizando una amplia gama de componentes disponibles en la biblioteca incorporada. Además, Circuit Magic ofrece herramientas de análisis poderosas que permiten a los estudiantes evaluar el rendimiento de sus diseños y realizar mediciones precisas.

Una de las características destacadas de Circuit Magic es su capacidad para simular circuitos tanto en corriente continua como en corriente alterna. Esto significa que los estudiantes pueden explorar el comportamiento de los circuitos en diferentes condiciones, incluyendo señales sinusoidales y respuestas transitorias. Esta funcionalidad es especialmente valiosa para comprender fenómenos complejos, como la respuesta en frecuencia de un circuito o la respuesta transitoria después de la aplicación de una señal de entrada.

Además, Circuit Magic ofrece opciones de visualización interactiva, lo que permite a los estudiantes observar gráficamente la evolución de las variables eléctricas en el circuito a lo largo del tiempo. Esto facilita la comprensión de conceptos abstractos y ayuda a los estudiantes a identificar patrones y tendencias.

Lcapy

Es una librería de código abierto para resolver circuitos lineales de manera simbólica. Lcapy trabaja por métodos:

- Superposición en DC.
- Análisis AC. por fasores.
- Análisis transitorios con Laplace.

Lcapy evalúa todas sus operaciones algebraicas simbólicas con la librería SymPY, de acuerdo con la documentación oficial [12] permite presentar resultados con unidades coherentes, asegurando una mayor precisión en los cálculos y facilita la interpretación de los resultados en términos de magnitudes físicas.

Permite generar esquemas a partir del netlist, con apariencia de libros de texto, y pueden ser modificados para seguir diferentes convenciones, las modificaciones a los esquemas de los circuitos se hacen desde la netlist, como poner alambres, tierras automáticas, cambiar el tamaño o el color de los elementos.

Las expresiones obtenidas son compatibles con formato LaTeX, es muy útil para la inclusión de documentos técnicos o académicos, facilitando la integración de resultados en informes y presentaciones.

Lcapy permite evaluar análisis numérico de un circuito, siempre y cuando se especifique el valor de los elementos que componen al circuito dentro de la netlist, también permite generar gráficos, para visualizar de una forma adicional los resultados obtenidos.

1.3. SCAM

De acuerdo con la documentación oficial del profesor Erik Cheever [13], permite resolver un conjunto de ecuaciones que representa un circuito de forma simbólica, SCAM es una librería para MATLAB, SCAM permite cargar el circuito mediante una netlist en formato .cir SCAM devuelve una matriz A de nxn una matriz columna x de tamaño n que representa las incógnitas del sistema y una matriz columna z de tamaño n que representa las entradas del sistema, es decir las fuentes independientes que pueden ser de corriente o voltaje, de igual forma devuelve una matriz de ecuaciones y por último un conjunto de ecuaciones para cada nodo.

Opera mediante una técnica llamada stamps que consiste en colocar los elementos que componen al circuito mediante el uso de tablas, este método permite incorporar modelos de ruido de elementos como resistores, MOSFET y BJT, toda la documentación puede ser consultada en el link https://lpsa.swarthmore.edu/Systems/Electrical/mna/MNA6.html. en donde se encuentran una serie de ejemplos, desde como representar cada elemento a un netlist, hasta como resolver circuitos con fuentes dependientes, en el ejemplo 4 Cheever [13] explica que para el circuito planteado en la figura 1.1:

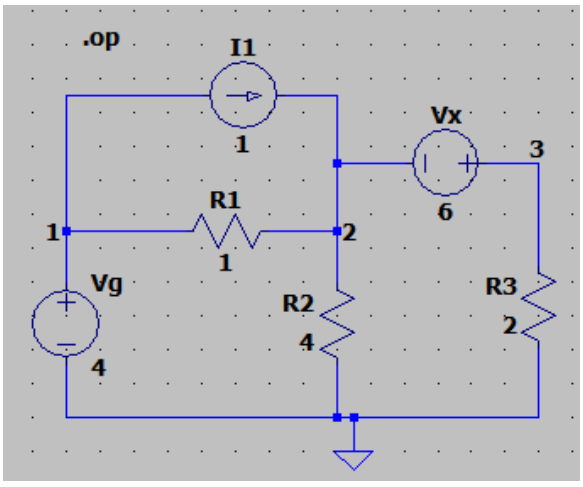


Figura 1.1: Circuito con fuentes independientes

Cheever [13] explica que este circuito tiene tres fuentes independientes, una de corriente y dos de voltaje, de modo que el netlist queda de la siguiente manera:

- Vg 1 0 4
- Vx 3 2 6
- R1 1 2 1
- R2 2 0 4
- R3 3 0 2
- It 1 2 1

De este netlist que representa el circuito 1.1 se observa que todos los elementos son de dos terminales, esto quiere decir que el primer nodo de la netlist siempre será el nodo positivo, mientras que el segundo nodo siempre será el nodo negativo de cada elemento del circuito para el caso de los elementos pasivos (resistores, capacitores e inductores) no importa la polaridad en la que se conecten. SCAM devuelve el vector de soluciones:

$$\begin{bmatrix} V1 \\ V2 \\ V3 \\ Ivg \\ Ivx \end{bmatrix} = \begin{bmatrix} Vg \\ \frac{R2*R3*Vg-R1*R2*Vx+It*R1*R2*R3}{R1*R2+R1*R3+R2*R3} \\ \frac{R3*(R2*Vg+R1*Vx+R2*Vx+It*R1*R2)}{R1*R2+R1*R3+R2*R3} \\ \frac{-(R2*Vg+R3*Vg+R2*Vx+It*R1*R2+It*R1*R3)}{R1*R2+R1*R3+R2*R3} \\ \frac{-(R2*Vg+R1*Vx+R2*Vx+It*R1*R2)}{R1*R2+R1*R3+R2*R3} \end{bmatrix}$$

De acuerdo con los valores del los elementos del circuito, se evalúa:

$$\begin{bmatrix} V1 \\ V2 \\ V3 \\ Ivg \\ Ivx \end{bmatrix} = \begin{bmatrix} 4 \\ 1.142857 \\ 7.142857 \\ -3.857142 \\ -3.571428 \end{bmatrix}$$

1.4. EI-SCAM

Refiere a SCAM aumentado y mejorado, que presenta una mejora en el número de elementos electrónicos [8], ya que a mayor número de elementos, las ecuaciones resultantes aumentan de manera exponencial, por años se ha tenido una no aceptación en la comunidad académica y científica, ya que representa un manejo de elementos considerablemente más pequeño que un simulador numérico, esto porque el método de solución de SCAM y E SCAM limita a resolver circuitos considerablemente grandes, de modo que EI-SCAM propone utilizar:

- Una técnica denominada Diagrama de decisión de determinantes DDD.
- Implementa la solución del sistema matricial mediante la regla de Cramer.

EI-SCAM se desarrolla como proyecto de maestría Arturo [8] en 2016 por la Universidad Politécnica de Puebla en 2016, SCAM no incluía elementos como las fuentes controladas, de acuerdo con el siguiente diagrama muestra una comparación en tiempo de cómputo entre EI

SCAM versus SCAM figura 1.2 en donde se observa EI SCAM tiene un tiempo de cómputo menor a SCAM, por el método de solución propuesto por Arturo [8].

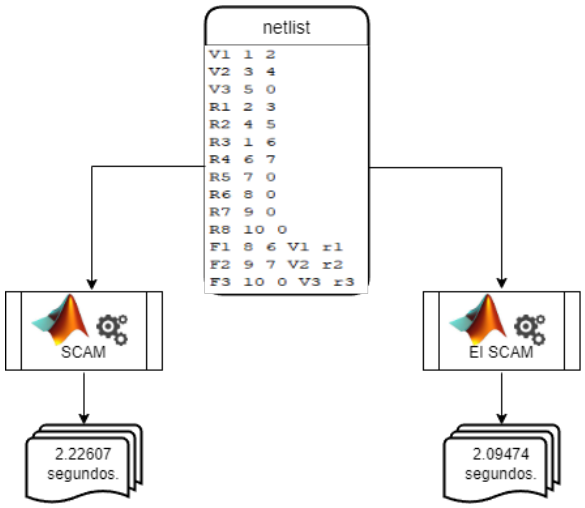


Figura 1.2: SCAM vs EI SCAM

EI-SCAM puede resolver circuitos con elementos pasivos, fuentes controladas, amplificador operacional y transformadores, EI SCAM es una herramienta académica con un entorno amigable para MATLAB, Gaspariano [10] explica la importancia de una herramienta académica en cursos de análisis de circuitos eléctricos, EI SCAM es una herramienta de análisis simbólico que permite a los alumnos una mejor comprensión de análisis en circuitos eléctricos.

1.5. Planteamiento del problema

Es claro que ya existen herramientas que devuelven un cálculo simbólico, entre los diferentes simuladores, haciendo énfasis en SCAM [13] y Lcapy [12], pero se tienen algunos inconvenientes que este trabajo de tesis cubre, para MATLAB se observó un tiempo de respuesta ejecutando scripts con la una netlist de 1 fuente de voltaje y tres resistencias, con un tiempo de cómputo de 1.81 segundos en una CPU Ryzen 7 serie 4000H, esto no es problema en circuitos pequeños, pero en circuitos de mayor complejidad sí que lo es, ahora también se observa un inconveniente y es la licencia de MATLAB, es claro que esta herramienta de tesis está dirigida a un público que puede no contar con una licencia para MATLAB, entonces se puso énfasis en Python, con Python esta tesis asegura dos cosas, la primera es que no se requiere de una licencia, cualquier persona interesada puede leer la documentación de esta herramienta y podrá obtener un análisis simbólico, y dos es un tiempo de ejecución rápido comparándolo con otros métodos y comparándolo con otros lenguajes.

1.6. Justificación

Así como en la industria de diseño electrónico, las herramientas EDA también son útiles en la educación [14], tanto entre pregrado como posgrado, especialmente en los planes de

estudio de ingenierías ligadas a la tecnología, como es el caso de disciplinas como Mecatrónica, Telecomunicaciones, Energía renovable y sistemas automotrices, donde la capacidad de simular sistemas electrónicos y emitir juicios basados en la simulación es una habilidad de importancia crítica. Los programas de simulación numérica como SPICE, en sus diferentes versiones, son muy útiles en actividades de enseñanza de temas relacionados con el análisis de circuitos, sin embargo, la simulación numérica no muestra explícitamente la influencia de cada elemento del circuito en el resultado de la simulación y, por lo tanto, generalmente se requieren varias simulaciones para verificar la influencia de cada elemento del circuito. En ese sentido, se prefiere el análisis simbólico ya que proporciona expresiones analíticas que permiten una visión más profunda del comportamiento del circuito. Además, algunas herramientas de software para el análisis de sistemas son preferidas tanto en la industria como en la educación, tal es el caso de MATLAB. Un ejemplo de la herramienta EDA es SCAM (acrónimo de Symbolic Circuit Analysis in MATLAB) que está disponible para el análisis simbólico de circuitos eléctricos. Sin embargo, las capacidades de SCAM están limitadas al uso de los elementos más básicos de circuitos. En su versión original, SCAM no puede resolver circuitos con un número relativamente alto de nodos. Con la realización de esta Tesis se plantea la realización de una herramienta en Python de análisis simbólico de circuitos mejorada, que incluya otros elementos de circuito, tales como los cuatro tipos de fuentes controladas, es decir, la fuente de voltaje controlada por voltaje (VCVS en inglés), la fuente de corriente controlada por voltaje (VCCS en inglés), la fuente de voltaje controlada por corriente (CCVS en inglés) y la fuente de corriente controlada por corriente (CCCS en inglés), que permita Modelar circuitos activos como amplificadores. De este modo, la herramienta de análisis simbólico propuesta permitiría resolver circuitos muy grandes en poco tiempo y mediante el uso de un software libre.

1.7. Objetivos

Objetivo General:
Desarrollar una herramienta de cómputo para el análisis simbólico de circuitos eléctricos en Python.
Objetivos Específicos:

- **O.E.1**- Estudiar el estado del arte de los programas de análisis simbólico de circuitos.
- **O.E.2**- Implementar un algoritmo de parseo en Python que traduzca una lista de descripción de circuito (*netlist*) a un sistema de ecuaciones lineales.
- **O.E.3**- Implementar un algoritmo en Python que resuelva el sistema de ecuaciones lineales entregado por el parseo mediante diagramas de decisión de determinantes para reducir el tiempo de cómputo de la solución simbólica.
- **O.E.4**- Realizar las pruebas de rendimiento (benchmarking) de la herramienta de análisis propuesta y análizar los resultados obtenidos al compararlos con los entregados por otras soluciones, por ejemplo SCAM.

1.8. Conclusiones

El capitulo uno cubre el panorama de simuladores simbólicos, perite observar los diferentes simuladores existentes y como es que estos devuelven expresiones matemáticas en comparación de los simuladores numéricos donde los valores de salida son resultados numéricos, permitiendo ubicar el trabajo futuro retomando algunas funcionalidades y características a integrar como lo son:

- Un entorno intuitivo.
- Una librería de licencia libre
- La posibilidad de bajar el costo computacional.
- La posibilidad de tener un gran numero de personas que usen esta librería.

Permite comprender algunos aspectos iniciales como el uso de netlist, y los elementos con los que se compromete a simular esta tesis.

Capítulo 2

Análisis simbólico de circuitos eléctricos

El presente capítulo aborda una comparativa, discusión y justificación del porqué se decidió usar el lenguaje de programación en Python frente a MATLAB, posteriormente se aborda el funcionamiento de algunas herramientas existentes y la diferencia entre trabajar en el entorno de Python con el entorno de MATLAB en específico con Lcapy y SCAM, donde se visualiza la formulación matemática y la manera en la que se introduce un circuito a estas librerías, esta comparativa permite presentar una discusión sobre las áreas de oportunidad para Python frente a MATLAB.

Con un punto de partida sólido, justificado y bien definidos se presenta la formulación matemática a utilizar abordando la teoría MNA (Análisis Nodal Modificado), y se presentan ejemplos de este método para cada elemento, hasta este capítulo se desarrolla un algoritmo de parseo con más de dos mil líneas de código (robusto) que permite hacer el llenado de la matriz MNA y el sistema de ecuaciones en general, este método se encuentra disponible en los enlaces del apéndice A de esta presente Tesis.

2.1. Python *vs* MATLAB

Antes de obtener resultados y seguir abordando el tema, es importante detallar una comparación entre Python y MATLAB.
SCAM y EI-SCAM corren en MATLAB, Lcapy y SCAPY corren en Python, es por eso que nace la necesidad de hacer una comparación entre MATLAB y Python.

MATLAB es un software empleado en la investigación y en la industria, es un lenguaje de alto nivel de cuarta generación. MATLAB es un software que interpreta el código mientras el programa es ejecutado, MATLAB interpreta las líneas de código línea a línea y traduce las instrucciones a nivel máquina en el momento, MATLAB no compila sus instrucciones.

Un software que funciona por medio de un intérprete de instrucciones disminuye la velocidad de ejecución al no tener estas instrucciones compiladas; sin embargo, esta misma característica permite una memoria dinámica y sesiones interactivas, una gran ventaja de tener un intérprete

es que las instrucciones son más cortas que los programas equivalentes que requieren ser compilados.

Todos los lenguajes que traducen sus líneas de código sacrifican velocidad, pero ganan tiempo en desarrollo, es por eso que MATLAB es muy usado en aplicaciones científicas y de ingeniería, por el tiempo de desarrollo. Una característica importante de MATLAB es que se pueden escribir líneas de código en el prompt y después regresar a esas líneas que fueron anteriormente escritas en el prompt para después ser nuevamente ejecutadas al instante [15].

Python es un lenguaje de alto nivel que trabaja con muchas ventajas similares a MATLAB. Es un lenguaje de programación que trabaja de igual manera con un intérprete, teniendo todas las ventajas de ser dinámico, con una sintaxis que ahorra mucho tiempo en el desarrollo, de igual manera tiene un prompt que ejecuta instrucciones y que de igual manera que MATLAB tiene una memoria magnética que permite recuperar las instrucciones pasadas y ejecutarlas al instante, Python tiene ventajas sobre MATLAB en el contexto de la enseñanza [16]:

- Tiene una sintaxis limpia e intuitiva, permite importar librerías.
- Tiene un bloque de instrucciones que proporciona la mayoría de funciones que necesita un principiante.
- Puede ser usado en su totalidad como una herramienta orientada a objetos.
- El intérprete de Python es software libre.
- Python puede correr en multiplataforma (Windows, Mac OS, Linux, Unix) pero también en plataformas con hardware limitado como Raspbian o Android.

Python es un lenguaje estable y que actualmente algunas organizaciones como Google, NASA, Intel o YouTube corren muchas de sus instrucciones en Python [16].
Algunos artículos muestran a Python más favorable que MATLAB; sin embargo, algunos otros muestran a MATLAB más favorable que Python.

Algo que sin duda se concluye es que si se pueden comparar algunas características entre un entorno y otro, como la velocidad de ejecución, o el tiempo ahorrado en desarrollo, o la fácil implementación entre uno y otro, pero que al final Python y MATLAB son herramientas con aplicativos diferentes, por ejemplo al programador le puede resultar encontrar errores más fácilmente en Python principalmente en cuanto a programas extensos de miles de líneas de código se refiere en comparación con MATLAB, pero que en aplicaciones científicas es entendible que se prefiera MATLAB, especialmente en cuanto a métodos numéricos se refiera el tema de interés.

Sin embargo, una característica importante a resaltar es la movilidad que Python puede tener frente a MATLAB, dicha característica es la movilidad multiplataforma, una de las proyecciones de SCAPY es la aplicación académica y Python permite que SCAPY pueda llegar a muchos estudiantes, muchas personas que quieren aprender circuitos eléctricos, muchas personas que quieren implementar un circuito eléctrico para acoplar alguna señal en algún proyecto

y requieran una predicción del cálculo simbólico.

Python puede ser fácilmente instalado en muchas plataformas, no requiere de múltiples gigas disponibles para ser instalado, y cuando se abre el entorno no toma tanto tiempo en cargar toda la interface, y que incluso teniendo el prompt no requiere más que dar clic al archivo *.py* para correr el script.

2.2. Herramientas disponibles

2.2.1. CircuitNAV

CircuitNAV (Circuit Network Analysis and Verification) Es una herramienta online que obtiene análisis simbólico de circuitos eléctricos, se utiliza para la verificación formal de circuitos digitales. La herramienta utiliza técnicas de análisis simbólico para probar la corrección de los circuitos digitales en términos de su comportamiento lógico.
utiliza una variedad de técnicas de análisis simbólico para realizar su análisis, incluyendo el modelado simbólico de circuitos y la construcción de fórmulas lógicas para representar el comportamiento del circuito en términos de sus entradas y salidas.

A su vez, esta herramienta permite cargar una netlist que puede ser exportada des LTSpice por ejemplo, permitiendo cargar por medio del navegador web el archivo en formato *.cir* y finalmente eligiendo un nodo de interés se puede saber el comportamiento simbólico en dicho nodo [17]

2.2.2. Lcapy

Lcapy librería de Python ofrece análisis de circuitos en forma simbólica y numérica mediante las siguientes técnicas:

- Análisis de mallas
- Análisis nodal NA
- Análisis nodal modificado MNA
- Análisis en espacio de estados SS

como análisis principal según la documentación de Lcapy [12] en donde para el análisis nodal devuelve la matriz **A** y el vector **b** donde $\mathbf{Ab} = \mathbf{b}$ como lo plantea la ecuación 8. Para el análisis MNA devuelve las matrices **A**, **B**, **C**, **D**, **G** y los vectores **I** y **E** como lo describe la ecuación 1, finalmente para el método de análisis por espacio de estados devuelve las matrices **A**, **B**, **C**, **D** y los vectores **E** e **I**
Mientras que de acuerdo con la documentación de SCAM el método empleado es MNA.
A continuación se muestra una comparación en el análisis de un circuito divisor de voltaje en *Lcapy* con espacio de estados y $SCAM$ con el método de MNA

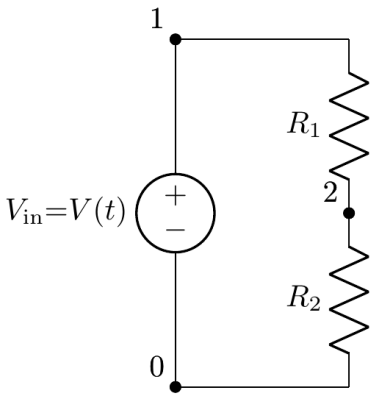


Figura 2.1: Test divisor de tensión

Haciendo un test en Lcapy:

```
In [75]: from lcapy import Circuit
         from lcapy import s
         a = Circuit("""
         Vin 1 0 {V(t)}; down
         W 1 1_0
         R1 1_0 2; down
         W 0 1_2
         R2 2 1_2; down
         """)
         ss = a.ss

In [76]: ss.u
Out[76]: [V(t)]

In [77]: ss.y
Out[77]: [ v1(t)
           v2(t)
           iVin(t)
           iR1(t)
           iR2(t) ]

In [78]: ss.output_equations()
Out[78]: [ v1(t)   [ ]          [ 1
           v2(t)   [ ]          [ R2/(R1+R2)
           iVin(t) [ ]  = []  + [ 1/(R1+R2)   [V(t)]
           iR1(t)  [ ]          [ 1/(R1+R2)
           iR2(t) ]             [ 1/(R1+R2) ]
```

Figura 2.2: Código para obtención ecuaciones simbólicas Lcapy

De la figura 2.2 se observa que el nodo de interés es el nodo 2, correspondiente a $v_2(t)$ el cual es una expresión simbólica correcta, de igual forma SCAM devuelve un conjunto de soluciones, que para el nodo 2 corresponde v_2 de la figura 2.4, la expresión $I_v in$ del conjunto de soluciones

de la figura 2.4, corresponde con la expresión $i_{vin}(t)$ de la figura 2.2, la cual es diferente en el signo, lo cual ya no es del todo confiable el análisis devuelto por Lcapy.

Haciendo un test en MATLAB:

```
64  %% SCAM vs Lcapy
65  clc; clear all;
66  fname="Lcapy1DivisorT.cir";
67  scam
68
69
70
71
72
73
```

Figura 2.3: Código para obtención ecuaciones simbólicas Matlab

```
Netlist:
Vin 1 0
R1 1 2
R2 2 0

The A matrix:
[ 1/R1,       -1/R1, 1]
[-1/R1, 1/R1 + 1/R2, 0]
[    1,           0, 0]


The x matrix:
  v_1
  v_2
I_Vin


The z matrix:
  0
  0
Vin


The matrix equation:
   I_Vin + v_1/R1 - v_2/R1 == 0
v_2*(1/R1 + 1/R2) - v_1/R1 == 0
                     v_1 == Vin


The solution:
                 v_1 == Vin
v_2 == (R2*Vin)/(R1 + R2)
  I_Vin == -Vin/(R1 + R2)
```

Figura 2.4: Resultados simbólicos de MATLAB

2.2.3. Área de oportunidad para una nueva herramienta de análisis simbólico de circuitos eléctricos en Python

Como pudo verse, existen herramientas que permiten obtener expresiones simbólicas, estas pueden ser comerciales o académicas, una solución académica es $SCAM\ EISCAM$, que a pesar de no tener un costo por usar estas librerías que permiten computar análisis simbólico, tienen la característica de correr en MATLAB el cual requiere de una licencia, es verdad que una licencia de MATLAB no es inconveniente para institutos académicos meramente grandes, ni para alumnos de estos institutos que pagan por el uso de la licencia, pero ¿Qué pasa con los alumnos que pertenecen a institutos o escuelas en donde no se cuenta con una licencia de MATLAB?, Respondiendo a esta pregunta es un hecho que en Latinoamérica existen una gran variedad de institutos de nivel medio superior y nivel superior que no cuentan con una licencia de MATLAB para sus estudiantes, además de que MATLAB requiere de un mínimo de hardware, de acuerdo con el sitio oficial https://la.mathworks.com/support/requirements/matlab-system-requirements.html para la versión $R2023a$ el hardware mínimo requerido es proporcionado en el apéndice B figura: 5.11

Por otro lado, se tienen alternativas en Python, como por ejemplo *Lcapy*, que aunque no requiere de una licencia, se tiene el inconveniente de no contar con la facilidad de trabajar con el archivo *.cir* directamente, que aunque en principio no es problema, es importante señalar que la manera de declarar la *netlist* es un poco diferente a la manera en la que se obtendría de un simulador numérico como *LTSpice*, por otro lado en las pruebas que se hicieron se observó una inconsistencia en los resultados simbólicos obtenidos en la sección 2.2.2, con lo cual no hay duda que se corregirán en un futuro, pero que por el momento no permite obtener resultados del todo confiables

Esto permite dar una oportunidad a escribir una biblioteca que permita obtener expresiones simbólicas en circuitos, eléctricos, que no requiera de una licencia, que no se tenga que inconveniente de requerir tanto espacio de almacenamiento para instalar el software que permita usar esta biblioteca, y lo más importante, que permita obtener expresiones simbólicas confiables

2.3. SCAPy

2.3.1. Análisis Nodal Modificado (MNA)

Se aborda el análisis nodal modificado por A. Roldán [4] en donde se aborda sobre los diferentes simuladores como:

- *Analog Insydes.*
- *SAPWIN.*
- *Circuit Magic.*
- *XFUNC*22.
- *SCAM.*

Que permiten el análisis de circuitos eléctricos por medio del método MNA para la expresión simbólica de expresiones de tensión, eléctrica y corriente en circuitos electrónicos, el análisis MNA permite describir un circuito lineal como un sistema de ecuaciones lineales para n nodos y m fuentes. [4]

$$Ax = z \rightarrow \begin{bmatrix} G & B \\ C & D \end{bmatrix} \begin{bmatrix} v \\ j \end{bmatrix} = \begin{bmatrix} i \\ e \end{bmatrix} \tag{2.1}$$

Donde:

$$A = \begin{bmatrix} G & B \\ C & D \end{bmatrix}, \quad x = \begin{bmatrix} v \\ j \end{bmatrix}, \quad z = \begin{bmatrix} i \\ e \end{bmatrix}$$

De modo que:

- **G** es la matriz de conductancias de tamaño $n \times n$. Sus elementos se obtienen de las derivadas de las ecuaciones de $Kirchhoff$ de tensión con respecto a las tensiones nodales
- **B** es una matriz de tamaño $n \times m$ donde m es el número de nuevas variables (corrientes) que aparecen al introducir los amplificadores operacionales y generadores de tensión tanto dependientes como independientes. Sus elementos se obtienen de las derivadas de las ecuaciones de $Kirchhoff$ de tensión con respecto a las nuevas variables.
- **C** matriz de tamaño $m \times n$ cuyos elementos se obtienen a partir de las derivadas de las nuevas ecuaciones introducidas con respecto a las tensiones nodales.
- **D** matriz de tamaño m×m cuyos elementos contienen las derivadas de las nuevas ecuaciones respecto de las nuevas variables.
- **v** es un vector de n tensiones nodales.
- **j** es un vector de m nuevas corrientes de malla introducidas.
- **i** es el vector de las corrientes de excitación. El elemento k-ésimo de i, ik, es la suma de todas las corrientes independientes entrantes al nodo k.
- **e** es el vector de las fuentes de tensión de malla.

Algo muy dicho con anterioridad es que el tiempo de cálculo dependerá del tamaño de elementos en la netlist, y que al ser análisis simbólico, consumirá mayor recurso de Hardware en comparación con un análisis numérico.
A continuación se muestra un diagrama de flujo que permite observar la obtención de la generación de resultados simbólicos.

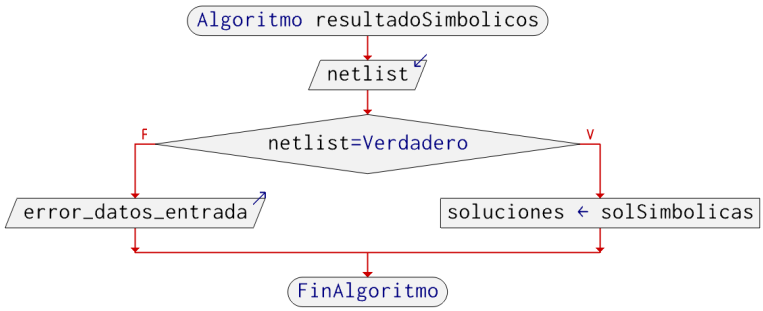


Figura 2.5: Generación de resultados simbólicos

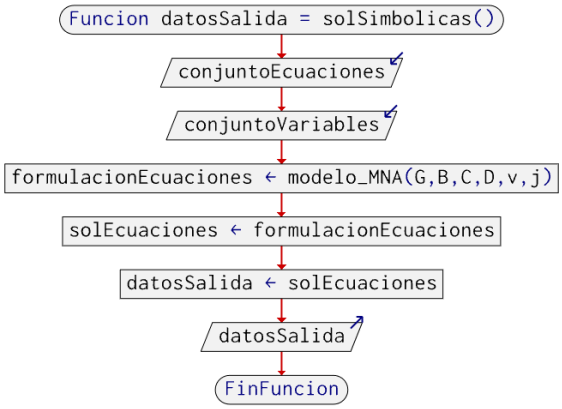


Figura 2.6: Generación de soluciones simbólicas

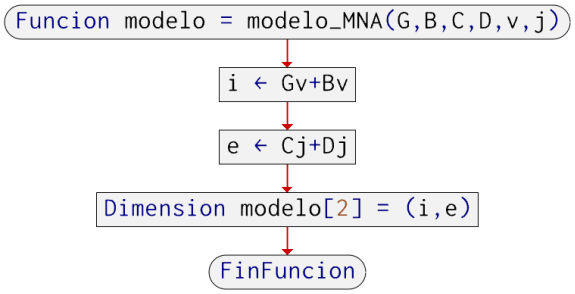


Figura 2.7: Modelo simbólico

El MNA permite computar tanto análisis de corriente y nodos, explica Mourad [6] en donde

profundiza acerca del análisis MNA sin antes hacer la introducción acerca del análisis NA nodal clásico, para posteriormente presentar tres modificaciones del análisis MNA, de modo que el NA explica Mourad [6] dependiendo del circuito, en donde para:

- circuitos que emplean dispositivos arbitrarios de dos terminales con conductancias bien definidas.
- Dispositivos multidimensionales arbitrarios que tienen la llamada conductancia o matrices de admitancia (modelos de transistores de pequeña señal, descritos por parámetros y, fuentes de corriente por voltaje, etc.).

el análisis por NA cubre perfectamente, haciendo frente a circuitos con estas características mencionadas.
Mientras que para circuitos que: Carecen de conductancia o admitancia bien definida (amplificadores operacionales OpAmps, transportadores, transformadores y otros componentes modernos). Deberá ser analizado por MNA explica Mourad [6] sea el análisis NA se basa en el siguiente procedimiento:

- Uno de los nodos se establece como nodos de referencia, el número 0 en simulación por computadora es un nodo de referencia para definir los voltajes de todos los nodos restantes. A los que se les llamará voltajes nodales, formando el conjunto de variables desconocidas de este método y a los nodos se les llamará nodos independientes, es recomendable dirigir todos los voltajes hacia el nodo de referencia. En donde los voltajes son las únicas variables desconocidas.
- LCK ley de corriente de $Kirchhoff$ se aplica a cada nodo que no sea de referencia de modo que: La suma de corrientes que ingresan al nodo desde fuentes de corriente externas es igual a la suma de corrientes que sales del nodo a través de las ramas del circuito.
- Se calculan los voltajes nodales. se deberán derivar las cantidades del lado derecho del punto 2 con ayuda de la ley de Ohm, como productos de admitancias y conductancias de rama y voltajes, los voltajes nodales aparecen del lado derecho de las ecuaciones.

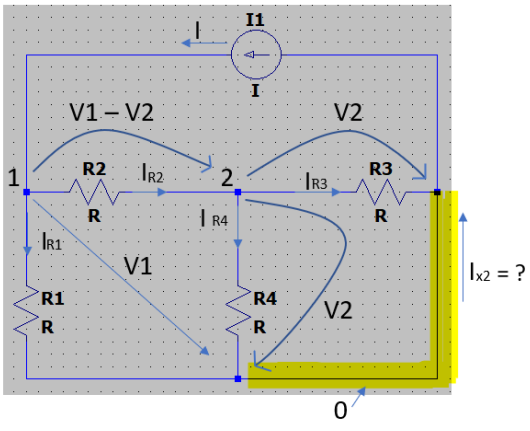


Figura 2.8: circuito de ejemplo para análisis NA

En un circuito primero se enumeran los nodos y el nodo de referencia es enumerado por el número cero, entonces observando que la resistencia R3 está conectada con la fuente de corriente, después son señalados los voltajes nodales V1 y V2 en, que forman un conjunto de 2 incógnitas y se deben construir dos ecuaciones para su cálculo, entonces las ecuaciones de LCK para los nodos 1 y 2. Para calcular I_{x2}, solo se necesita calcular el voltaje nodal V2 porque IR_3 puede resultar de V_2 y luego I_{x2} se puede encontrar como una diferencia entre I e IR_3 Entonces las ecuaciones quedan de la siguiente manera.

$$I = I_{R1} + I_{R2} \tag{2.2}$$

$$0 = -I_{R2} + I_{R3} + I_{R4} \tag{2.3}$$

Las ramas se pueden dirigir arbitrariamente. El valor computado diferirá por la arbitrariedad, se deberán derivar las corrientes antes de construir las ecuaciones del circuito. Para esto se usa G símbolos con los sufijos apropiados de modo que:

$$I = G_1V_1 + G_2(V_1 - V_2) \tag{2.4}$$

$$0 = -G_2(V_1 - V_2) + G_3V_2 + G_4V_2 \tag{2.5}$$

Arreglando las ecuaciones de la forma:

$$I = (G_1 + G_2)V_1 - G_2V_2 \tag{2.6}$$

$$0 = -G_2V_1 + (G_2 + G_3 + G_4)V_2 \tag{2.7}$$

Entonces el sistema queda de la siguiente forma:

$$\begin{bmatrix} I \\ Null \end{bmatrix} = \begin{bmatrix} G_1 + G_2 & -G_2 \\ -G_2 & G_2 + G_3 + G_4 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix} \tag{2.8}$$

Por otro lado, el análisis nodal modificado MNA a diferencia del NA que no permite circuitos que carecen de la matriz de admitancia, el primer método de análisis MNA es **el método de stamp** este análisis lo permite teniendo M ecuaciones adicionales con su respectivo número de ecuaciones desconocidas, por lo que la ecuación matricial tendrá una estructura especial, en donde se dice que la matriz de admitancia original se ampliará debido a que algunas de las ecuaciones de LCK se modifican, llamando a esta matriz de *pseudoadmitancia*, la cual contiene los llamados sellos de la matriz, este método es computacionalmente caro por el número de ecuaciones que si bien en análisis numérico no es un problema para la máquina, cuando se trata de análisis simbólico si lo es, considerando un circuito descrito por ecuaciones de NA clásica, de dos terminales descrito por el modelo de Thevenin que se encuentra entre los nodos a y b, retomando la idea de ampliar la matriz de admitancia, dado el circuito.

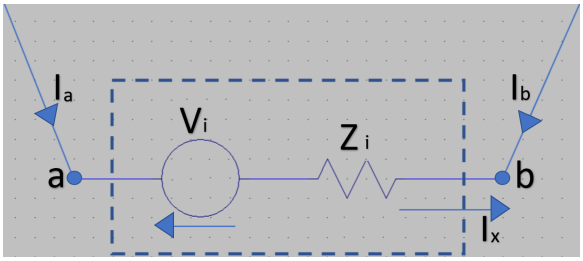


Figura 2.9: circuito de ejemplo para análisis MNA

La ecuación que describe el nodo a será completada por el lado derecho por una corriente I_x que sale del nodo, y para el nodo b por una corriente I_x con signo negativo, y los voltajes en los nodos están vinculados mediante:

$$Z_i I_x + V_b = V_i + V_a \tag{2.9}$$

$$V_i = Z_i I_x + V_b - V_a \tag{2.10}$$

De este modo el sistema de ecuaciones MNA será:

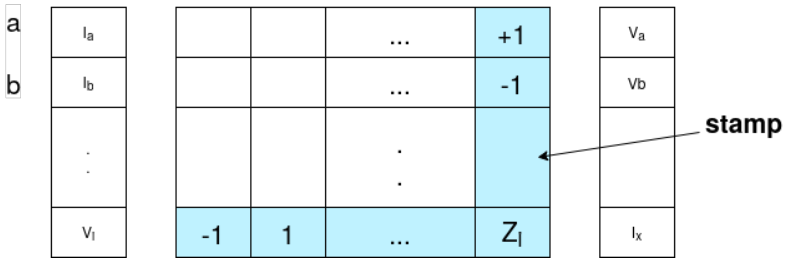


Figura 2.10: Formulación del análisis MNA a partir de NA

La corriente I_x es el resultado de que el vector de voltajes nodales sea extendido por otra cantidad. El número de ecuaciones se incrementa debido al acoplamiento de voltajes nodales V_a y V_b, El voltaje V_i es añadido en la parte inferior del vector izquierdo, la modificación de las ecuaciones de LCK, se logra mediante $+1$ y -1, las fuentes de voltaje independientes Z_i también pueden ser cero, entonces la fuente de voltaje será ideal, cuando $V_i = 0$ entonces $Z_i = 0$ este enfoque se puede usar para el análisis de circuitos que tienen fuentes controladas por corriente.

Se describe el planteamiento matemático MNA (Análisis Nodal Modificado), de elementos pasivos RLC (resistencias, capacitores e inductores).
De igual manera se presentan los diagramas de flujo que describen el funcionamiento de del llenado de la matriz MNA, así como los vectores x y z

Del circuito figura 2.11:

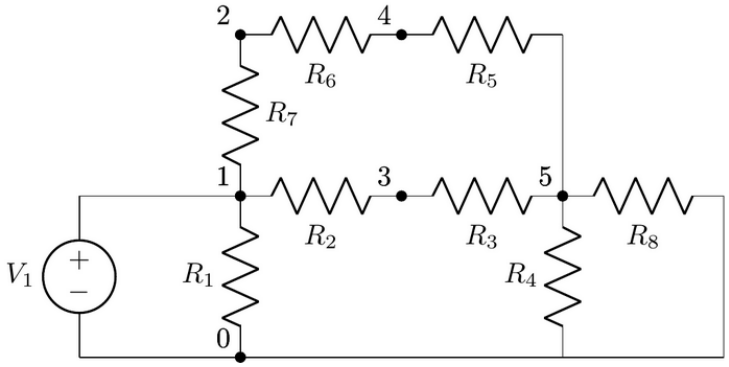


Figura 2.11: Arreglo de resistencias

Como se observa se enumeran los nodos de forma arbitraria, designando el nodo cero para la tierra.

De las leyes de corrientes de Kirchhoff, se obtienen las ecuaciones para el circuito.
(nota: G = conductancia en el resistor = $\frac{1}{R}$):

$$G1 * V1 + G2V1 + G7V1 - G7V2 - G2V3 = I_{v1}$$

$$G7V2 + G6V2 - G7V1 - G6V4 = 0$$

$$G2V3 + G3V5 - G2V1 - G3V5 = 0$$

$$G6V4 + G5V4 - G6V2 - G5V5 = 0$$

$$G3V5 + G4V5 + G8V5 + G5V5 - G3V3 - G5V4 = 0$$

De modo que de acuerdo a la metodología de análisis por nodos modificada (MNA) puede ser expresado como:

$$A = \begin{bmatrix} G1+G2+G7 & -G7 & -G2 & 0 & 0 & 1 \\ -G7 & G6+G7 & 0 & -G6 & 0 & 0 \\ -G2 & 0 & G2+G3 & 0 & -G3 & 0 \\ 0 & -G6 & 0 & G5+G6 & -G5 & 0 \\ 0 & 0 & -G3 & -G5 & G3+G4+G5+G8 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$x = \begin{bmatrix} v1 \\ v2 \\ v3 \\ v4 \\ v5 \\ I_{v1} \end{bmatrix}$$

$$z = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ V_1 \end{bmatrix}$$

En donde el vector z son las entradas al sistema de ecuaciones y x es el vector de incógnitas que se desea conocer.
de acuerdo con el modelo [4]:

$$A = \begin{bmatrix} G & B \\ C & D \end{bmatrix}, \quad x = \begin{bmatrix} v \\ j \end{bmatrix}, \quad z = \begin{bmatrix} i \\ e \end{bmatrix}$$

2.3.2. Parser de la lista de descripción de un circuito mediante el Método de los STAMPS

Cuando se habla de parseo referente a esta tesis, de acuerdo con Tanaka [18], refiere al proceso de analizar una lista de símbolos eléctricos, que se encuentran interconectados y alojados en una lista entre símbolos, a esta lista se le conoce como Netlist, los símbolos de esta Netlist se interpretan y analizan mediante un método llamado STAMPS.
Se sabe que las herramientas EDA (Electronic Design Automation) así como las herramientas SPICE (Simulation Program with Integrated Circuit Emphasis) trabajan con extensiones: *.cir* de este modo la librería SCAM (Symbolic Circuit Analisys in Matlab) [13] es compatible para esta exención.
Siguiendo esta forma de trabajo, en la que los circuitos puedan ser empaquetados en formato *.cir* y ser leídos por SCAPY (Symbolic Circuit Analisys in Python).
Esta es la razón por la que se decide trabajar con Netlist en formato *.cir*.
Para ilustrar el método de parseo en esta tesis se toma el circuito de la figura 2.12:

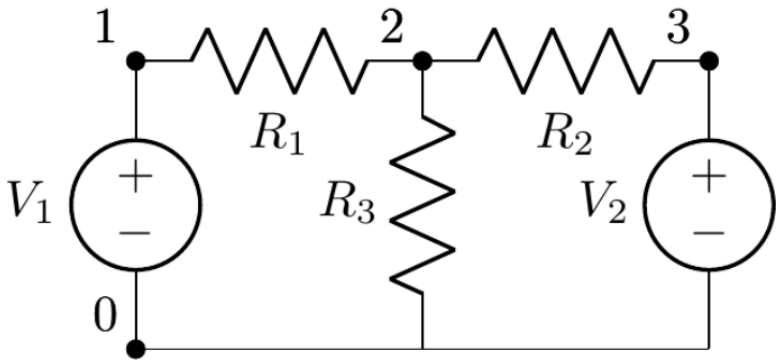


Figura 2.12: Circuito paral ejemplificar

Para el circuito de la figura 2.12 se empieza por enumerar los nodos partiendo por el nodo cero como nodo tierra, posteriormente se pasa este circuito a una lista descriptiva de elementos con el número de nodos de cada elemento, esta lista descriptiva es conocida como netlist la cual quedaría de la siguiente manera:

- V1 1 0
- R1 1 2
- R2 2 3
- R3 2 0
- V2 3 0

Para el algoritmo, se trabaja con pandas de Python y Numpy (librerías públicas de Python), Pandas y Numpy fueron descritas en el inicio de esta tesis, como librerías de:

- Pandas: Librería para formular tablas (para esta tesis)

- Numpy: Librería para crear arreglos vectoriales de mxn

A continuación se presenta el diagrama de flujo figura 2.13 para la creación de una función llamada *read_cir* que permitirá leer el netlist *.cir* que puede ser creado o editado desde *Ltspice*:

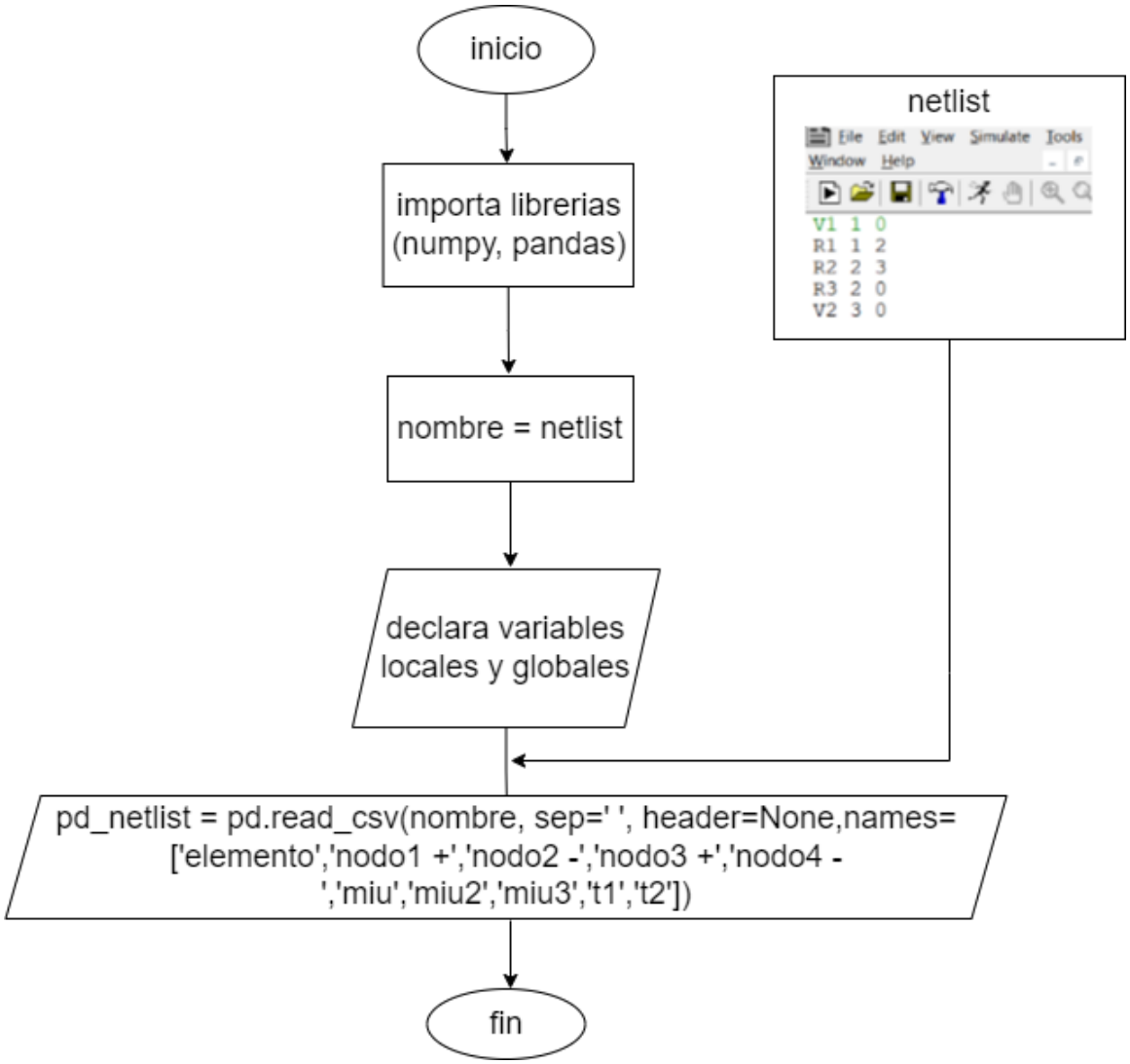


Figura 2.13: Netlist

Resultados al mandar a traer la función: *read_cir* como se muestra en la figura 2.14

```
>>>
= RESTART: C:\Users\luico\
 corrientes.py
  elemento  nodo1  nodo2
0       V1      1      0
1       R1      1      2
2       R2      2      3
3       R3      2      0
4       V2      3      0
```

Figura 2.14: netlist

La figura 2.14 muestra el netlist en la terminal de Python en un formato de tabla dinámica generado por Pandas, la idea de trabajar de esta manera es poder tener un campo de visualización del circuito en el netlist. En esta parte del código también es importante mencionar que se dejan preparados todos los vectores como:

- elemento
- nodo 1
- nodo 2

Con la finalidad de trabajar con arreglos por numpy.
A continuación se muestra una tabla con los STAMPS de cada elemento.

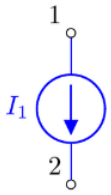
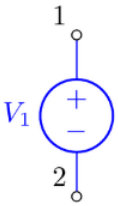
Elemento	Simbolo	Matriz y o vector
Fuente de corriente	 In n1 n2	1 -I1 2 I1
Fuente de voltaje	 Vn n1 n2	V1 V2 I 1 1 2 -1 m+1 1 -1 E

Figura 2.15: Método de los STAMPS (1)

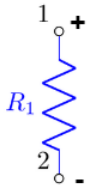
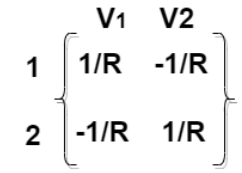
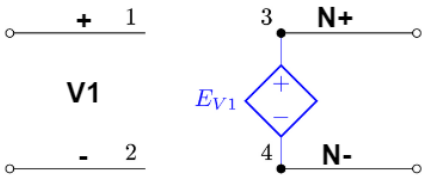
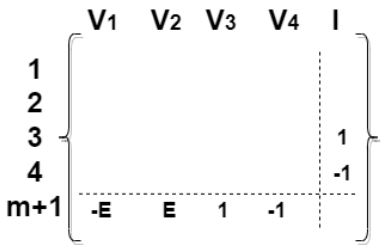
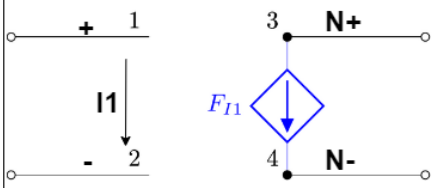
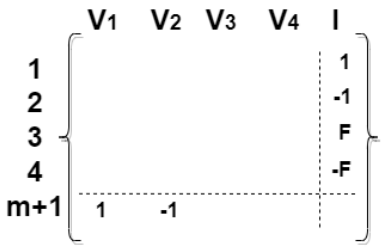
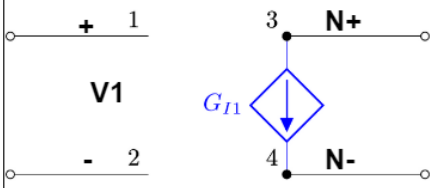
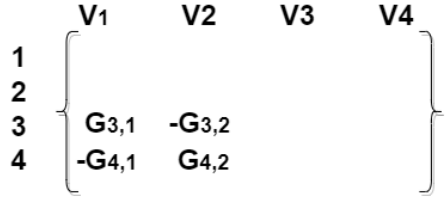
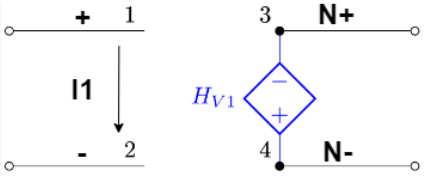
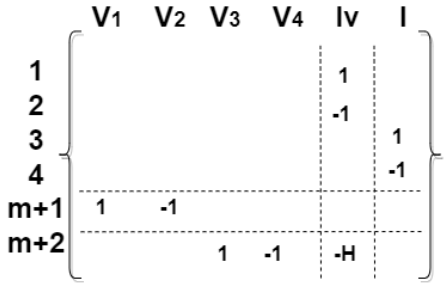
Resistencia	 **Rn n1 n2**	
(VCVS) fuente de voltaje controlada por voltaje		
(CCCS) fuente de corriente controlada por corriente		
(VCCS) fuente de corriente controlada por voltaje		
(CCVS) fuente de voltaje controlada por corriente		

Figura 2.16: Método de los STAMPS (2)

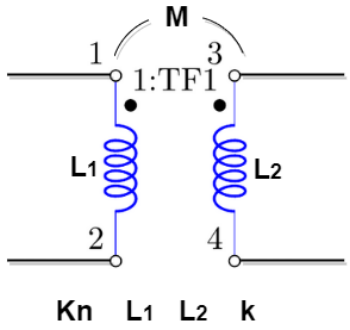
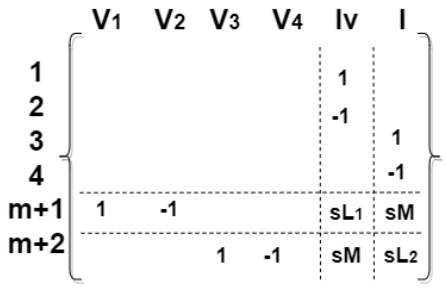
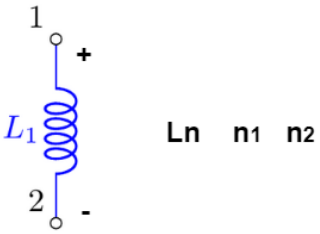
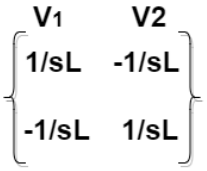
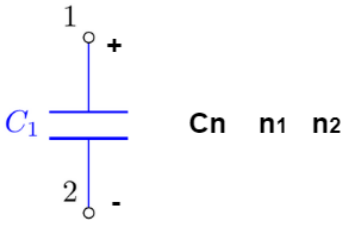
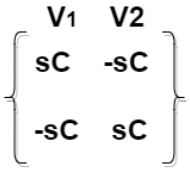
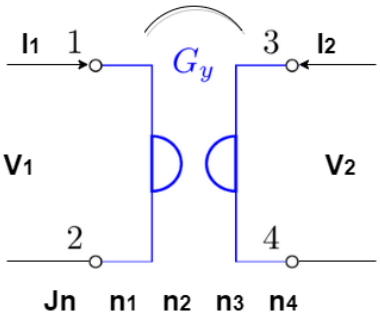
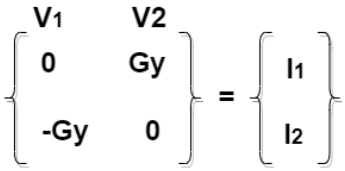
Transforma-dor		
Inductor		
Capacitor		
Girador		

Figura 2.17: Método de los STAMPS (3)

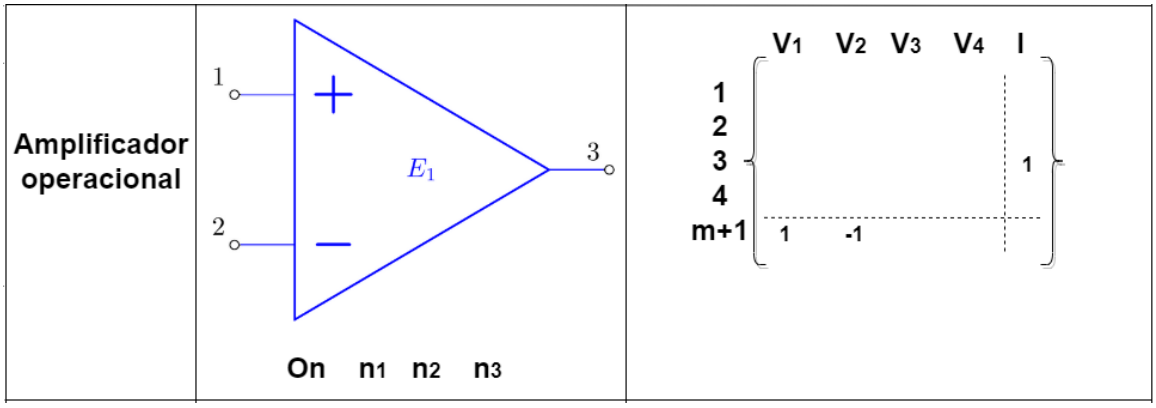


Figura 2.18: Método de los STAMPS (4)

2.3.3. Formulación del sistema de ecuaciones del circuito bajo análisis a partir del (MNA)

Es importante observar que en la subsección 2.3.1 se explicaron algunos ejemplos. Sin embargo, estos tratan a los elementos resistivos como la impedancia G que es igual a: $\frac{1}{R_n}$. En esta subsección del presente capítulo, se planteará una formulación formal de diferentes circuitos a partir de la formulación MNA.

Circuito con fuentes independientes

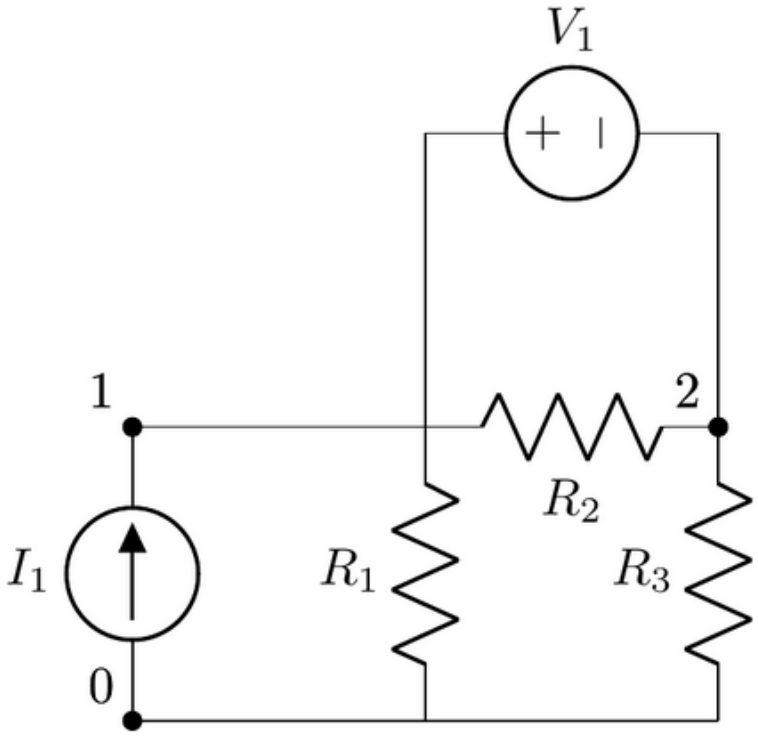


Figura 2.19: Circuito de ejemplo 2

Para el presente circuito se analizan los nodos 1 y 2

$$\frac{1}{R_2}V_1 + \frac{1}{R_1}V_1 - \frac{1}{R_2}V_2 + I_1 = I_{v1}$$

$$\frac{1}{R_2}V_2 + \frac{1}{R_3}V_3 - \frac{1}{R_2}V_1 = -I_{v1}$$

De esta manera la matriz G de conductancias queda como:

$$G = \begin{bmatrix} \frac{1}{R_2} + \frac{1}{R_1} & -\frac{1}{R_2} \\ -\frac{1}{R_2} & \frac{1}{R_2} + \frac{1}{R_3} \end{bmatrix}$$

La matriz C de tamaño 2x1 quedaría de la siguiente manera:

$$\begin{bmatrix} 1 & -1 \end{bmatrix}$$

La matriz B de tamaño 1x2 quedaría de la siguiente manera:

$$\begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Mientras que la matriz D de tamaño 1x1 quedaría de la siguiente manera:

$$\begin{bmatrix} 0 \end{bmatrix}$$

Así el $MNA = A$ quedaría de la siguiente manera:

$$A = \begin{bmatrix} \frac{1}{R_2} + \frac{1}{R_1} & -\frac{1}{R_2} & 1 \\ -\frac{1}{R_2} & \frac{1}{R_2} + \frac{1}{R_3} & -1 \\ 1 & -1 & 0 \end{bmatrix}$$

El vector x queda de la siguiente manera:

$$x = \begin{bmatrix} V_1 \\ V_2 \\ I_{v1} \end{bmatrix}$$

El Vector z queda de la siguiente manera:

$$z = \begin{bmatrix} -I_1 \\ 0 \\ V_1 \end{bmatrix}$$

```
In [3]:
1 SCAPY.MNAf('1_circuito_con_fuentes_independientes.cir')
2 SCAPY.formula()
3
```

```
estamos verificando..
  elemento  nodo1 +  nodo2 - nodo3 + nodo4 -  miu miu2 miu3   t1   t2
0       V1        1        2     NaN     NaN  NaN  NaN  NaN  NaN  NaN
1       R1        1        0     NaN     NaN  NaN  NaN  NaN  NaN  NaN
2       R2        1        2     NaN     NaN  NaN  NaN  NaN  NaN  NaN
3       R3        2        0     NaN     NaN  NaN  NaN  NaN  NaN  NaN
4       I1        1        0     NaN     NaN  NaN  NaN  NaN  NaN  NaN
```

Out[3]:

$$\left(\begin{bmatrix} \frac{1}{R_2} + \frac{1}{R_1} & -\frac{1}{R_2} & 1 \\ -\frac{1}{R_2} & \frac{1}{R_3} + \frac{1}{R_2} & -1 \\ 1 & -1 & 0 \end{bmatrix}, \begin{bmatrix} V_1 \\ V_2 \\ Iv_1 \end{bmatrix}, \begin{bmatrix} -I_1 \\ 0 \\ V_1 \end{bmatrix} \right)$$

Figura 2.20: Circuito RLC en SCAPy

Elementos RLC

Definiciones:

- 1.- Reactancia capacitiva (XC = $1/2\pi$fC = $1/\omega$C) es la medida de la oposición de un capacitor al flujo de corriente eléctrica (se expresa en ohmios).
- 2.- Reactancia inductiva (XL = 2πfL = ωL) es la medida de la oposición de un inductor al flujo de la corriente (se expresa en ohmios).
- 3.- Susceptancia capacitiva (B_C), es la medida de un capacitor para almacenar energía en forma de campo eléctrico y es el inverso de la reactancia capacitiva (XC) expresada en siemens (S).
- 4.- Susceptancia inductiva (B_L), es la medida de un inductor para almacenar energía en forma de campo eléctrico y es el inverso de la reactancia inductiva (XL) expresada en siemens (S).

De Mourad Fakhfakh, Esteban Tlelo Cuautle y Francisco V. Fernández [6] página 57 establece que la susceptancia de un inductor es: $\frac{1}{LS}$ y en la página 37 expresa la susceptancia de un inductor como: CS De modo que $G_C = CS$ y $G_L = \frac{1}{LS}$
Donde la susceptancia el inverso de la reactancia, y S pertenece al dominio de la frecuencia, es decir, S = jω.

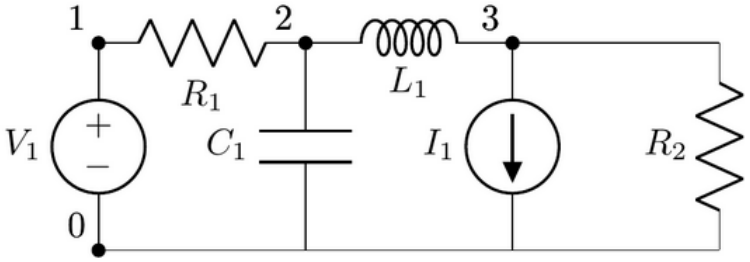


Figura 2.21: Circuito RLC

Se hace el análisis para el circuito de la figura 2.21
Nodo 1:

$$\frac{1}{R_1}V_1 - \frac{1}{R_1}V2 = I_{v1}$$

Nodo 2:

$$\frac{1}{R_1}V_2 + C_1SV_2 + \frac{1}{L_1S}V_2 - \frac{1}{R_1}V_1 - \frac{1}{L_1S}V_3 = 0$$

Nodo 3:

$$\frac{1}{L_1S}V_3 + \frac{1}{R_2}V_3 - \frac{1}{L_1S}V_2 = -1$$

$$\begin{bmatrix} \frac{1}{R_1} & -\frac{1}{R_1} & 0 & 1 \\ -\frac{1}{R_1} & \frac{1}{R_1} + C_1S + \frac{1}{L_1S} & -\frac{1}{L_1S} & 0 \\ 0 & -\frac{1}{L_1S} & \frac{1}{L_1S} + \frac{1}{R_2} & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ I_{v1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ I_1 \\ V_1 \end{bmatrix}$$

In [5]:
```
SCAPY.MNAf('2_elementos_RLC.cir')
SCAPY.formula()
```

```
estamos verificando..
  elemento  nodo1 +  nodo2 - nodo3 + nodo4 -  miu miu2 miu3   t1   t2
0       V1        1        0     NaN     NaN  NaN  NaN  NaN  NaN  NaN
1       R1        1        2     NaN     NaN  NaN  NaN  NaN  NaN  NaN
2       C1        2        0     NaN     NaN  NaN  NaN  NaN  NaN  NaN
3       L1        2        3     NaN     NaN  NaN  NaN  NaN  NaN  NaN
4       R2        3        0     NaN     NaN  NaN  NaN  NaN  NaN  NaN
5       I1        0        3     NaN     NaN  NaN  NaN  NaN  NaN  NaN
```

Out[5]:

$$\left(\begin{bmatrix} \frac{1}{R_1} & -\frac{1}{R_1} & 0 & 1 \\ -\frac{1}{R_1} & C_1 S + \frac{1}{R_1} + \frac{1}{L_1 S} & -\frac{1}{L_1 S} & 0 \\ 0 & -\frac{1}{L_1 S} & \frac{1}{R_2} + \frac{1}{L_1 S} & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ Iv_1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ I_1 \\ V_1 \end{bmatrix}\right)$$

Figura 2.22: Circuito RLC en SCAPy

Amplificadores operacionales

Para este elemento es agregado un renglón adicional, y una columna adicional a la formulación MNA en donde esta columna y este renglón, de acuerdo con Mourad Fakhfakh, Esteban Tlelo Cuautle y Francisco V. Fernández [6] dicho elemento representado en la figura 2.23

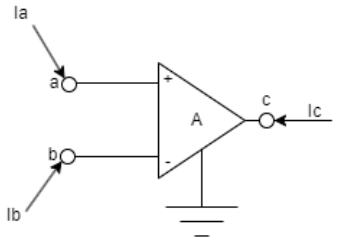


Figura 2.23: OpAmp

Se establecen los nodos: (a, b, c) respectivamente y de acuerdo con [4]:

$$A = \begin{bmatrix} G & B \\ C & D \end{bmatrix}, \quad x = \begin{bmatrix} v \\ j \end{bmatrix}, \quad z = \begin{bmatrix} i \\ e \end{bmatrix}$$

En C se agregan los nodos: (a, b) mientras que en B se agrega el nodo: (c), a continuación se presenta un ejemplo retomado de [6], figura 2.24:

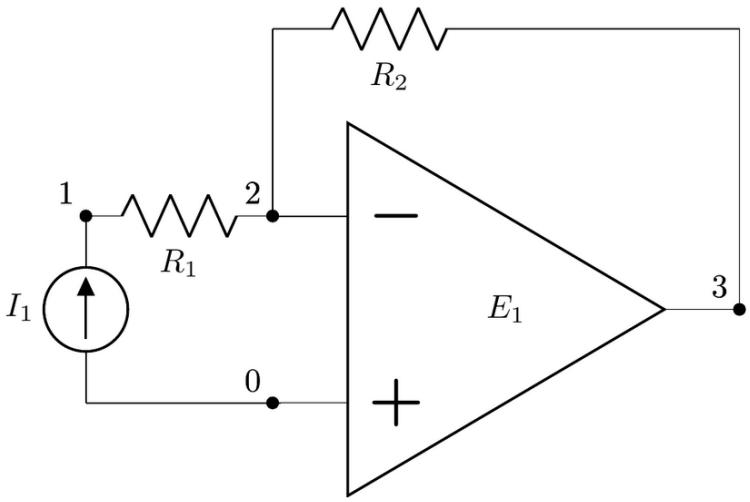


Figura 2.24: Circuito con amplificador operacional

De modo que haciendo el análisis por nodos.
Para el nodo 1, 2 y 3 respectivamente:

$$\frac{1}{R_1}V_1 - \frac{1}{R_1}V_2 = I_1$$

$$\frac{1}{R_2}V_2 + \frac{1}{R_1}V_2 - \frac{1}{R_1}V_1 - \frac{1}{R_2}V_3 = 0$$

$$\frac{1}{R_2}V_3 - \frac{1}{R_2}V_2 = 0$$

Formulando la matriz G:

$$\begin{bmatrix} \frac{1}{R_1} & -\frac{1}{R_1} & 0 \\ -\frac{1}{R_1} & \frac{1}{R_1}+\frac{1}{R_2} & -\frac{1}{R_2} \\ 0 & -\frac{1}{R_2} & \frac{1}{R_2} \end{bmatrix}$$

De este modo tomando en cuenta que de acuerdo con Mourad [6] al agregar B y C al MNA queda como:

$$\begin{bmatrix} \frac{1}{R_1} & -\frac{1}{R_1} & 0 & 0 \\ -\frac{1}{R_1} & \frac{1}{R_1}+\frac{1}{R_2} & -\frac{1}{R_2} & 0 \\ 0 & -\frac{1}{R_2} & \frac{1}{R_2} & 1 \\ 0 & -1 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ I_{op_Amp} \end{bmatrix} = \begin{bmatrix} -I_1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

```
In [6]:  1  SCAPY.MNAf('3_OpAmps.cir')
         2  SCAPY.formula()
```

```
estamos verificando..
  elemento  nodo1 +  nodo2 - nodo3 + nodo4 -  miu miu2 miu3   t1   t2
0       I1        1        0     NaN     NaN  NaN  NaN  NaN  NaN  NaN
1       R1        1        2     NaN     NaN  NaN  NaN  NaN  NaN  NaN
2       R2        2        3     NaN     NaN  NaN  NaN  NaN  NaN  NaN
3       O1        0        2     3.0     NaN  NaN  NaN  NaN  NaN  NaN
```

Out[6]:

$$\left(\begin{bmatrix} \frac{1}{R_1} & -\frac{1}{R_1} & 0 & 0 \\ -\frac{1}{R_1} & \frac{1}{R_2}+\frac{1}{R_1} & -\frac{1}{R_2} & 0 \\ 0 & -\frac{1}{R_2} & \frac{1}{R_2} & 1 \\ 0 & -1 & 0 & 0 \end{bmatrix}, \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ I_{OAmp1} \end{bmatrix}, \begin{bmatrix} -I_1 \\ 0 \\ 0 \\ 0 \end{bmatrix}\right)$$

Figura 2.25: Formulación matemática en SCAPy

Transformador

La teoría de *Mourad* [6] explica que un transformador puede ser modelado mediante dos inductores, y de igual manera en *LTSpice* [19], explica que para modelar un transformador, basta con acoplar dos inductores en donde de acuerdo con la figura 2.26:

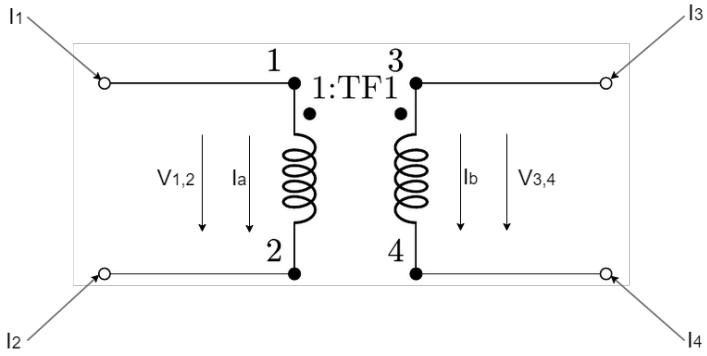


Figura 2.26: Transformador

El llenado de la matriz MNA es añadiendo dos filas y dos columnas en B, C, en la que se pondrán:

- Fila 1: Nodo 1 (negativo), Nodo 2 (positivo)
- Fila 1: Nodo 3 (negativo), Nodo 4 (positivo)
- Columna 1: Nodo 1 (positivo), Nodo 2 (negativo)
- Columna 1: Nodo 3 (positivo), Nodo 4 (negativo)

Adicionalmente el llenado en D será de la siguiente manera:

- Posición 1,1: S(inductor 1)
- Posición 1,2: s(M)
- Posición 2,1: s(M)
- Posición 2,2: s(inductor 2)

En donde M representa el modelo del circuito con inductancia mutua. En la figura 2.27

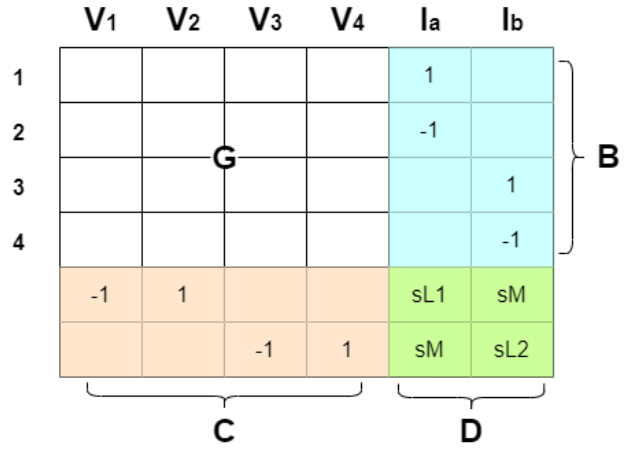


Figura 2.27: Llenado de la matriz MNA con datos del transformador

Para el circuito mostrado en la figura 2.28:

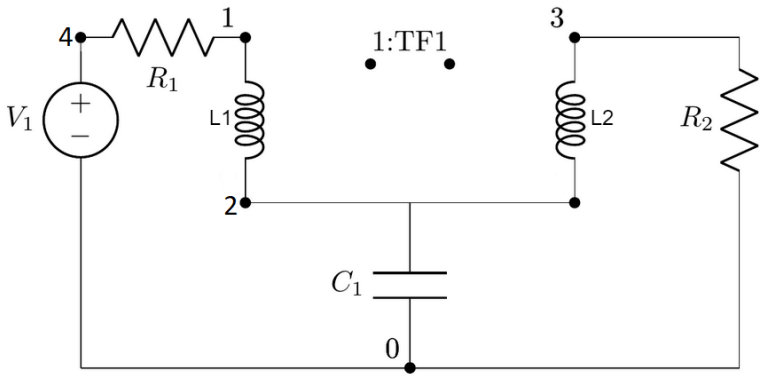


Figura 2.28: Circuito transformador

Se realiza el análisis siguiente de acuerdo a los nodos.
Nodo 1:

$$\frac{1}{R_1}V_1 - \frac{1}{R_1}V_4 = 0$$

Nodo 2:

$$SC_1V_2 = 0$$

Nodo 3:

$$\frac{1}{R_2}V_3 = 0$$

Nodo 4:

$$\frac{1}{R_1}V_4 - \frac{1}{R_1}V_1$$

La matriz A queda de la siguiente manera, tomando en cuenta que input (V_1) está conectada entre los nodos 4 y 0, y retomando la metodología de Mourad [6]:

$$A = \begin{bmatrix} \frac{1}{R_1} & 0 & 0 & -\frac{1}{R_1} & 0 & -1 & 0 \\ 0 & SC_1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & \frac{1}{R_2} & 0 & 0 & 0 & -1 \\ -\frac{1}{R_1} & 0 & 0 & \frac{1}{R_1} & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & sL_1 & sM \\ 0 & 1 & -1 & 0 & 0 & sM & sL_2 \end{bmatrix}$$

El vector x se formula de la siguiente manera:

$$x = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ I_{v1} \\ I_{L1} \\ I_{L2} \end{bmatrix}$$

El vector z se formula de la siguiente manera:

$$z = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ V_1 \\ 0 \\ 0 \end{bmatrix}$$

```
In [3]:
1 SCAPY.MNAf('4_transformador.cir')
2 SCAPY.formula()
```

```
estamos verificando..
  elemento  nodo1 +  nodo2 - nodo3 + nodo4 -  miu miu2 miu3   t1   t2
0       V1        4        0     NaN     NaN  NaN  NaN  NaN  NaN  NaN
1       R1        4        1     NaN     NaN  NaN  NaN  NaN  NaN  NaN
2   NaN_L_        1        2     NaN     NaN  NaN  NaN  NaN  NaN  NaN
3   NaN_L_        3        2     NaN     NaN  NaN  NaN  NaN  NaN  NaN
4       C1        2        0     NaN     NaN  NaN  NaN  NaN  NaN  NaN
5       R2        3        0     NaN     NaN  NaN  NaN  NaN  NaN  NaN
6       K1        1        2       3       2  NaN  NaN  0.5   L1   L2
```

Out[3]:

$$\left(\begin{bmatrix} \frac{1}{R_1} & 0 & 0 & -\frac{1}{R_1} & 0 & -1 & 0 \\ 0 & C_1 S & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & \frac{1}{R_2} & 0 & 0 & 0 & -1 \\ -\frac{1}{R_1} & 0 & 0 & \frac{1}{R_1} & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & L1S & SM \\ 0 & 1 & -1 & 0 & 0 & SM & L2S \end{bmatrix}, \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ Iv_1 \\ I_{L1} \\ I_{L2} \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ V_1 \\ 0 \\ 0 \end{bmatrix}\right)$$

Figura 2.29: Formulación matemática en SCAPy

En donde el vector de incógnitas siempre y para rodos los casos de la formulación del método MNA es y será el vector, x mientras que el vector z para esta formulación siempre será el vector donde se encontraran los inputs del sistema algebraico, es decir, las fuentes independientes de corriente y o voltaje siempre serán los inputs.

Fuente de voltaje controlada por voltaje (VCVS)

La descripción de una fuente de voltaje controlada por voltaje, tomando como ejemplo el circuito de la figura 2.30, la fuente E_1 es una fuente de voltaje que se encuentra conecta entre el nodo 4 y el nodo 3 y que son dependientes del voltaje entre los nodos 2 y 3 con una ganancia E_{x1} veces el voltaje de dichos nodos, de igual forma la fuente E_2 es una fuente de voltaje conectada entre los nodos 5 y 3 y que depende de los nodos 2 y 3 respectivamente, con una ganancia E_{x2} veces el voltaje de los nodos dichos anteriormente, las resistencias R_4 y R_5 respectivamente, representan las cargas de las fuentes controladas.

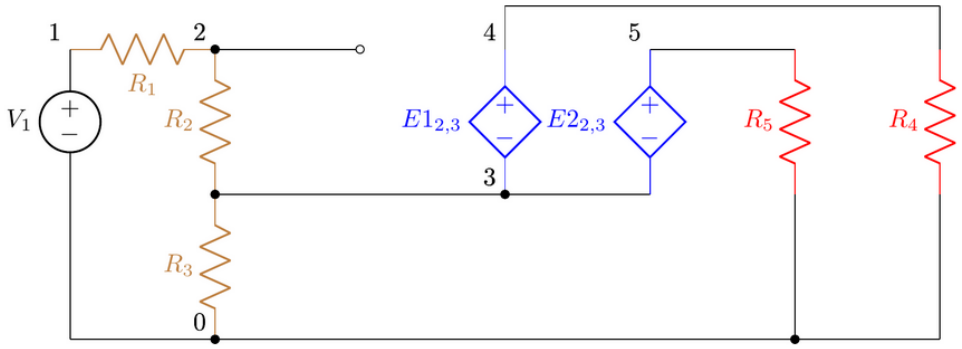


Figura 2.30: VCVS

Se analizan los nodos:
Nodo 1:

$$\frac{1}{R_1}V_1 - \frac{1}{R_1}V_2 = I_{V1}$$

Nodo 2:

$$\frac{1}{R_1}V_2 + \frac{1}{R_2}V_2 - \frac{1}{R_1}V_1 - \frac{1}{R_2}V_3 = 0$$

Nodo 3:

$$\frac{1}{R_2}V_3 + \frac{1}{R_3}V_3 - \frac{1}{R_2}V_2 = 0$$

Nodo 4:

$$\frac{1}{R_4}V_4 = 0$$

Nodo 5:

$$\frac{1}{R_5}V_5 = 0$$

Con estas ecuaciones, se estructura la matriz A, y de acuerdo con el método de los STAMPS, para la fuente V_1 y para las fuentes independientes E_1, E_2:

$$\begin{bmatrix} \frac{1}{R_1} & -\frac{1}{R_1} & 0 & 0 & 0 & 1 & 0 & 0 \\ -\frac{1}{R_1} & \frac{1}{R_1}+\frac{1}{R_2} & -\frac{1}{R_2} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{R_2} & \frac{1}{R_2}+\frac{1}{R_3} & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & \frac{1}{R_4} & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{R_5} & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -E_1 & E_1-1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -E_2 & E_2-1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ I_{V1} \\ I_{E1} \\ I_{E2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ V1 \\ 0 \\ 0 \end{bmatrix}$$

In [8]:
```
SCAPY.MNAf('5_vcvs.cir')
SCAPY.formula()
```

```
estamos verificando..
  elemento  nodo1 +  nodo2 - nodo3 + nodo4 -  miu miu2 miu3   t1   t2
0       V1       1       0    NaN     NaN  NaN  NaN  NaN  NaN  NaN
1       R1       1       2    NaN     NaN  NaN  NaN  NaN  NaN  NaN
2       R2       2       3    NaN     NaN  NaN  NaN  NaN  NaN  NaN
3       R3       3       0    NaN     NaN  NaN  NaN  NaN  NaN  NaN
4       R4       4       0    NaN     NaN  NaN  NaN  NaN  NaN  NaN
5       R5       5       0    NaN     NaN  NaN  NaN  NaN  NaN  NaN
6       E1       4       3    2.0     3.0   g1  NaN  NaN  NaN  NaN
7       E2       5       3    2.0     3.0   g2  NaN  NaN  NaN  NaN
```

Out[8]:

$$\left(\begin{bmatrix} \frac{1}{R_1} & -\frac{1}{R_1} & 0 & 0 & 0 & 1 & 0 & 0 \\ -\frac{1}{R_1} & \frac{1}{R_2}+\frac{1}{R_1} & -\frac{1}{R_2} & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{R_2} & \frac{1}{R_3}+\frac{1}{R_2} & 0 & 0 & 0 & -1 & -1 \\ 0 & 0 & 0 & \frac{1}{R_4} & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{R_5} & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -E_1 & E_1-1 & 1 & 0 & 0 & 0 & 0 \\ 0 & -E_2 & E_2-1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ Iv_1 \\ I_{E1} \\ I_{E2} \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ V_1 \\ 0 \\ 0 \end{bmatrix}\right)$$

Figura 2.31: Formulación matemática de fuente VCVS

Fuente de voltaje controlada por corriente (CCVS)

Para abordar esta fuente se toma el circuito de la figura 2.32, en el presente circuito, se encuentran dos fuentes de voltaje controladas por corriente, conectadas en los nodos (5,3) y (6,0) respectivamente, la corriente que controla estas fuentes, es la corriente I_{V1}, que a continuación se describe en las ecuaciones de cada nodo.

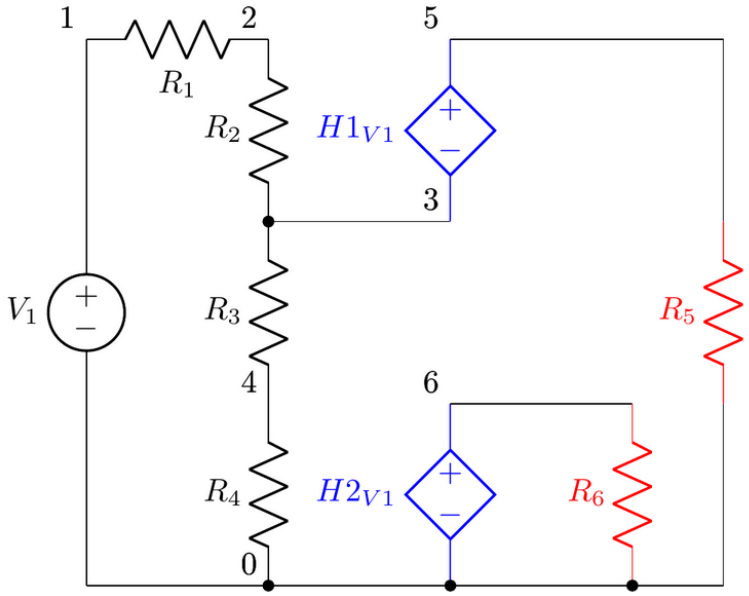


Figura 2.32: CCVS

Nodo 1:

$$\frac{1}{R_1}V_1 - \frac{1}{R_1}V_2 = I_{v1}$$

Nodo 2:

$$\frac{1}{R_1}V_2 + \frac{1}{R_2}V_2 - \frac{1}{R_1}V_1 - \frac{1}{R_2}V_3 = 0$$

Nodo 3:

$$\frac{1}{R_2}V_3 + \frac{1}{R3}V_3 - \frac{1}{R_2}V_2 - \frac{1}{R_3}V_4 = 0$$

Nodo 4:

$$\frac{1}{R_3}V_4 + \frac{1}{R_4}V_4 - \frac{1}{R_3}V_3 = 0$$

Nodo 5:

$$\frac{1}{R_5}V_5 = 0$$

Nodo 6:

$$\frac{1}{R_6}V_6 = 0$$

Formulando el sistema de ecuaciones a partir de las ecuaciones nodales, y tomando a las fuentes de voltaje controladas por corriente (H_1 y H_2).

$$\begin{bmatrix} \frac{1}{R_1} & -\frac{1}{R_1} & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -\frac{1}{R_1} & \frac{1}{R_1}+\frac{1}{R_2} & -\frac{1}{R_2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{R_2} & \frac{1}{R_2}+\frac{1}{R3} & -\frac{1}{R_3} & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -\frac{1}{R_3} & \frac{1}{R_3}+\frac{1}{R_4} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{R_5} & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{R_6} & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & -H & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -H & 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \\ I_{V1} \\ I_{H1} \\ I_{H2} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ V1 \\ 0 \\ 0 \end{bmatrix}$$

```
In [14]: 1 SCAPY.MNAf('6_ccvs.cir')
         2 SCAPY.formula()
```

estamos verificando..

	elemento	nodo1 +	nodo2 -	nodo3 +	nodo4 -	miu	miu2	miu3	t1	t2
0	V1	1	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	R1	1	2	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	R2	2	3	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	R3	3	4	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	R4	4	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	R5	5	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
6	R6	6	0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
7	H1	5	3	V1	g1	NaN	NaN	NaN	NaN	NaN
8	H2	6	0	V1	g2	NaN	NaN	NaN	NaN	NaN

Out[14]:
$$\left(\begin{bmatrix} \frac{1}{R_1} & -\frac{1}{R_1} & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ -\frac{1}{R_1} & \frac{1}{R_2}+\frac{1}{R_1} & -\frac{1}{R_2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -\frac{1}{R_2} & \frac{1}{R_3}+\frac{1}{R_2} & -\frac{1}{R_3} & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & -\frac{1}{R_3} & \frac{1}{R_4}+\frac{1}{R_3} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{R_5} & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \frac{1}{R_6} & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & -H_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & -H_2 & 0 & 0 \end{bmatrix}, \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \\ I_{V1} \\ I_{H1} \\ I_{H2} \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ V_1 \\ 0 \\ 0 \end{bmatrix} \right)$$

Figura 2.33: Formulación matemática de fuente CCVS

Fuente de corriente controlada por voltaje (VCCS)

Para la fuente de VCCS se toma el circuito de la figura 2.34 en donde se encuentra una fuente G_1 que refiere a una fuente de corriente que es controlada por el voltaje en los nodos 1, 2 siendo el nodo 1 positivo, y el nodo 2 negativo.
La resistencia de carga para esta fuente es la R_3 y es representada por el color *rojo*, nuevamente retomando los $STAMPS$ se presenta su análisis nodal seguido de su formulación algebraica lineal, mediante el método MNA

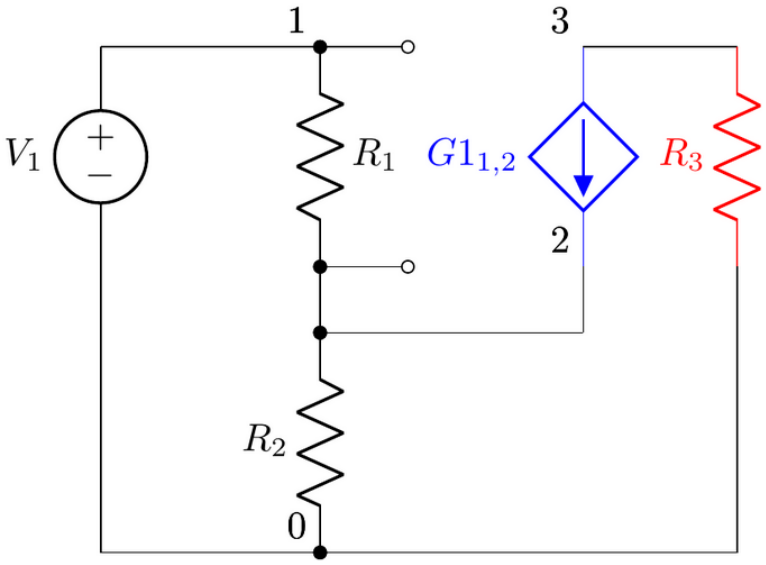


Figura 2.34: VCCS

Nodo 1:

$$\frac{1}{R_1}V_1 - \frac{1}{R_1}V_2 = I_{V1}$$

Nodo 2:

$$\frac{1}{R_1}V_2 + \frac{1}{R_2}V_2 - \frac{1}{R_1}V_1 = 0$$

$$\frac{1}{R_3}V_3 = 0$$

Partiendo de las ecuaciones de cada nodo, se formula el sistema de ecuaciones y para la fuente G_1 se usa el método de los STAMPS.

$$\begin{bmatrix} \frac{1}{R_1} & -\frac{1}{R_1} & 0 & 1 \\ -\frac{1}{R_1} - G & \frac{1}{R_1} + \frac{1}{R_2} + G & 0 & \\ G & -G & \frac{1}{R_3} & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ I_{V1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ V1 \end{bmatrix}$$

```
In [13]:
1 SCAPY.MNAf('7_vccs.cir')
2 SCAPY.formula()
```

```
estamos verificando..
  elemento  nodo1 +  nodo2 - nodo3 + nodo4 -  miu miu2 miu3   t1   t2
0       V1        1        0     NaN     NaN  NaN  NaN  NaN  NaN  NaN
1       R1        1        2     NaN     NaN  NaN  NaN  NaN  NaN  NaN
2       R2        2        0     NaN     NaN  NaN  NaN  NaN  NaN  NaN
3       R3        3        0     NaN     NaN  NaN  NaN  NaN  NaN  NaN
4       G1        3        2     1.0     2.0   g1  NaN  NaN  NaN  NaN
```

Out[13]:

$$\left(\begin{bmatrix} \frac{1}{R_1} & -\frac{1}{R_1} & 0 & 1 \\ -G_1 - \frac{1}{R_1} & G_1 + \frac{1}{R_2} + \frac{1}{R_1} & 0 & 0 \\ G_1 & -G_1 & \frac{1}{R_3} & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ Iv_1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ V_1 \end{bmatrix}\right)$$

Figura 2.35: Formulación matemática de fuente VCCS

Fuente de corriente controlada por corriente (CCCS)

Por último, el abordamiento de esta fuente es de forma similar a las demás fuentes dependientes, y de igual forma se analizará mediante las conductancias de los nodos para encontrar la matriz G, recordando que dicha matriz se obtiene de los elementos pasivos, y haciendo uso del método de los $STAMPS$ para formular el sistema de ecuaciones para las presentes tres fuentes del circuito de la figura 2.36

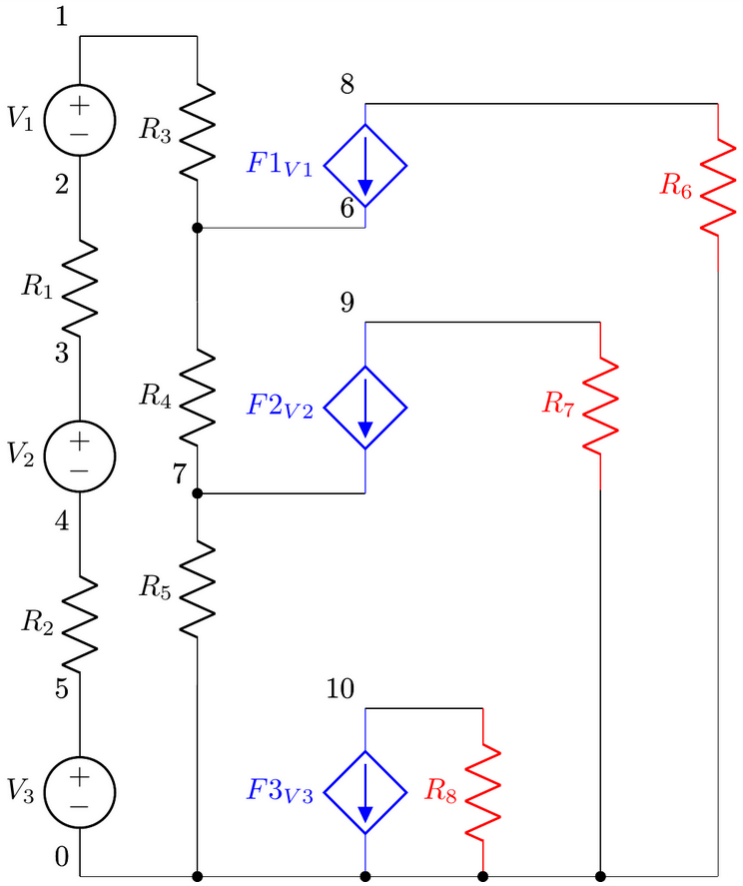


Figura 2.36: CCCS

Nodo 1:

$$\frac{1}{R_3}V_1 - \frac{1}{R_3}V_6 = I_{V1}$$

Nodo 2:

$$\frac{1}{R_1}V_2 - \frac{1}{R_1}V_3 = -I_{V1}$$

Nodo 3:

$$\frac{1}{R_1}V_3 - \frac{1}{R_1}V_2 = I_{V2}$$

Nodo 4:

$$\frac{1}{R_2}V_4 - \frac{1}{R_2}V_5 = -I_{V2}$$

Nodo 5:

$$\frac{1}{R2}V_5 - \frac{1}{R_2}V_4 = I_{V3}$$

Nodo 6:

$$\frac{1}{R_3}V_6 + \frac{1}{R_4}V_6 - \frac{1}{R_4}V_7 - \frac{1}{R3}V_1 = 0$$

Nodo 7:

$$\frac{1}{R_4}V_7 + \frac{1}{R_5}V7 - \frac{1}{R_4}V_6 = 0$$

Nodo 8:

$$\frac{1}{R_6}V_8 = 0$$

Nodo 9:

$$\frac{1}{R_7}V_9 = 0$$

Nodo 10:

$$\frac{1}{R_8}V_{10} = 0$$

Se formula el sistema de ecuaciones por el método MNA, $Ax = z$

$$A = \begin{bmatrix} \frac{1}{R_3} & 0 & 0 & 0 & 0 & -\frac{1}{R_3} & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{1}{R_1} & -\frac{1}{R_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & -\frac{1}{R_1} & \frac{1}{R_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{1}{R_2} & -\frac{1}{R_2} & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -\frac{1}{R_2} & \frac{1}{R2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -\frac{1}{R3} & 0 & 0 & 0 & 0 & \frac{1}{R_3} + \frac{1}{R_4} & -\frac{1}{R_4} & 0 & 0 & 0 & -F_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{R_4} & \frac{1}{R_4} + \frac{1}{R_5} & 0 & 0 & 0 & 0 & -F_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{R_6} & 0 & 0 & F_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{R_7} & 0 & 0 & F_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{R_8} & 0 & 0 & F_3 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$x = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \\ V_7 \\ V_8 \\ V_9 \\ V_{10} \\ I_{V1} \\ I_{v2} \\ I_{V3} \end{bmatrix}$$

$$z = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ V1 \\ V2 \\ V3 \end{bmatrix}$$

In [12]:
```
SCAPY.MNAf('8_cccs.cir')
SCAPY.formula()
```

```
estamos verificando..
   elemento  nodo1 +  nodo2 - nodo3 + nodo4 -  miu miu2 miu3   t1   t2
0        V1       1       2     NaN     NaN  NaN  NaN  NaN  NaN  NaN
1        V2       3       4     NaN     NaN  NaN  NaN  NaN  NaN  NaN
2        V3       5       0     NaN     NaN  NaN  NaN  NaN  NaN  NaN
3        R1       2       3     NaN     NaN  NaN  NaN  NaN  NaN  NaN
4        R2       4       5     NaN     NaN  NaN  NaN  NaN  NaN  NaN
5        R3       1       6     NaN     NaN  NaN  NaN  NaN  NaN  NaN
6        R4       6       7     NaN     NaN  NaN  NaN  NaN  NaN  NaN
7        R5       7       0     NaN     NaN  NaN  NaN  NaN  NaN  NaN
8        R6       8       0     NaN     NaN  NaN  NaN  NaN  NaN  NaN
9        R7       9       0     NaN     NaN  NaN  NaN  NaN  NaN  NaN
10       R8      10       0     NaN     NaN  NaN  NaN  NaN  NaN  NaN
11       F1       8       6      V1      r1  NaN  NaN  NaN  NaN  NaN
12       F2       9       7      V2      r2  NaN  NaN  NaN  NaN  NaN
13       F3      10       0      V3      r3  NaN  NaN  NaN  NaN  NaN
```

Out[12]:

$$\left(\begin{bmatrix} \frac{1}{R_3} & 0 & 0 & 0 & 0 & -\frac{1}{R_3} & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & \frac{1}{R_1} & -\frac{1}{R_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & -\frac{1}{R_1} & \frac{1}{R_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & \frac{1}{R_2} & -\frac{1}{R_2} & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -\frac{1}{R_2} & \frac{1}{R_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ -\frac{1}{R_3} & 0 & 0 & 0 & 0 & \frac{1}{R_4} + \frac{1}{R_3} & -\frac{1}{R_4} & 0 & 0 & 0 & -F_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -\frac{1}{R_4} & \frac{1}{R_5} + \frac{1}{R_4} & 0 & 0 & 0 & 0 & -F_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{R_6} & 0 & 0 & F_1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{R_7} & 0 & 0 & F_2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{R_8} & 0 & 0 & F_3 \\ 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \\ V_7 \\ V_8 \\ V_9 \\ V_{10} \\ Iv_1 \\ Iv_2 \\ Iv_3 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ V_1 \\ V_2 \\ V_3 \end{bmatrix} \right)$$

Figura 2.37: Formulación matemática de fuente CCCS

2.4. Conclusiones

Este capítulo muestra toda la metodología de la formulación matemática para los elementos (lineales) que se simularán en la librería SCAPy, que resultan importantes para ilustrar la metodología de formulación matemática.

Como pudo observarse, partiendo de la ley de corrientes de Kirchhoff se puede construir el método de los STAMPS un método innovador que permite resumir el llenado del sistema de ecuaciones en general para cada elemento, y que sirve como guía para formular el algoritmo de parseo.

El presente capítulo dos resulta ser importante para el desarrollo de esta tesis, ya que parte fundamental recae en la formulación matemática esencial para obtener resultados confiables, la formulación matemática debe ser esencialmente confiable, además de conectar con el capítulo tres, en donde se toma como punto de partida todo lo visto en el capítulo dos.

Capítulo 3

Método de solución

En el presente capítulo tres, se aborda una justificación del porqué se busca un método de solución que permita el ahorro computacional, en la búsqueda del método de solución se presentan los orígenes de DDD (Diagrama de Decisión a Determinantes) presentando a BDD (Diagrama de Decisiones Binarias) hasta llegar al DDD de Greedy y DDD por capas, posteriormente, se aborda el planteamiento del algoritmo de solución presentando un DDD por capas bien definido, explicado y funcional. Este mismo algoritmo se encuentra disponible en el alojamiento en GitHub y disponible en PyPI para su instalación. Puede consultarse el apéndice A de esta Tesis.

3.1. Teorema de fundamental para resolver determinantes

Como resultado a la formulación matricial por el parser descrito en el capítulo dos de esta tesis, la forma del planteamiento matemático es una matriz cuadrada de A^{nxn} con n incógnitas que representan las tensiones nodales en la que se encuentran conectados los elementos eléctricos y las corrientes, de forma general el sistema de ecuaciones es presentado por la forma:

$$Ax = b$$

En general son considerados dos métodos de solución a sistemas lineales que comprenden dos categorías una es basada en composiciones jerárquicas y la segunda está basada en aproximaciones [20].

La idea de composiciones jerárquicas basa su forma en el método de solución de la regla de Cramer [20].

$$\frac{\sum_{i=1}^{n} b_i(-1)^{i+k}\Delta A_{i,k}}{\Delta A}$$

Donde: Δ denota el determiante de la matriz A y $(-1)^{i+k}\Delta A$ denota el cofactor del determinante A. La regla de Cramer es usada de forma estratégica para obtener las funciones de transferencia en cada incógnita, posteriormente esta función de transferencia $H(s)$ pasa del dominio de la frecuencia al dominio del tiempo usando la transformada de La-place.

De este modo, la forma de la ecuación de acuerdo al teorema de La-place en una matriz cuadrada:

$$\sum_{i=1}^{n} a_{i,j} \cdot \Delta A_{i,j}$$

En donde $\Delta A_{i,j} = (-1)^{i+k} \cdot M_{i,j}$ siendo M el menor, este teorema ilustra la idea de que para resolver el determinante de una matriz A^{nxn} se requieren $n!$ operaciones dicho de otro modo mientras que para una matriz A^{2x2} se requieren 2 multiplicaciones, para una matriz A^{10x10} se requieren 3,628,800 multiplicaciones para encontrar el resultado. Un símbolo en un lenguaje de alto nivel implica un objeto, Python permite hacer operaciones elementales entre objetos, de esta manera se obtienen las expresiones algebraicas simbólicas, sin embargo, por muchos años el problema de reducir el número de las operaciones multiplicación al encontrar las incógnitas en un sistema de ecuaciones.

Un ejemplo que permite ilustrar la problemática que los investigadores se han enfrentado al resolver este tipo de sistemas lineales algebraicos es tomando como base un procesador ryzen 7 de la serie 5800H y los siguientes ciclos:

$$a1 = \sum_{1}^{3{,}628{,}800} x \cdot y$$

$$a2 = \sum_{1}^{3{,}628{,}800} \frac{1}{y} \cdot \frac{1}{x}$$

$$a3 = \sum_{1}^{3{,}628{,}800} (\frac{1}{y} \cdot \frac{1}{x} + s\frac{x \cdot y}{s^2 \cdot (x+y)}) \cdot (\frac{1}{x})$$

El ciclo $a1$ representa la multiplicación numérica en cada iteración del ciclo el resultado de la multiplicación es almacenado como un nuevo número y la próxima iteración será una nueva multiplicación normal, mientras que el ciclo a2 acerca un poco mejor la problemática, simbólicamente se tienen dos fracciones multiplicándose, al no poderse reducir más; sin embargo, sigue siendo ideal, el ciclo a3 permite dar un panorama más profundo de que en una multiplicación puede haber más operaciones, sin embargo, sigue siendo ideal, porque en cada iteración del ciclo no solo hay operaciones simbólicas, además de esto la expresión simbólica crece aún más en cada nueva iteración, para ilustrar esta naturaleza de computar símbolos objeto se computaron los ciclos a1, a2 y a3 obteniendo los siguientes tiempos de cómputo 3.1:

Cuadro 3.1: Tabla de tiempos en cada ciclo

Ciclo	Tiempo
a1	1.6684s
a2	9.2317s
a3	30.1223

La tabla 3.1 ilustra el problema al trabajar con símbolos, y que en una multiplicación puede haber n términos, de modo que una teoría bien aceptada el método de Diagrama de Decisión

para Determinantes propuesto por primera vez en 1997 por C.J. Richard Shi y Xiangdong Tan [21], pero fue hasta el 2001 que aparecieron las primeras generaciones eficientes [22]. El algoritmo ha ido evolucionando y se puede encontrar dos ideas: la primera es el DDD de Greedy y la segunda es DDD por capas y presentada por Sheldon Tan y Esteban Tlelo [20] y retomado por Efraín [23]

3.2. Introducción al Diagrama de Decisión de Determinantes

Un sistema de ecuaciones resultante entregado por la formulación MNA usualmente suele tener en su mayoría ceros en los triángulos superior e inferior. Siendo innecesario computar todo el determinante con el teorema de La-place un diagrama permite discriminar los cálculos que vana a ser multiplicados por cero y evitar hacer todo el desarrollo posterior, y además guardar los datos resultantes de cada multiplicación en un área de la memoria para que en el próximo cálculo sea comprado y si el cálculo ya se encuentra en la memoria se mande a llamar evitando un conjunto de operaciones elementales esta idea permite ahorrar tiempo de cómputo.

3.2.1. BDD

Un paso antes de un DDD es un BDD (Diagrama de Decisión Binaria) Esta metodología estima las posibilidades de switcheo en un circuito [24] partiendo del siguiente diagrama de contactos figura 3.1:

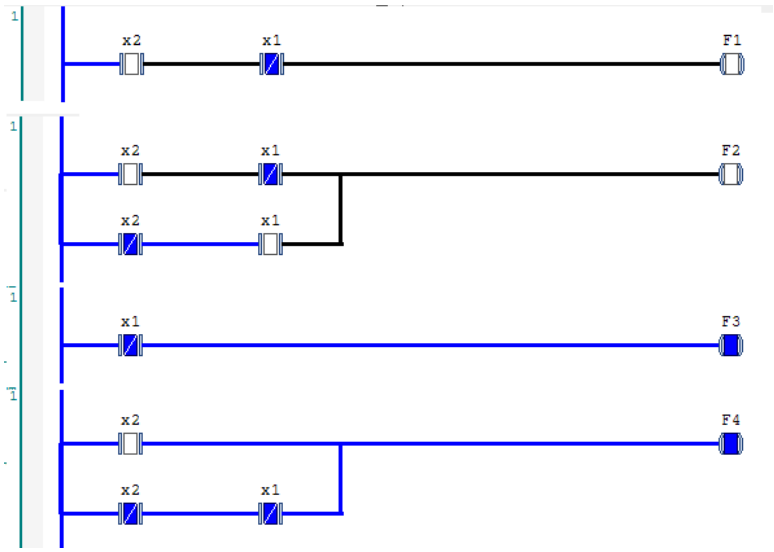


Figura 3.1: Lógica de relé

En donde las funciones son dadas en la tabla 3.2

Un BDD mapea los posibles switcheos de esta lógica a continuación la entrada de la función uno "$F1$.$^{\text{en}}$tra por $x2$ y solo es validada si en su camino llega con un uno lógico de x2 pasa a x1 esta transferencia es representada por la operación AND para llegar a x1, una vez en x1 para llegar al uno lógico para por cero entonces el resultado es una operación AND de x1 negado el

Cuadro 3.2: Funciones

Función	Operación
F1	(x2) AND (NOT(x1))
F2	x2 OR X1
F3	NOT x1
F4	(x2) OR (NOT (x1))

mapeo de $F2$ es con la entrada en x2 y una vez en x2 tiene dos caminos para llegar a x1 que a su vez conecta con un uno lógico la operación que representa este camino es una operación OR el mapeo de $F3$ y $F4$ son análogos.

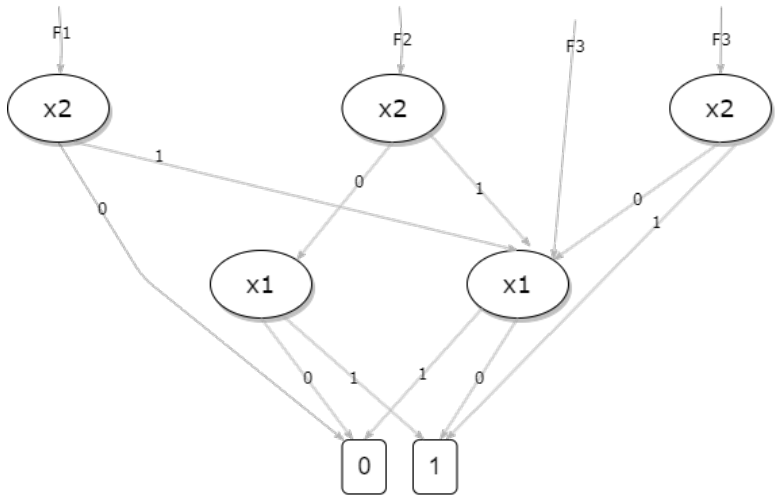


Figura 3.2: BDD

3.3. El Diagrama de Decisión de Determinantes (DDD) para la solución del sistema de ecuaciones del circuito bajo análisis

Un diagrama de decisión por determinantes se basa en una topología para resolver determinantes de tamaño nxn en donde la regla de Sarrus se limita a sistemas de 3x3 y donde usar una expansión de cofactores o usar la eliminación de Gauss hace que el cálculo sea extenso, este motivo hizo que por muchos años el cálculo simbólico tuviera un estancamiento, el método DDD es un diagrama en donde en primera instancia va guardando los cálculos y posteriormente consulta los nuevos posibles cálculos, en caso de tener el cálculo hecho, extrae el cálculo guardado evitando computar la aritmética del nuevo cálculo, en otras palabras el método DDD comparte el mayor número de cálculos aritméticos que puedan hacerse en un determinante, además es importante que entre mayor crece un determinante, el número de operaciones crece de manera exponencial, el hecho de compartir sub operaciones permite ahorrar mucho tiempo de cómputo, si bien en determinantes peños no es tan notorio, en determinantes grandes es de mucha importancia.

Tan [20] se explican dos metodologías de un DDD y básicamente existe el DDD de Greedy y el DDD por capas o por renglones.
Sea la matriz A:

$$A = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix}$$

En donde su determinante es:
a*f*k*p - a*f*l*o - a*g*j*p + a*g*l*n + a*h*j*o - a*h*k*n - b*e*k*p + b*e*l*o + b*g*i*p - b*g*l*m - b*h*i*o + b*h*k*m + c*e*j*p - c*e*l*n - c*f*i*p + c*f*l*m + c*h*i*n - c*h*j*m - d*e*j*o + d*e*k*n + d*f*i*o - d*f*k*m - d*g*i*n + d*g*j*m
El DDD de Greedy para la matriz A consiste en formular los siguientes cofactores:

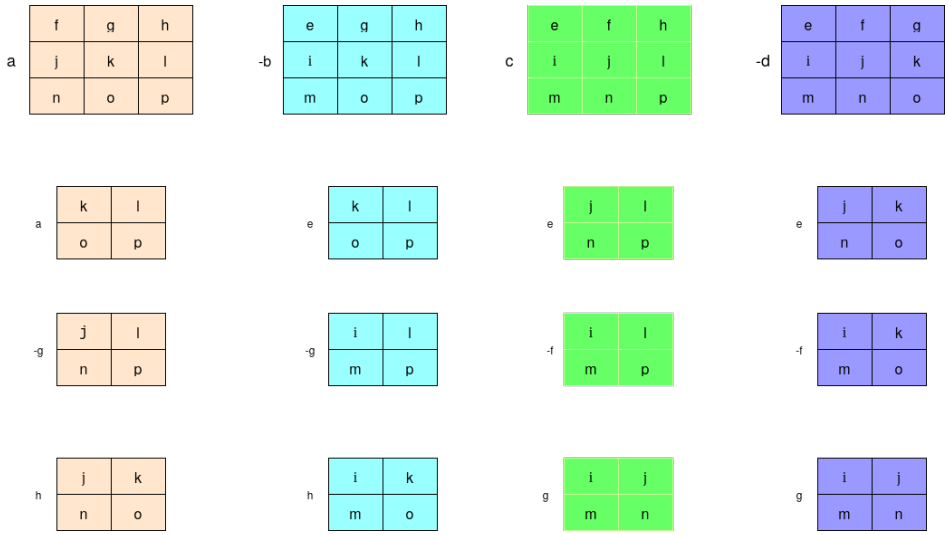


Figura 3.3: Cofactores de la matriz A

3.3.1. DDD Greedy

Se formula a partir de los cofactores en donde cada variable es encerrada en un círculo y de cada variable sale dos ramificaciones, una en línea continua y otra en línea punteada, la línea continua transfiere la variable a la siguiente rama y multiplica a la variable siguiente por (1), mientras que la línea punteada transfiere la variable a la siguiente capa y multiplica a la variable siguiente por (-1). Todas las soluciones válidas son mandadas a un 1, mientras que las salidas no válidas son mandadas a un 0.

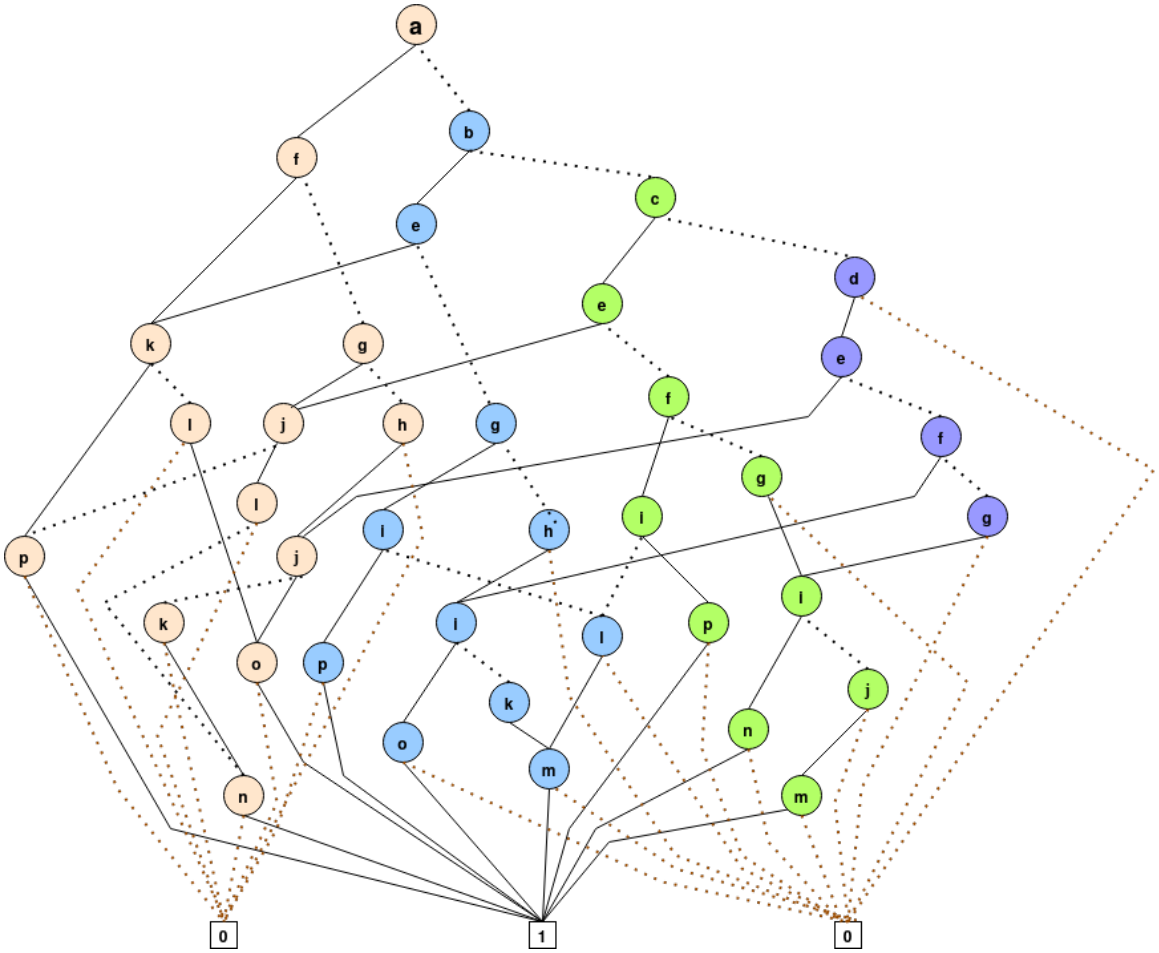


Figura 3.4: DDD de Greedy

En la figura 3.5 se muestra una ramificación que lleva a cero, de modo que no es considerada como una solución.

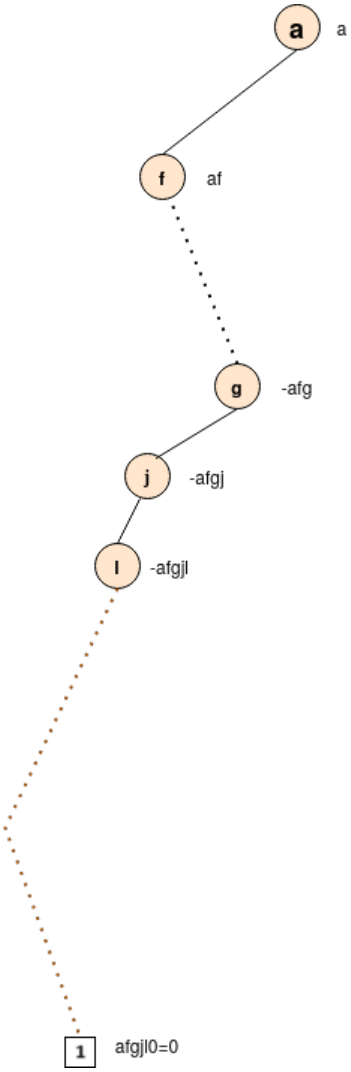


Figura 3.5: DDD de Greedy rama a cero

Mientras que dos de sus soluciones son obtenidas de la siguiente manera:

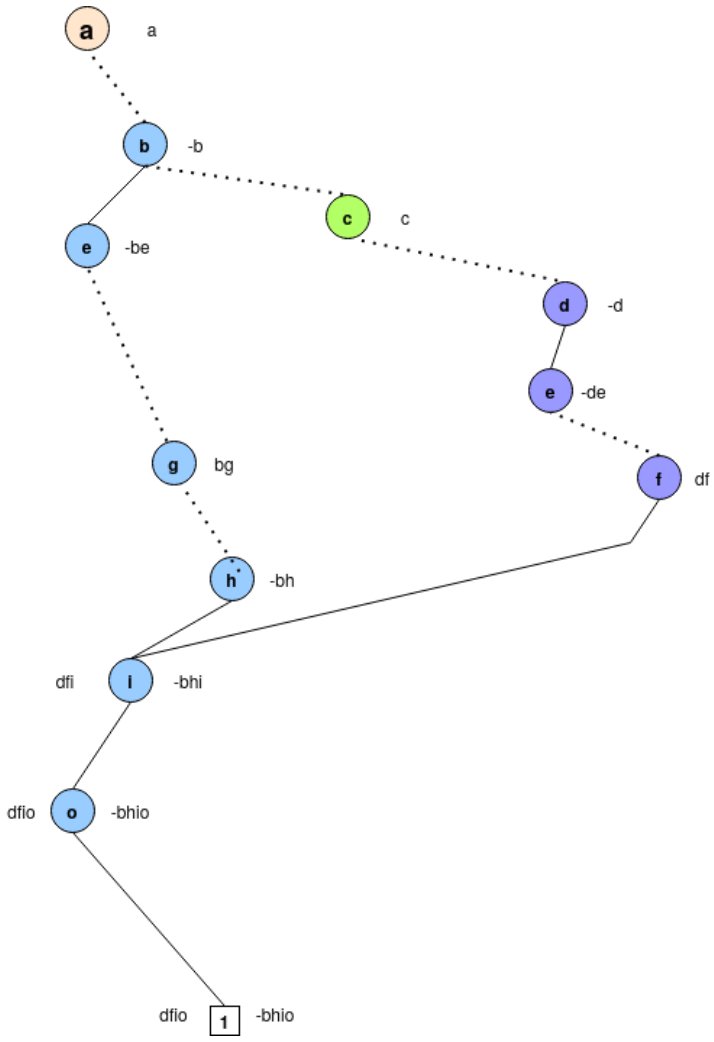


Figura 3.6: DDD de Greedy rama a uno

Otra característica a resaltar es que en los determinantes mayores a 3x3 las operaciones crecen exponencialmente, como se observa en el DDD de Greedy el mismo diagrama comparte ramificaciones entre sí, esto reduce cálculos, por el ejemplo los cálculos son compartidos entre las ramas azules y morado, visto de otro modo, en el primer cofactor (color crema), son usadas 12 variables, para el segundo cofactor (color azul) son usadas 11 variables, para el tercer cofactor (color verde) son usadas 10 variables, mientras que para el cuarto cofactor apenas son requeridas 4 variables, este método reduce las variables requeridas de los cofactores para el cálculo del determinante.

3.3.2. DDD por capas

Este diagrama también es conocido como DDD por renglones, la metodología es mucho más ordenada, la primera capa consiste en forma un renglón con el primer renglón de la matriz A,

el segundo renglón consiste en elegir las variables agrupadas de los cofactores, si hay un grupo de variables idénticas esta se omite, es decir en todas las variables que puedan estar repetidas estas se eliminan dejando solo un grupo, y es aquí donde se comparten los grupos de variables, de igual manera para el tercer renglón y así hasta n renglones donde el cofactor resultante sea de 2x2, siempre omitiendo las variables repetidas.

La segunda característica de importancia es que todos los renglones formados por grupos son ligados por una línea punteada, esta línea entre grupo multiplica a la variable vecina por un (-1), al final de cada grupo de variables entre renglón es llevada a cero, y de cada variable salen dos ramas a la capa de abajo una capa es punteada que indica una multiplicación por (-1) y la otra línea es continua que indica una multiplicación por (1), al final de la capa las salidas son llevadas a un uno, esto indica que las operaciones obtenidas son válidas figura 3.7.

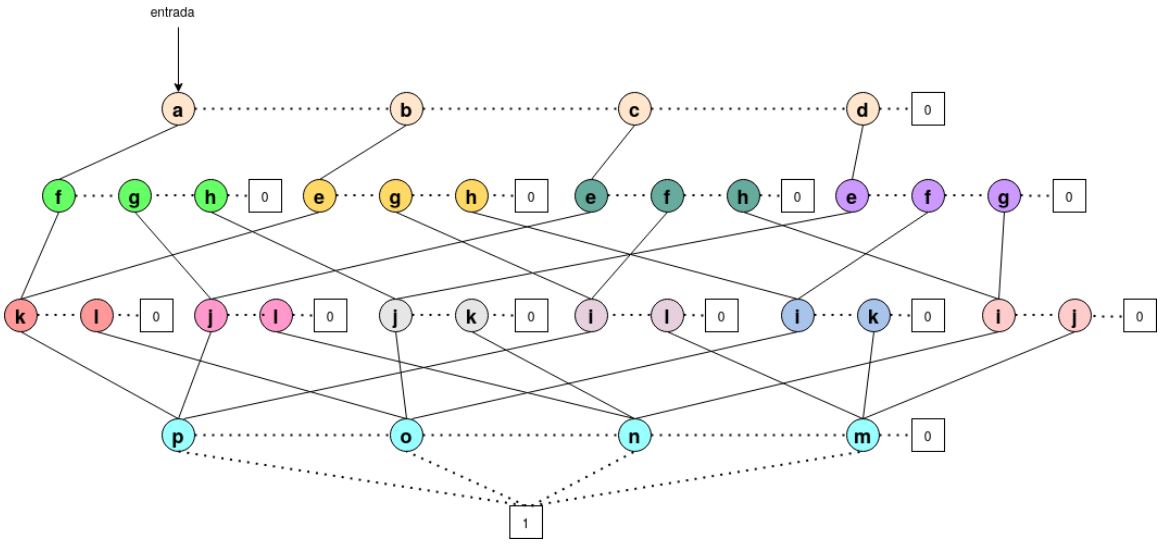


Figura 3.7: DDD por capas

La obtención de resultados del cofactor 'a' es mostrado en la figura 3.8

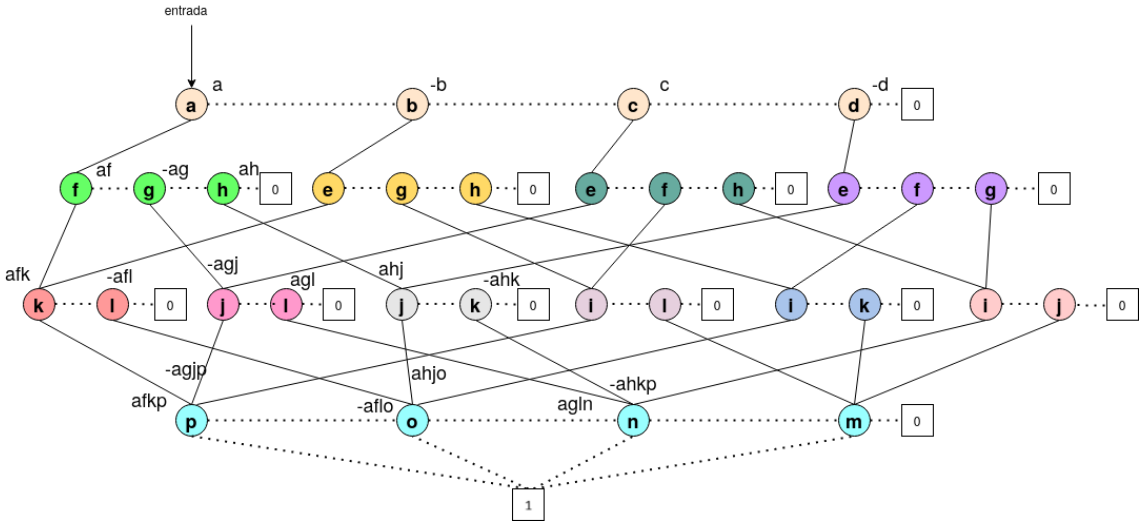


Figura 3.8: Resultados DDD cof a

Mientras que las ecuaciones por el cofactor 'd' se encuentra mediante:

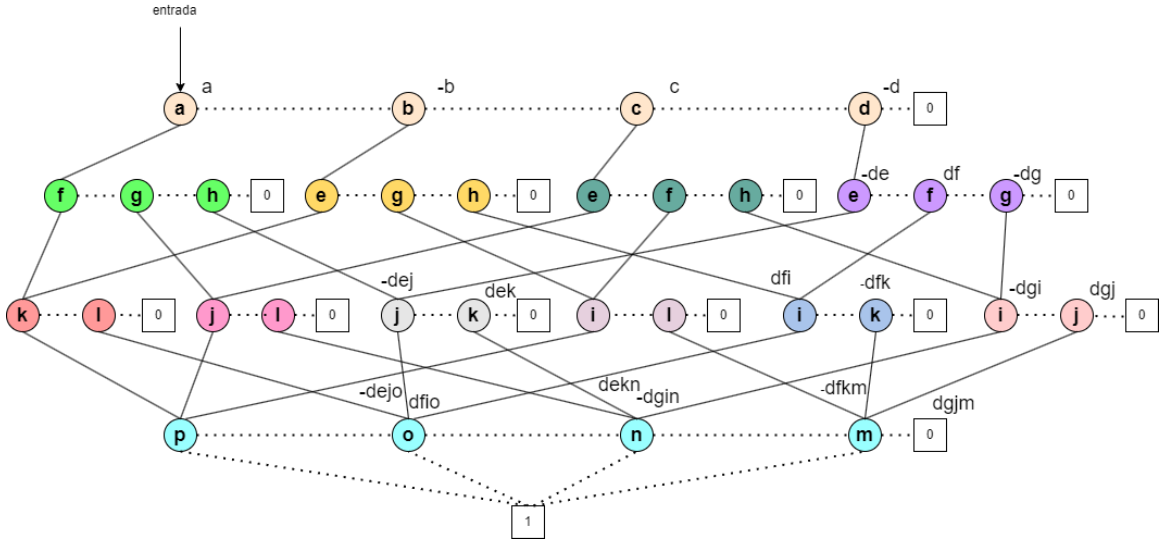


Figura 3.9: Resultados DDD cof d

Como puede observarse en la tabla 3.3, los cálculos en las capas última y ante penúltima se comparten. Algunos resultados experimentales presentados por Tan [20] escritos en C++ se observa un menor número de cálculos resultantes entre un DDD por capas de un DDD de Greedy.

Tamaño de la matriz	—DDD—Renglon	—DDD—Greedy
2	4	4
3	12	13
4	32	40
5	80	118
6	192	340
7	448	962
8	1,204	2,708
9	2,304	7,535
10	5,120	20,828
11	11,264	57,266
12	24,576	156,764
13	53,248	427,571
14	114,688	1,162,580
15	245,760	3,152,681
16	524,288	8,529,668
17	1,114,112	23,030,492
18	2,359,296	62,072,002

Cuadro 3.3: Número de cálculos de un DDD por capas vs un DDD de Greedy

3.3.3. Implementación

La implementación del algoritmo para resolver el sistema de ecuaciones simbólicas lineales de acuerdo con la metodología por el DDD de Greedy y el DDD por capas. Lo más conveniente es usar el método de solución DDD por capas porque requiere menor cantidad de operaciones.

Uno de los primeros pasos para la implementación del algoritmo es trazar un mapa general sobre lo que se pretende hacer, puesto que la idea del DDD por capas está bien documentada como idea con grafos con el procedimiento, pero no se tiene hasta ahora documentado un algoritmo computacional disponible para implementar en Python.

A continuación se la figura 3.10 el planteamiento del algoritmo DDD por capas, en donde dos partes importantes en el diseño del algoritmo son: trayectoria y recursividad, definiendo trayectoria como el primer camino que hace el algoritmo por vez primera, en esta se van creando las capas y todo lo necesario para que el algoritmo pueda tener una recursividad sobre la trayectoria generada. En la trayectoria se contemplan en todo momento el ir generando los datos de la multiplicación del determinante, y la trayectoria resulta en una subrutina que corre en la diagonal de la matriz, todos los datos de objetos son generados si cualquiera de los símbolos objeto son diferentes de cero

Una vez trazada la trayectoria se empieza con la recursividad del algoritmo, ubicando una subrutina que corra de la capa n hasta la capa cero siempre y cuando tenga todas las condiciones para correr entre capa y capa de mayor a menor, ya que de otro modo volverá a bajar

naturalmente a la capa n y en su trayecto generando todas las operaciones simbólicas contemplando en todo momento la consulta y reciclaje de operaciones simbólicas del diccionario uno.

Esta idea de ir reciclando operaciones simbólicas en el diccionario se hace haciendo una predicción en dos variables de tipo *string* concatenando caracteres de los próximos objetos símbolos que se van a multiplicar, y se toman dos predicciones contemplando que la multiplicación puede ser contemplando los objetos simbólicos a y b estos puedan ser multiplicados como: "$a*b$" o "$b*a$".

También de acuerdo a los recursos de Python es conveniente usar diccionarios para almacenar las operaciones simbólicas adjunto a un nombre es decir suponiendo que se tiene la operación simbólica: $a*b$ la forma de guardar la información en un diccionario contemplando su nombre en un tipo de dato *string* adjunto al objeto simbólico, es: ["$a*b$" : $a*b$] esta estructura resulta conveniente en los algoritmos de búsqueda porque si el resultado de cualquiera de las predicciones es: "$a*b$" entonces se pregunta al diccionario si contiene algún dato con dicho nombre, si el resultado es afirmativo. Es decir, diferente de la palabra reservada *None* entonces se usa esa operación evitando computar la operación.

Esta técnica resulta benéfica, ya que suponiendo que se tiene una perdición: $(a**(-1)+b*$ $*(-1))*c**(-1)$ el resultado de computar esta expresión es operar todo el conjunto de objetos simbólicos esta tarea en cientos de miles de operaciones que se encuentran en una matriz resulta en un tiempo muy extenso, es por eso que métodos numéricos bien definidos y bien definidos que trabajan perfectamente como: L-U o Gauss Jordan resultan en tiempos extensos por la acumulación de nuevos símbolos en cada operación. Sea la matriz a de la figura 3.10

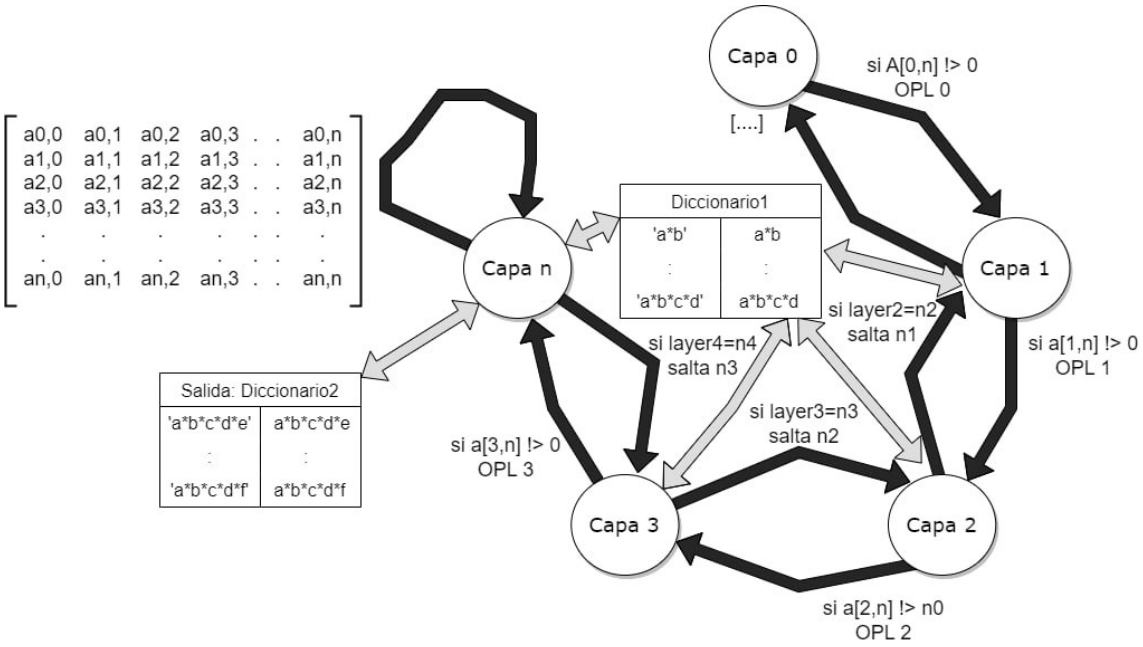


Figura 3.10: Planteamiento DDD por capas

Continuando con la idea y aterrizando mejor el concepto DDD por capas se define el siguiente grafo 3.11 es de gran ayuda para seguir definiendo el funcionamiento general del DDD por capas, en donde se subdividen cada trayectoria en un proceso que más adelante puede ser procesado en serie o en paralelo según sea el tamaño de la matriz y el hardware en el que se corra el algoritmo.

La idea es simple, retomando el grafo de la figura 3.10 esta idea es divida para cada elemento del renglón primario de la matriz, con esta idea se puede dividir un algoritmo extenso de búsqueda en bloques más pequeños que se pueden mandar a computar de diferentes maneras.

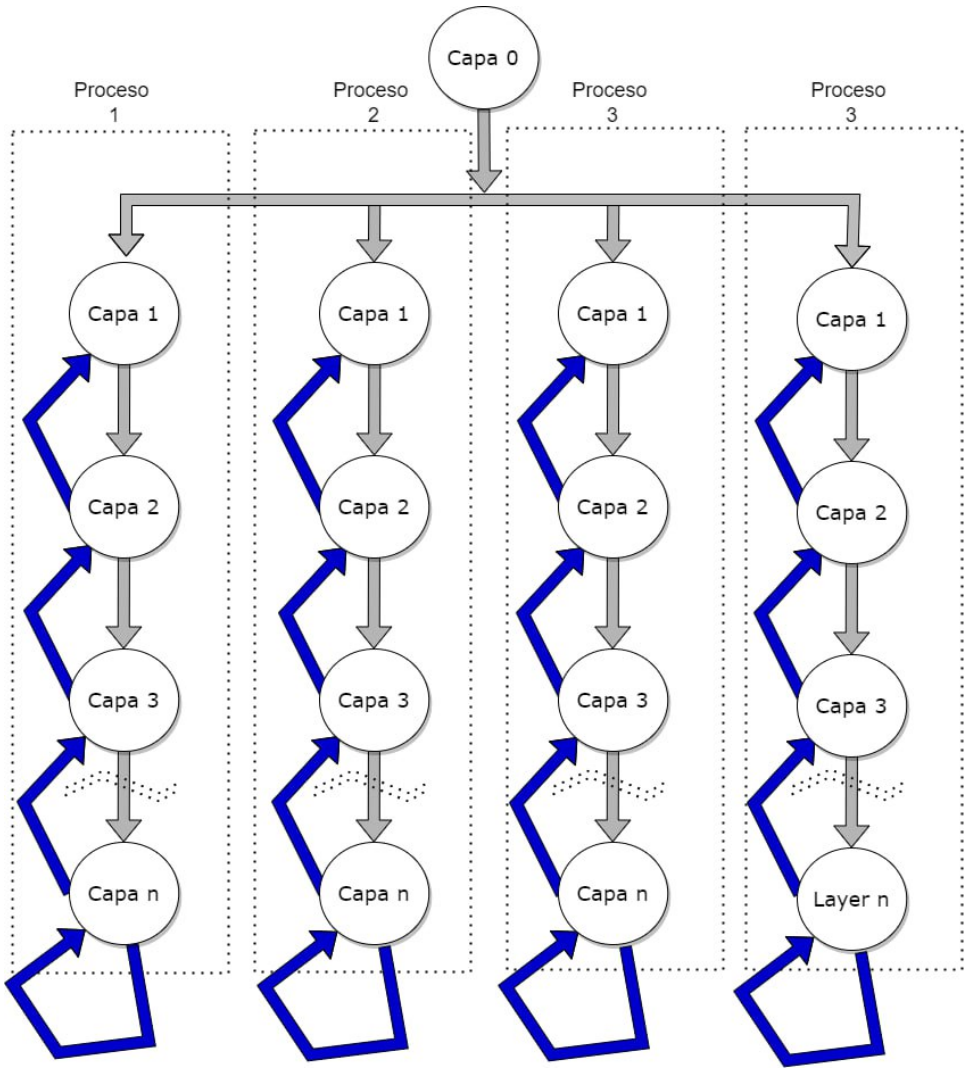


Figura 3.11: Planteamiento DDD por hilos

Un Panorama general sobre el algoritmo está dado en el diagrama de la figura 3.12 en donde para cumplir con el algoritmo es importante alimentar el inicio del algoritmo con la matriz A devuelta de la formulación MNA, o de manera general con alguna matriz simbólica, seguido del vector de excitaciones z.

El segundo paso es generar todos los procesos necesarios en donde se puede mandar a procesar de forma serial o paralela.

De acuerdo con la regla de Cramer, se preparan más matrices resultando en n+1 matrices resultantes que son almacenadas en un arreglo de la siguiente forma: $[[], [], .., []]$. A este paso se le llama formulación de determinantes a calcular.

El último paso es pedir un modo de operación, dicho modo de operación puede ser serie, paralelo o mixto.

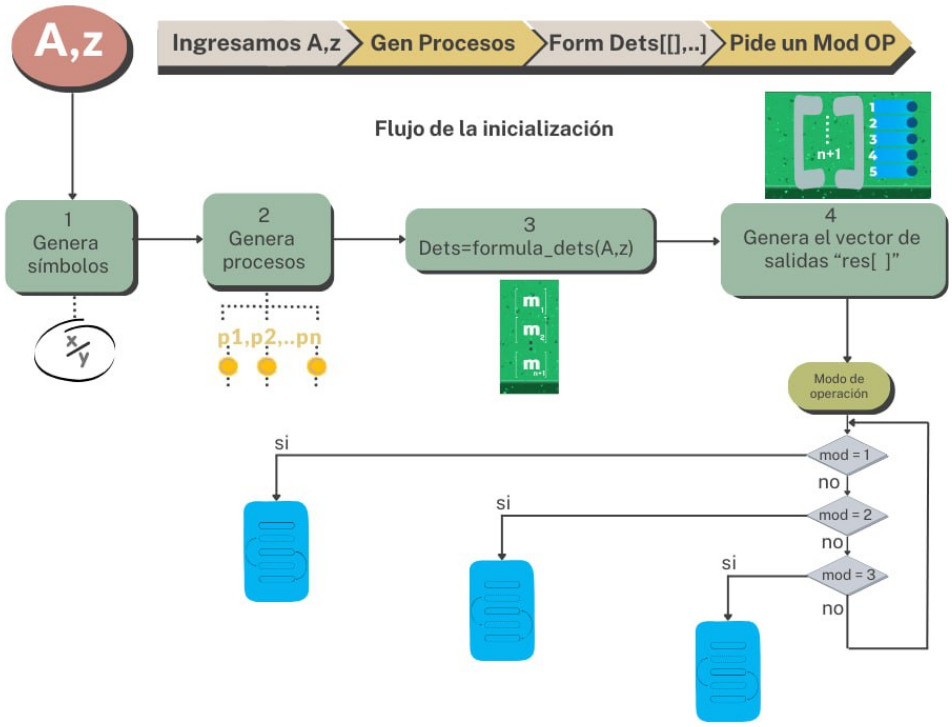


Figura 3.12: Planteamiento DDD panorama general

Cada método de operación ejecuta el algoritmo compuesto de todas las subrutinas que a continuación se mostraran, este algoritmo computacional está diseñado y adaptado en sintaxis Python:

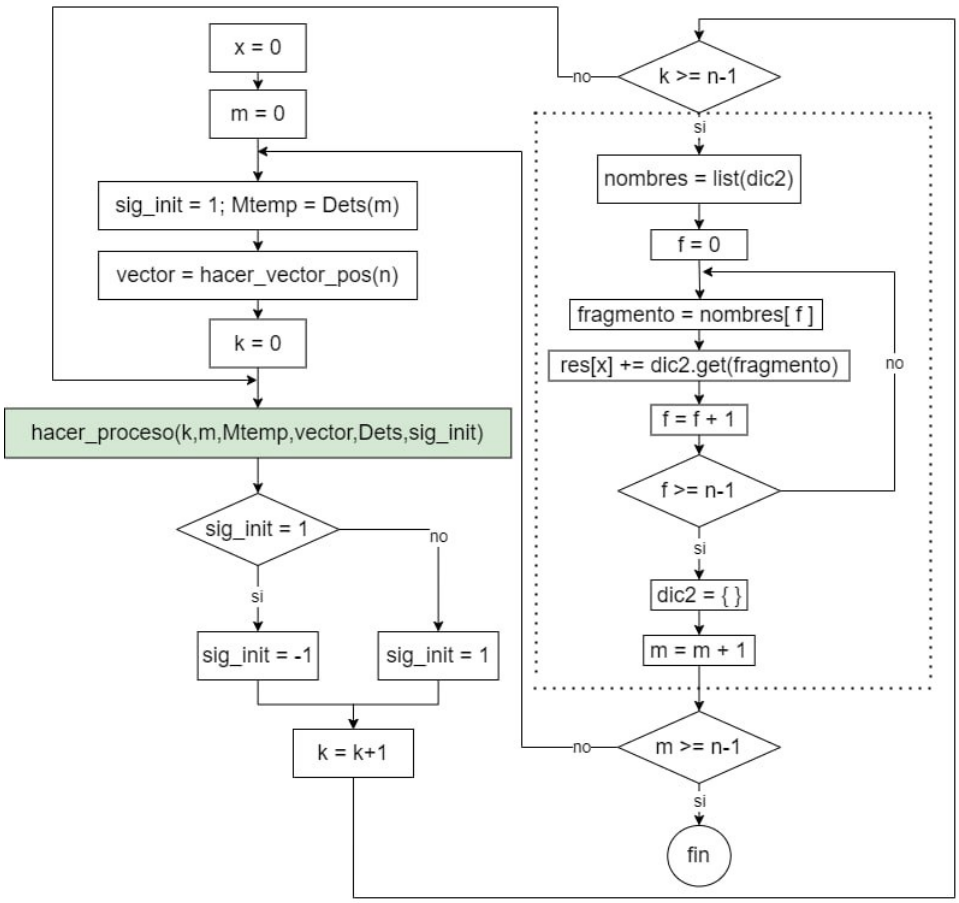


Figura 3.13: Iniciación del algoritmo

El diagrama de flujo mostrado en la figura 3.13 muestra el arranque del algoritmo en el modo uno de operación que consiste en dos ciclos que ejecutan el contenido del algoritmo en la instrucción $hacer_proceso$ resaltado en color verde olivo, algo importante a considerar es el signo que arranca en "1" y en cada iteración el signo es cambiado entre: $1 --> -1$ y de: $-1 --> 1$ y es definido como: sig_init.

El ciclo remarcado en líneas punteadas es usado para vaciar el diccionario e ir almacenando su contenido en un vector de resultados que más tarde es evaluado con la regla de Cramer, dicho vector es definido como $res[]$.

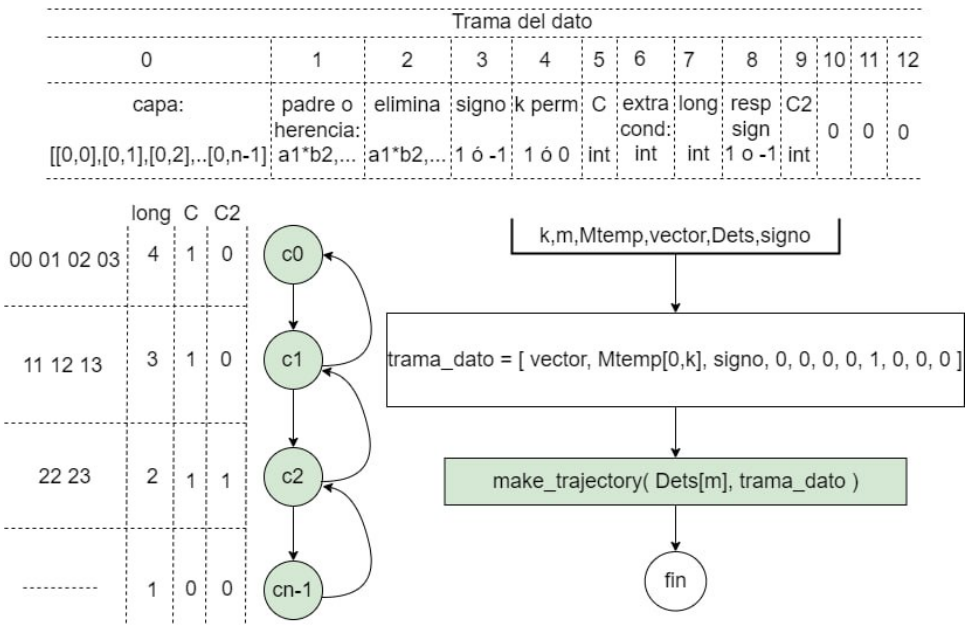


Figura 3.14: Subrutina: modo de operación

En la figura 3.14 se muestra el contenido de la función $hacer_proceso$ con datos de entrada k provenientes del contador del ciclo definido en la función modo de operación y es importante para correr el elemento de búsqueda definido en el grafo de la figura 3.11 y que sirve para arrancar cada proceso, la entrada m refiere al número de matriz recordando que estas se encuentran almacenadas en como los futuros determinantes a resolver de $n+1$ número de matrices de acuerdo con el tamaño nxn de la matriz.

La entrada $Mtemp$ contiene el determinante a resolver, el dato de entrada: $vector$ contiene la siguiente estructura de tamaño n: $[[0,0],[0,1].,...,[0,n]]$ y que define la capa uno inicial de arranque.

En la figura 3.14 se define la trama de datos con la que el algoritmo trabaja y que son consultadas y devueltas en la mayoría de subrutinas que se presentan más adelante, también se encuentra un esquemático de las capas y del contenido de las capas iniciando en la capa $C0$ hasta $Cn-1$ en donde en cada capa el contenido de cada capa es considerado como el primer renglón de cada cofactor y como se observa en el primer inicio está corriendo a través de la matriz diagonal.

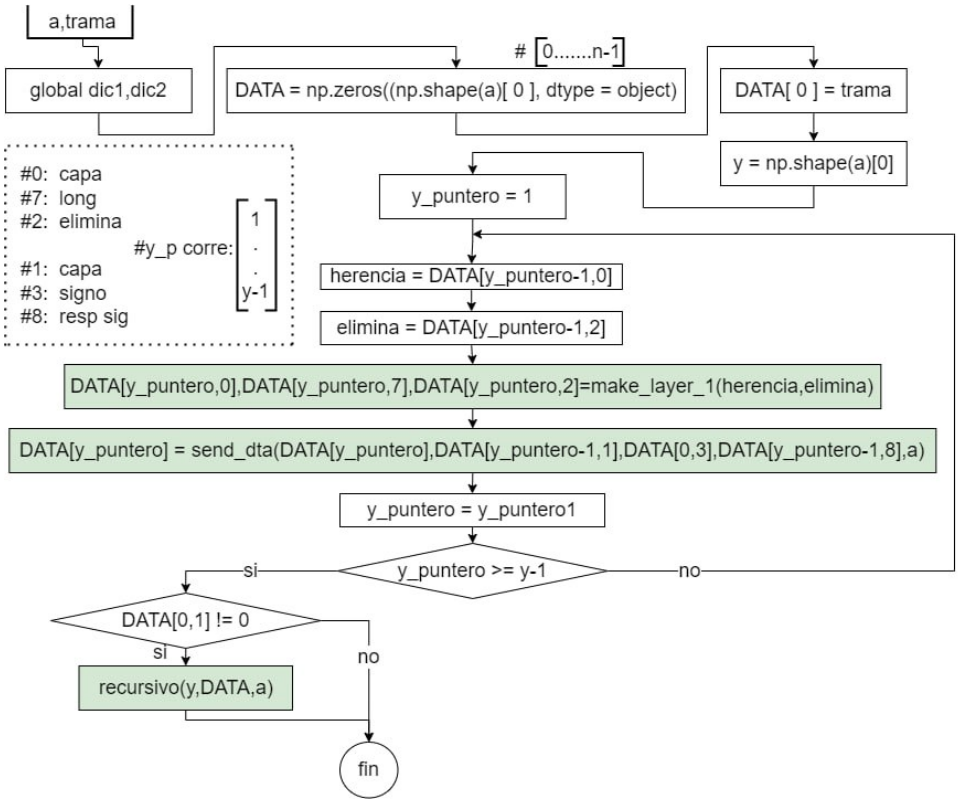


Figura 3.15: Subrutina: Haz la trayectoria

La subrutina contenida en el diagrama de flujo de la figura 3.15 muestra el proceso de trazar la trayectoria por primera vez para cada proceso definido en el grafo de la figura 3.11 esta subrutina trabaja con los datos de entrada a que contiene una matriz cuadrada y la trama de datos.

En el inicio de la subrutina se genera un espacio definido como: $DATA$ de n columnas por 1 para guardar el contenido de la trama de datos para cada capa teniendo en cuenta que para un tamaño n Python correrá siempre de: $0, 1, ..., n-1$ entonces la trayectoria correrá siempre de 1 hasta $y-1$ siendo y un escalar correspondiente al tamaño n.

En el corrimiento de $1, .., y-1$ se ejecutan tres subrutinas importantes resaltadas en verde olivo que a continuación se mostraran detalladamente, pero que comprenden la columna vertebral del algoritmo DDD.

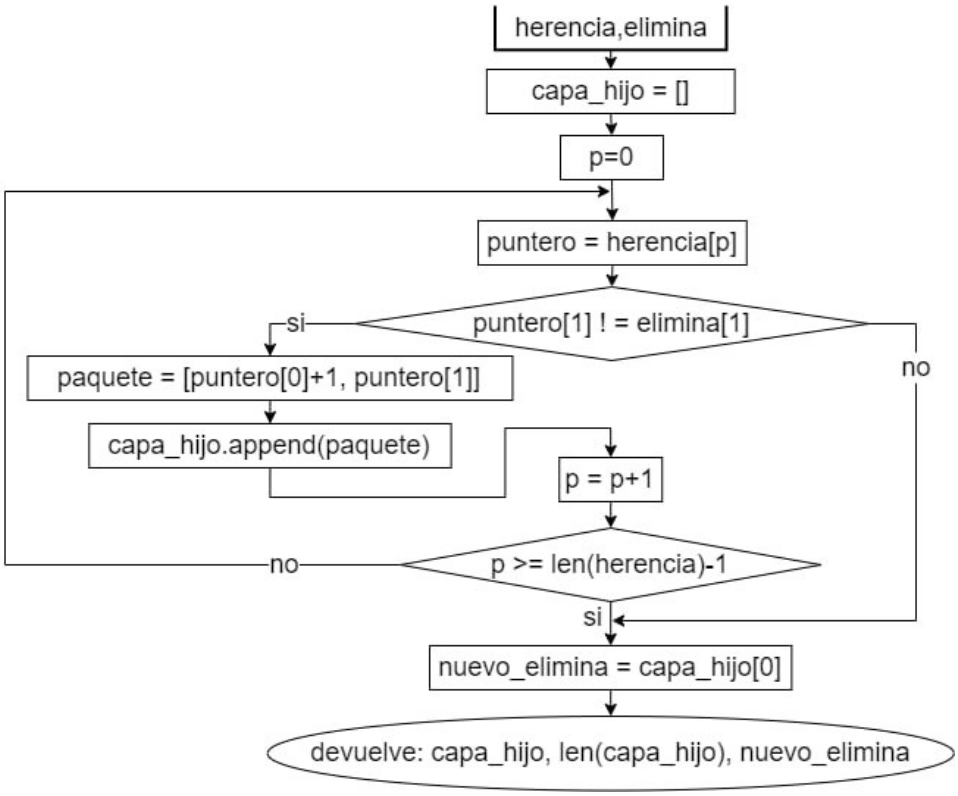


Figura 3.16: Subrutina: Haz la capa

El diagrama de flujo contenido en la figura 3.16 explica la subrutina encargada de crear cada capa nueva y es una parte importante en la subrutina de trayectoria como en recursividad, esta subrutina trabaja con dos elementos de entrada: el primero es la herencia de posiciones definida como *herencia* y el segundo es el dato a eliminar de la capa, definido como *elimina*

Esta subrutina devuelve la nueva capa definida como *capa hijo* también devuelve la longitud de la capa y el nuevo dato a eliminar definido como: *nuevo elimina*.

A continuación se presenta el algoritmo que se encarga de procesar los datos, esta subrutina se encarga de decidir si manda hacer las operaciones simbólicas o serán omitidas, también se encarga de revisar el estatus de la capa recordando el grafo de la figura 3.10 en las capas intermedias se hace una multiplicación entre el elemento padre y el elemento hijo mientras que para la última capa se hace el cálculo final de la capa.

El manejo de información es diferente para capas intermedias que para las últimas dos capas, ya que en las últimas capas $n-1$ y $n-2$ recordando que en un sistema de nxn Python corre de $0..n-1$

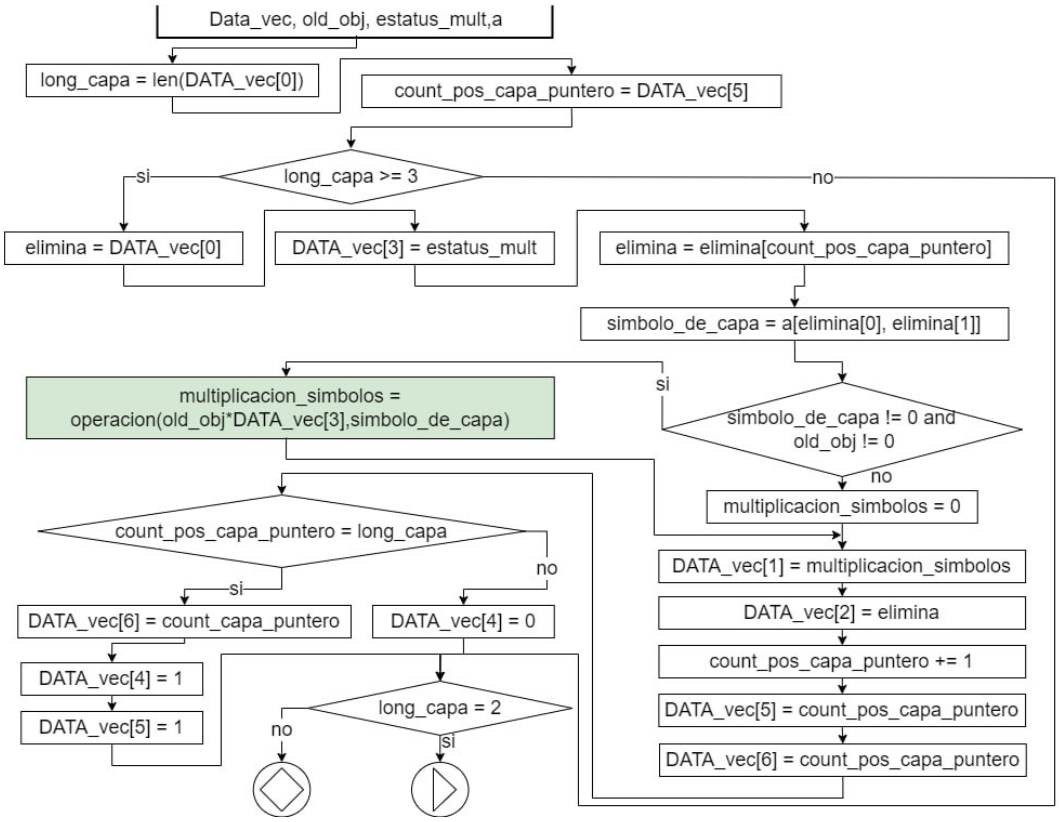


Figura 3.17: Subrutina: Manejo de datos parte 1

El algoritmo de la figura 3.17 muestra la parte uno de la subrutina $send_data$ del código anexado y es parte de la columna vertebral del algoritmo DDD. Esta subrutina inicia pidiendo cuatro datos de entrada, el primero definido como: *Data_vec* es toda la trama de dato perteneciente a la capa, esta trama es definida en diagrama de la figura 3.13.

El segundo dato que pide esta subrutina es el dato simbólico de la capa anterior definido como: *old_obj* y que sirve para analizar si este es diferente de cero y tomar la decisión de mandarlo a multiplicar o salir del ciclo.

El tercer dato a contemplar es estatus de la multiplicación que refiere a un estado positivo o negativo representado por: $1, -1$ y que es definido como: *estatus_mult*, finalmente el último dato de entrada es la matriz simbólica con la que se está trabajando esta matriz viene directamente de la entrada definida en el panorama general que es definido por el diagrama de la figura 3.12.

El panorama general de la subrutina es identificar la longitud de la capa actual, si la longitud es mayor o igual a tres realizará una toma de decisión que efectúa el cálculo intermedio entre capas, pero si es igual a dos entonces efectúa el cálculo final de la capa.

El cálculo intermedio de la capa consiste en ordenar la información necesaria para obtener el objeto simbólico de la matriz y hacer una previa evaluación de sí se mandará a multiplicar o no, dicha multiplicación es efectuada por la función: *multiplicacion_simbolos*. El resultado de la evaluación de la longitud en que si es iguala a dos o no es llevada a los conectores romboide y triángulo.

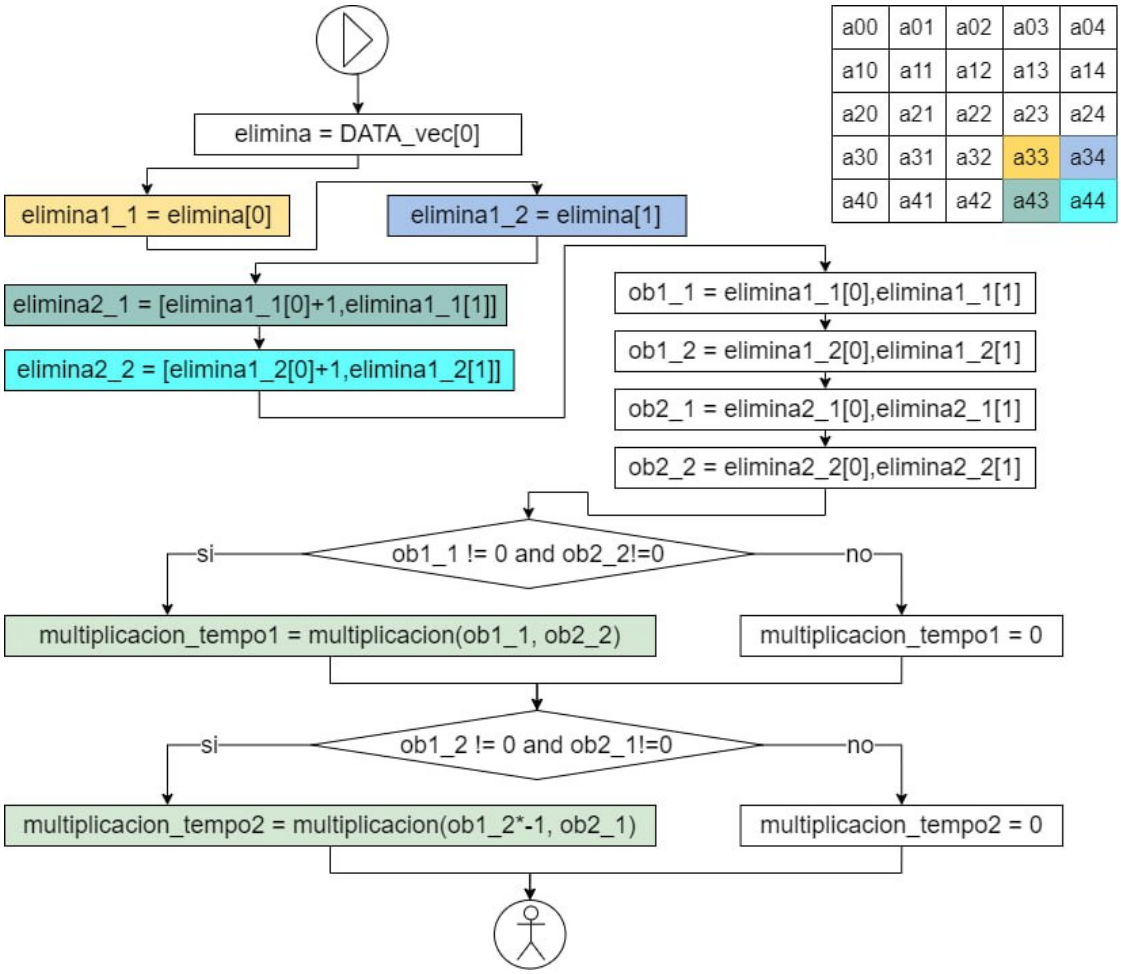


Figura 3.18: Subrutina: Manejo de datos parte 2

El algoritmo mostrado en la figura 3.18 es la continuación de la subrutina iniciada en el diagrama de la figura 3.17 y refiere al caso en el que la longitud de la capa es igual a dos, en ese caso se encuentra en la parte final de la capa y el cálculo que se hace en esta parte refiere al determinante: $2x2$ de la matriz de entrada, este algoritmo una consultas en el diccionario 1 al igual que la condición mayor o igual a tres, pero se contemplan dos operaciones cruzadas contemplando el signo y además la evaluación en la que el dato simbólico sea diferente de cero.

Si la evaluación es diferente de cero, entonces se hacen las multiplicaciones cruzadas, siempre teniendo en cuenta la posibilidad de reciclar operaciones que se encuentran en el diccionario uno y almacenando el resultado final en el diccionario dos.

A continuación se muestra la parte tres y final de la subrutina *sen_dta* esta parte final contempla el cálculo final del determinante cruzado, y los estados del número en el que se encuentra la capa el signo y el número de operación en el que se encuentra la capa, estas condiciones son tomadas en la recursividad para una evaluación de continuar en la capa o mandar a una capa arriba.

Finalmente, se devuelve la trama de dato completa y se termina la subrutina.

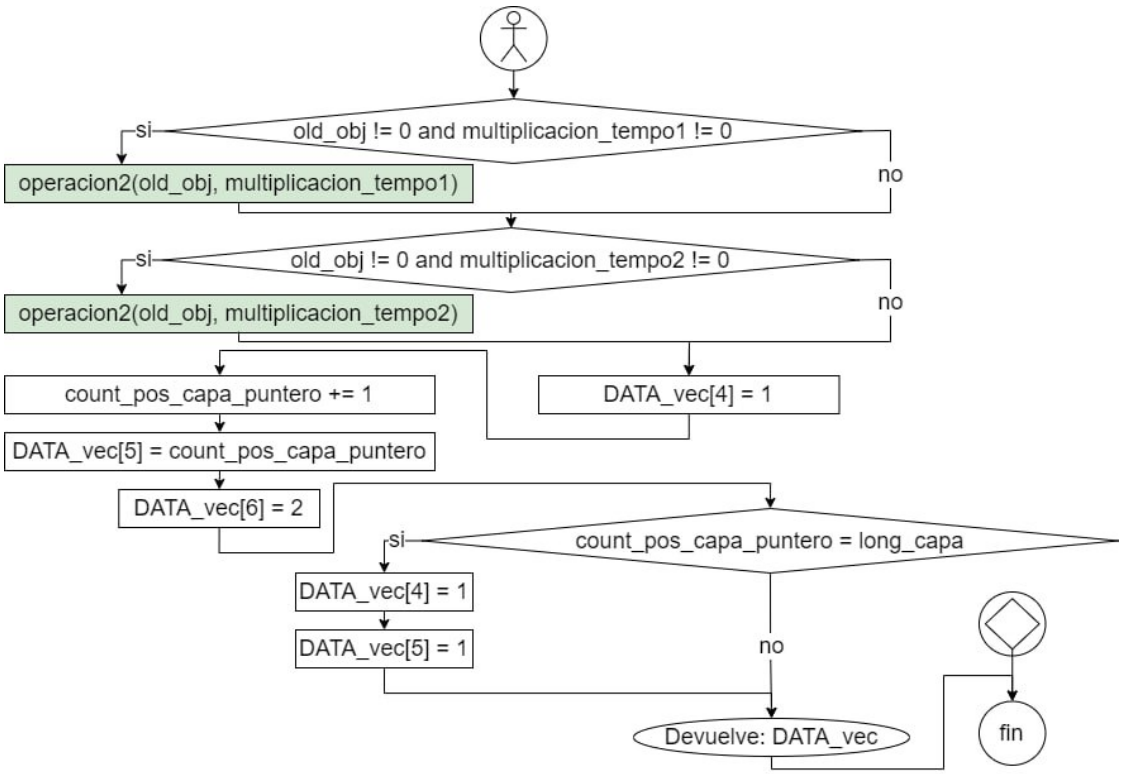


Figura 3.19: Subrutina: Manejo de datos parte 3

A continuación se muestran las operaciones que realiza el algoritmo DDD que como se vio en la subrutina *send_dta* pueden ser de dos tipos: uno al estar en medio de las capas identificándola como longitud de capa mayor o igual a tres y dos en la capa última identificándola por una longitud igualada dos.

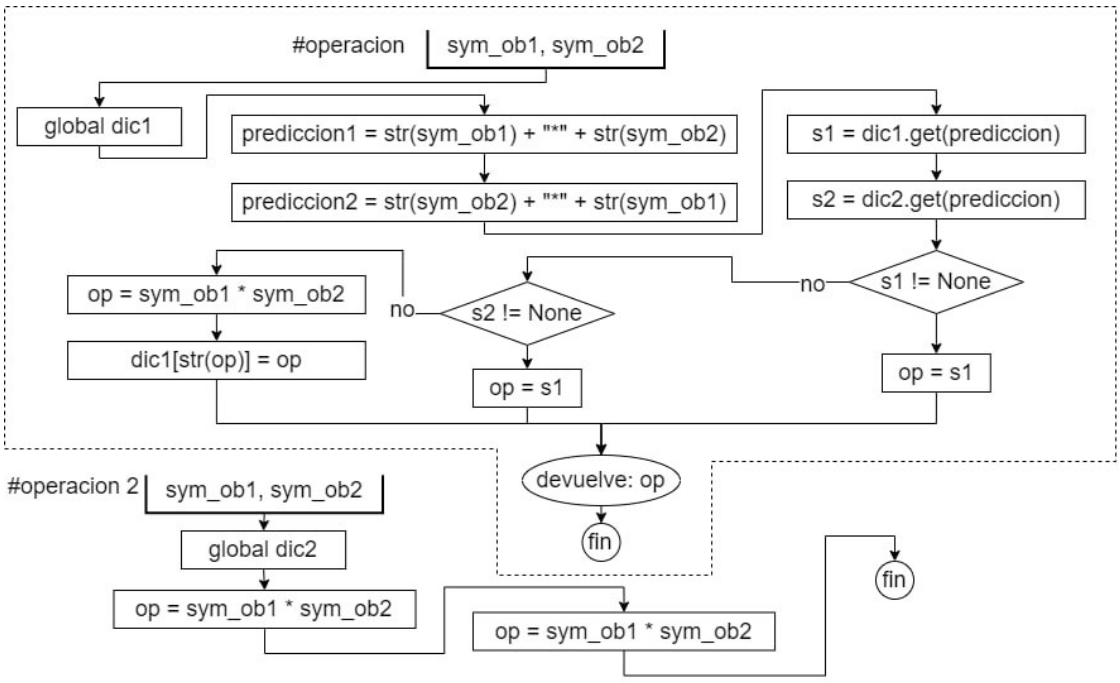


Figura 3.20: Subrutina: Operaciones

El diagrama presentado en la figura 3.20 muestra dos subrutinas que contienen el modo de operación entre capas definido como: *operacion* y el modo de operación para la última capa definido como: *operacion*2

En la subrutina *operacion* y *operacion*2 solo requieren dos datos de entrada definidos como: *sym_ob*1 y *sym_ob*2 la diferencia entre ambas subrutinas es la inspección en el diccionario 1 para la subrutina *operacion* en la que hace una predicción previa para buscar en el diccionario esta predicción es doble y en caso de que cualquiera de las dos predicciones sea verdadera se recicla la operación simbólica.

De lo contrario, dicha operación es calculada por primera vez y almacenada en el diccionario 1 para una futura consulta.

Por otro lado, la subrutina: *operacion*2 es usada para la última capa del proceso tanto en trayectoria como en recursividad y el objetivo de esta subrutina es hacer una multiplicación final de los objetos simbólicos y devolverla al diccionario 2.

A continuación se presenta otra subrutina que forma parte de la columna vertebral del algoritmo DDD. Esta es definida como: *recursivo* y es fundamental en la obtención de resultados simbólicos.

Esta subrutina es definida en la figura 3.21 el inicio de esta subrutina es definido por los datos de entrada definidos como: y y $DATA$ y a, se encuentra dividido en tres secciones.

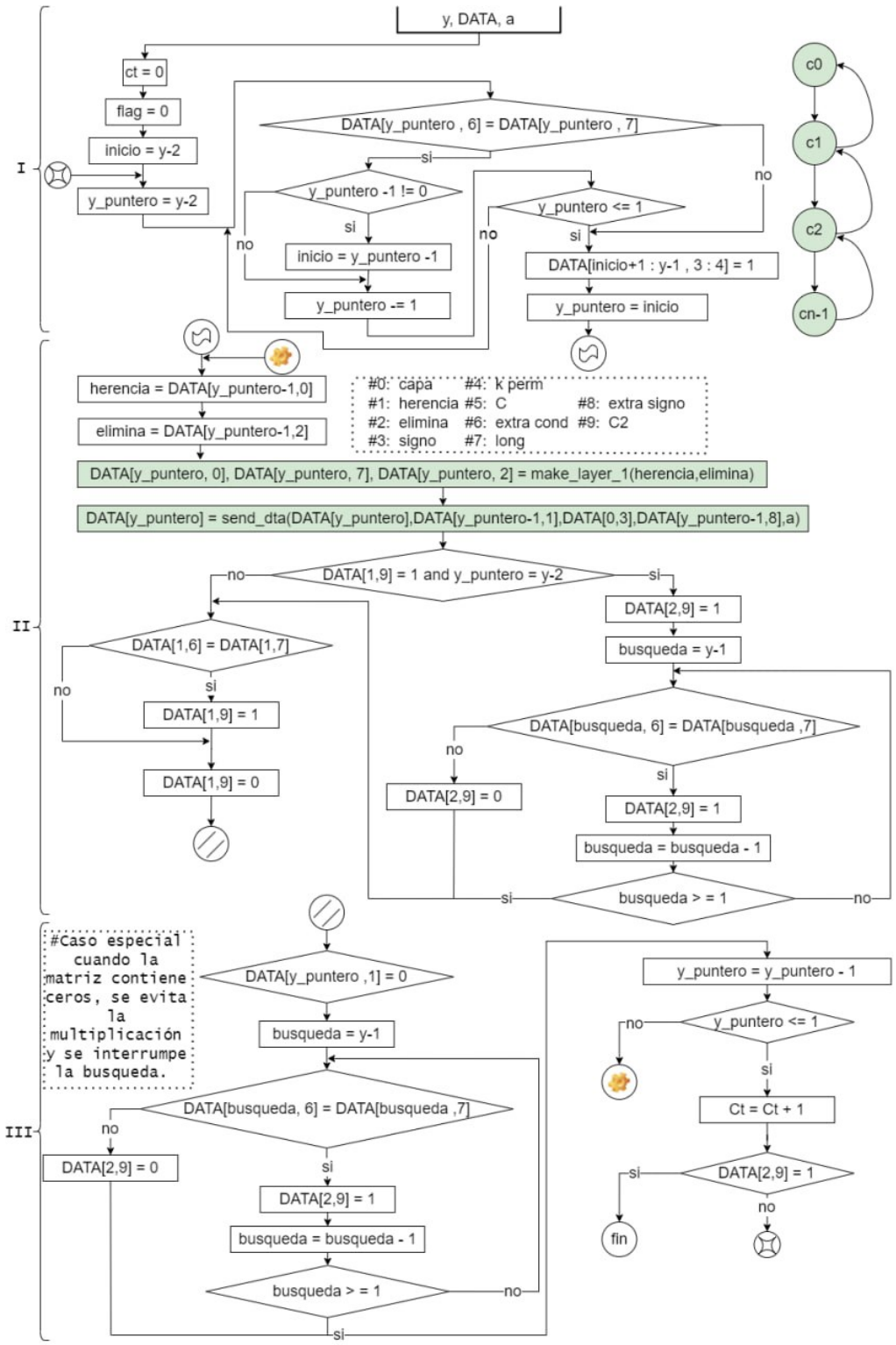


Figura 3.21: Subrutina: Recursividad parte 1, 2 y 3

En la parte I de este diagrama inicia pidiendo los datos de entrada y de acuerdo al diagrama verde olivo de las capas: $C0$ $C1$ $C2$ y $Cn-1$ esta primera parte se encarga de buscar de abajo hacia arriba, puede iniciar en cualquier capa entre: $y-2$ que es el equivalente a: $cn-1$ de acuerdo con la estructura planteada, y corre de mayor a menor siempre que sea mayor o igual que uno.

La primera parte establece el punto de inicio y compara una igualdad entre $DATA[y\_puntero, 6]$ y $DATA[y\_puntero, 7]$ definidos en la trama de datos como: *extra_condicion* y *longitud_capa* si esta condición es verdadera entonces se compra que el puntero sea diferente de cero, ya que la capa cero es definida como cada proceso a trabajar en serie o paralelo como se muestra en el grafo de la figura 3.11.

Al romper el ciclo se guarda el inicio de la recursividad y se hace una puesta en uno para todo valor mayor a inicio hasta $y-1$ pala todos valor definido en la trama de datos como: *signo* de esta manera se obtiene el inicio para el ciclo de la sección II y un reinicio en los cambios de signo.

La parte dos del algoritmo es la actualización de las capas y las herencias, con cada nuevo cálculo se requiere usar nuevos datos que son dados por una actualización de la capa, y almacenadas en el espacio en: $DATA[y\_capa]$ después se manda hacer todo el cálculo a la subrutina *send_dta*, esta subrutina alimentada correctamente es la encargada de hacer todo el cálculo simbólico mandando a traer las subrutinas de operación

Después se compara una condición de igualdad entre el dato definido de la trama de datos como: $C2$ con un uno y *y_puntero* igual con $y-2$, ya que esta comparación aplica a la última capa y a la condición $C2$ para romper el ciclo indefinido haciendo una puesta en uno en: $DATA[2, 9]$ esta condición es puesta de esta manera para no tener una perdida de datos en la última capa en el último ciclo y asegurar el rompimiento del ciclo en la última capa en el último cálculo.

La parte III y última de la subrutina de recursividad es una condición especial puesta para romper el ciclo de búsqueda y operación si se encuentra un cero, ya que la formulación MNA es resultante en matrices con ceros mayormente en el triángulo inferior y superior de los elementos pasivos para ello se inicia una comparación del elemento simbólico definido en la trama de datos como: padre o herencia y asignado en: DATA[y,1] igual a cero si esta comparación es verdadera se prepara todo lo necesario para salir replicando la salida de la parte II.

Finalmente, en la parte III de la subrutina de expresividad se pregunta en cada iteración de cálculo preguntando si el espacio designado como $DATA[2, 9]$ es igual con uno, entonces rompe el ciclo y se devuelven todos los resultados simbólicos.

Algunas subrutinas adicionales fueron añadidas para este algoritmo DDD en la práctica se observó que si se ordena lo mejor posible todos los elementos lo más cerca de la diagonal el algoritmo de recursividad tardará mucho menos en converger a la solución incluso en un 50 % más rápido es por esto que se prepara una subrutina que acerca los valores a la diagonal y cuando la diagonal resulta en tener algún cero es un hecho que el determinante de esa matriz es cero y se prepara para evitar buscar una solución.

Otra subrutina adicional es la de preparar todos los determinantes para llevarlos al DDD, consiste en copiar la matriz n veces en un sistema de nxn y de acuerdo con la regla de Cramer reemplazar el vector de entradas: z para cada renglón.

Otra subrutina adicional es la que prepara la división en cada determinante de acuerdo con la regla de Cramer, ya que DDD está pensado solo para resolver determinantes simbólicos.

3.4. Conclusiones

Es crucial entender las limitaciones de los métodos numéricos tradicionales al abordar la convergencia de resultados simbólicos. Los métodos numéricos, aunque efectivos para obtener soluciones aproximadas, a menudo requieren una alta precisión y un gran número de iteraciones, lo que puede conducir a errores acumulativos y un mayor consumo de recursos computacionales. En contraste, los métodos simbólicos que utilizan la memoria para predecir futuras operaciones simbólicas evitando computar operaciones redundantes.

Los resultados obtenidos con este algoritmo confirman su eficacia como una herramienta de solución simbólica en Python, permitiendo no solo una reducción significativa en el uso de recursos computacionales, sino también una mayor precisión en los resultados. Además, el ahorro computacional logrado contribuye a mejorar la eficiencia en el procesamiento de datos, haciendo que este enfoque sea altamente beneficioso para la solución al sistema de ecuaciones devuelta por el algoritmo de parseo.

Se presenta un algoritmo claramente estructurado y definido, que no solo es implementable en Python sino también adaptable a otros lenguajes de programación. Esta flexibilidad permite realizar comparaciones entre diferentes plataformas y tecnologías, lo cual es invaluable para evaluar la eficiencia y eficacia del algoritmo en diversos entornos de desarrollo. Implementarlo en múltiples lenguajes puede facilitar su integración en diferentes sistemas y aplicaciones, ampliando su potencial impacto y utilidad en la solución a sistemas de ecuaciones lineales en general.

Capítulo 4

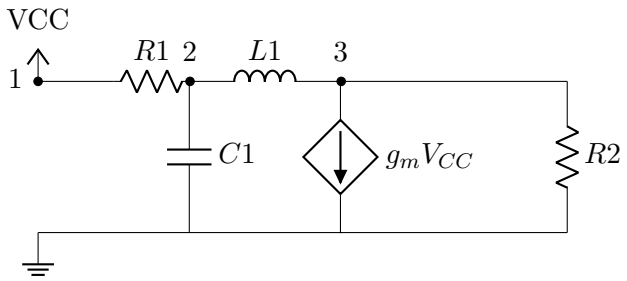
Resultados

A continuación se presentan los resultados obtenidos por el simulador simbólico SCAPy, empezando por la validación de resultados, ante dos circuitos propuestos el primero con una fuente de corriente controlada por voltaje y el segundo corresponde con el circuito que propone Gielen [25], haciendo una comparativa de resultados en el dominio de la frecuencia con ESCAM de MATLAB y la herramienta propuesta SCAPy, más delante se presenta la cantidad de memoria RAM y el tiempo de cómputo obtenido bajo diferentes matrices, para finalmente mostrar tres circuitos no convencionales correspondientes con el integrador fraccional, el filtro WTA LTA, y terminar con el memristor comparando en todos los resultados obtenidos con LTSPICE un simulador numérico con la finalidad de observar y discutir los resultados obtenidos.

4.1. Validación de resultados

Esta sección presenta la validación de resultados, es importante obtener resultados correctos en determinantes sin ceros en filas y o columnas de la matriz y obtener resultados correctos en determinantes que involucren marices con ceros en los elementos que compone la matriz como en la formulación MNA.

A continuación se presentan los resultados de las pruebas que se hicieron para validar los resultados entregados por el algoritmo DDD, se contemplaron dos circuitos y en el primero de ellos se hicieron tres comparaciones entre los resultados obtenidos por DDD, ESCAM y LTSPICE siendo los dos primeros resultados simbólicos que después se evaluaron con valores en los elementos eléctricos obteniendo resultados semi-simbólicos, y el ultimo con simulación numérica, a continuación el primer circuito:



Sistema de ecuaciones:

$$A = \begin{bmatrix} \frac{1}{R_1} & -\frac{1}{R_1} & 0 & 1 \\ -\frac{1}{R_1} & C_1S + \frac{1}{R_1} + \frac{1}{L_1S} & -\frac{1}{L_1S} & 0 \\ G_1 & -\frac{1}{L_1S} & \frac{1}{R_2} + \frac{1}{L_1S} & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$x = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ Iv_1 \end{bmatrix} \qquad z = \begin{bmatrix} 0 \\ 0 \\ 0 \\ V_1 \end{bmatrix}$$

Contemplando los siguientes valores para los elementos eléctricos:

```
R1 = 1e3; R2 = 1.5e3; L1 = 0.005; C1 = 1e-6;
V1 = 1.3; G1 = 0.8
```

Cuadro 4.1: Nodo 2

Python:	$\frac{623.22\sqrt{1.74045954049431 \cdot 10^{-17} + \frac{1}{\omega^2}}}{\sqrt{4.0 \cdot 10^{-18}\omega^2 + 3.58404 \cdot 10^{-7} + \frac{1}{\omega^2}}}$
MATLAB:	$\frac{623.22\sqrt{1.74045954049432 \cdot 10^{-17}\omega^2 + 1}}{\sqrt{4.0 \cdot 10^{-18}\omega^4 + 3.58404 \cdot 10^{-7}\omega^2 + 1}}$

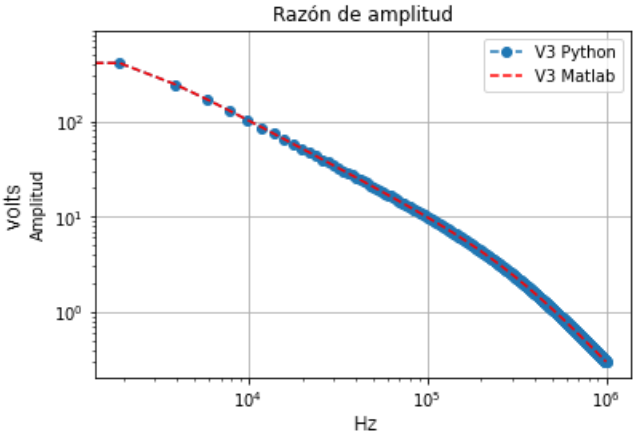


Figura 4.1: Frecuencia en nodo 2 semi-simbólico

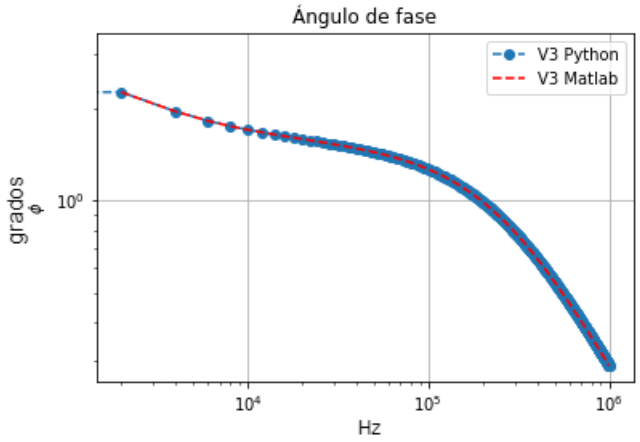


Figura 4.2: Ángulo de fase en nodo 2 semi-simbólico

Como puede observarse en las figuras 4.1 y 4.2, se muestran las gráficas correspondientes a la amplitud y al ángulo de fase. Los resultados obtenidos mediante SCAPy se presentan en color azul, mientras que los obtenidos mediante SCAM están en color rojo. Además de esta comparación entre SCAM y Python, se incluye una simulación numérica por LTspice de la amplitud y el ángulo de fase, representada en la figura 4.3. En esta simulación, se observa un comportamiento idéntico entre las distintas herramientas, lo cual permite la validación de los resultados para el nodo analizado.

Es importante señalar que tanto SCAM como LTspice son conocidos por ofrecer resultados muy confiables. Por tanto, obtener un resultado simbólico idéntico al de SCAM no solo valida la precisión de SCAPy, sino que también garantiza la confianza en los resultados obtenidos para el circuito. Esta validación contribuye significativamente a la certeza y exactitud en el análisis del nodo tres.

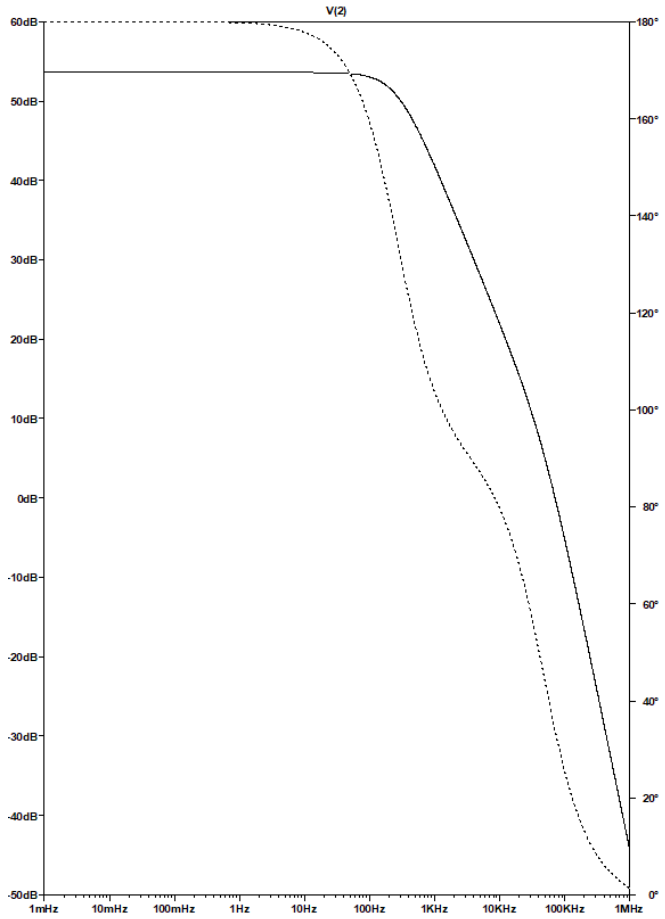


Figura 4.3: Simulación numérica por LTSPICE nodo 2

Cuadro 4.2: Nodo 2

Python:	$\frac{623.22\sqrt{2.50626173831181 \cdot 10^{-17}\omega^4 - 9.98745302717258 \cdot 10^{-9}\omega^2 + 1}}{\sqrt{4.0 \cdot 10^{-18}\omega^4 + 3.58404 \cdot 10^{-7}\omega^2 + 1}}$
MATLAB:	$\frac{623.22\sqrt{2.50626173831181 \cdot 10^{-17}\omega^4 - 9.98745302717258 \cdot 10^{-9}\omega^2 + 1}}{\sqrt{4.0 \cdot 10^{-18}\omega^4 + 3.58404 \cdot 10^{-7}\omega^2 + 1}}$

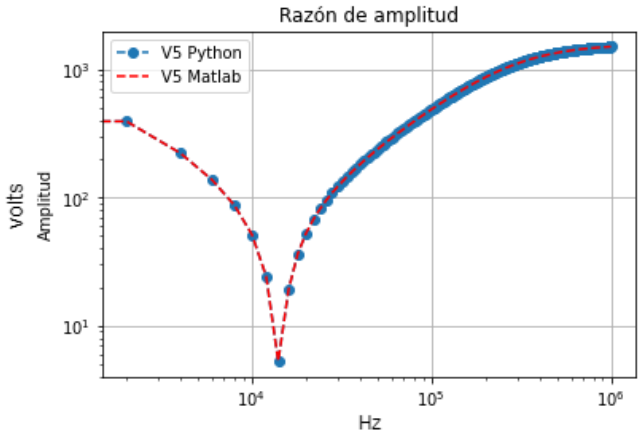


Figura 4.4: Frecuencia en nodo 3 semi-simbólico

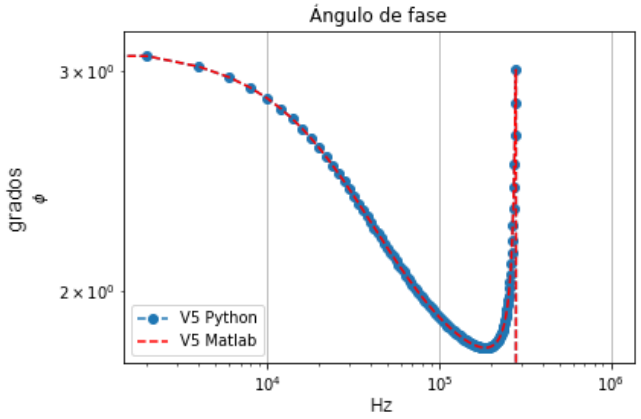


Figura 4.5: Ángulo de fase en nodo 3 semi-simbólico

En las figuras 4.4 y 4.5, se muestra el análisis del nodo número cinco, correspondiente a la amplitud y el ángulo de fase. La comparación entre los resultados obtenidos por SCAM y SCAPy resulta satisfactoria, ya que se observa una similitud exacta entre ellos. Esto demuestra la precisión de SCAPy en la reproducción de los cálculos simbólicos.

Una comparación adicional se puede observar en las ecuaciones presentadas en el cuadro 4.2, donde ambas salidas simbólicas son idénticas, reafirmando la consistencia entre los dos métodos de análisis. Además, la figura 4.6 ilustra una simulación numérica realizada en LTspice, que refuerza aún más los resultados obtenidos por SCAPy. Esta validación adicional confirma la exactitud de los resultados devueltos por este simulador, asegurando una validación correcta en el análisis de los circuitos estudiados.

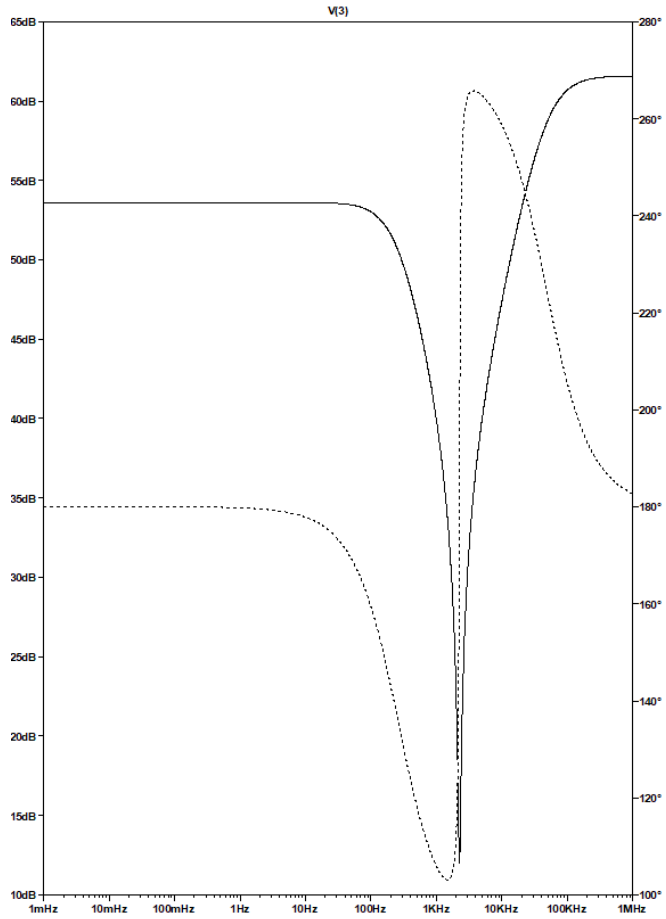
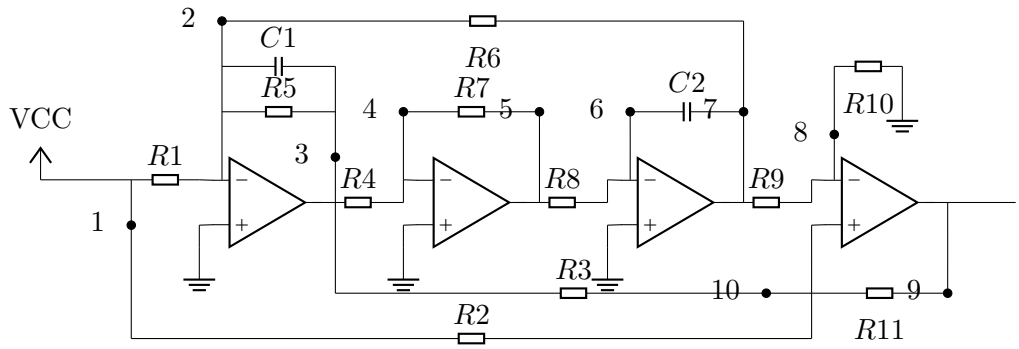


Figura 4.6: Simulación numérica por LTSPICE nodo 3

A continuación se presenta la validación con el filtro activo RC que propone Gielen [25], fijando los nodos de interés en el nodo 1, 9, 3, 5 y 7.



El sistema de ecuaciones que se obtiene es el siguiente:

$$A = \begin{bmatrix}
\frac{1}{R_2}+\frac{1}{R_1} & -\frac{1}{R_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{R_2} & 1 & 0 & 0 & 0 & 0 \\
-\frac{1}{R_1} & C_1S+\frac{1}{R_6}+\frac{1}{R_5}+\frac{1}{R_1} & -C_1S+\frac{1}{R_5} & 0 & 0 & 0 & -\frac{1}{R_6} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -C_1S+\frac{1}{R_5} & C_1S+\frac{1}{R_5}+\frac{1}{R_4}+\frac{1}{R_3} & -\frac{1}{R_4} & 0 & 0 & 0 & 0 & 0 & -\frac{1}{R_3} & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & -\frac{1}{R_4} & \frac{1}{R_7}+\frac{1}{R_4} & -\frac{1}{R_7} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -\frac{1}{R_7} & \frac{1}{R_8}+\frac{1}{R_7} & -\frac{1}{R_8} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & -\frac{1}{R_8} & C_2S+\frac{1}{R_8} & -C_2S & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -\frac{1}{R_6} & 0 & 0 & 0 & -C_2S & C_2S+\frac{1}{R_9}+\frac{1}{R_6} & -\frac{1}{R_9} & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & -\frac{1}{R_9} & \frac{1}{R_9}+\frac{1}{R_{10}} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{R_{11}} & -\frac{1}{R_{11}} & 0 & 0 & 0 & 0 & 1 \\
-\frac{1}{R_2} & 0 & -\frac{1}{R_3} & 0 & 0 & 0 & 0 & 0 & -\frac{1}{R_{11}} & \frac{1}{R_3}+\frac{1}{R_2}+\frac{1}{R_{11}} & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

$$x = \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \\ V_5 \\ V_6 \\ V_7 \\ V_8 \\ V_9 \\ V_{10} \\ Iv_1 \\ I_{OAmp1} \\ I_{OAmp2} \\ I_{OAmp3} \\ I_{OAmp4} \end{bmatrix} \qquad z = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ V_1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Después de evaluar el sistema de ecuaciones simbólico en el DDD implementado en Python se evalúa también en MATLAB, posteriormente se sustituyen los valores de R, C y V por los mostrados a continuación y se sustituye: s por $i\omega$.

```
R1 = 1e3;R2 = 1.5e3;R3 = 2.5e3;R4 = 2.7e3;R5 = 3.0e3;R6 = 3.3e3;
R7 = 3.6e3;R8 = 3.9e3;R9 = 4.1e3;R10 = 4.3e3;R11 = 4.6e3;R12 = 4.9e3;
R13 = 5.2e3;C1 = 1e-6;C2 = 1e-5;V1 = 10.5;
```

Posteriormente, se corre el valor absoluto de cada expresión semi-simbólica de $[1e{-}2, ..., 1e3]$ y se observa el comportamiento presentado a continuación:

Cuadro 4.3: Nodo 1

Python:	$\frac{10.5\sqrt{9.317075625\cdot10^{-9}\omega^4+0.000842180625\omega^2+1}}{\sqrt{9.317075625\cdot10^{-9}\omega^4+0.000842180625\omega^2+1}}$
MATLAB:	$\frac{10.5\sqrt{9.31707562500001\cdot10^{-9}\omega^4+0.0008421806250000001\omega^2+1}}{\sqrt{9.317075625\cdot10^{-9}\omega^4+0.000842180625\omega^2+1}}$

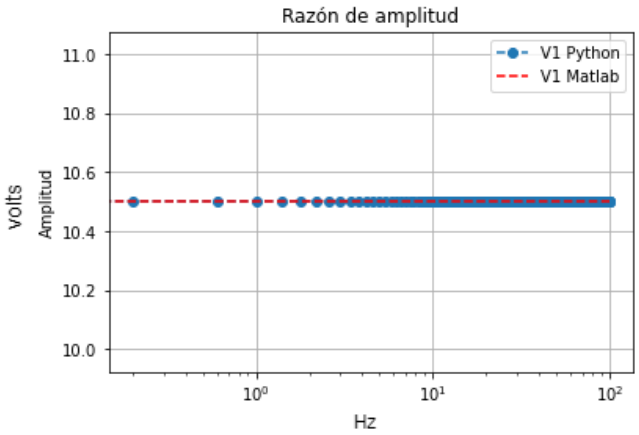


Figura 4.7: Frecuencia en nodo 1 semi-simbólico

La figura 4.7 muestra el resultado esperado, ya que la entrada de voltaje suministrada es constante e invariante en el tiempo, las ecuaciones del cuadro 4.3 donde se muestra la salida simbólica de SCAM y SCAPy son equivalentes y se puede reafirmar en el gráfico mostrado, donde ambos voltajes siguen la misma trayectoria.

Cuadro 4.4: Nodo 9

Python:	$\frac{136.9695\sqrt{5.14924579718105\cdot10^{-10}\omega^4-8.76704033307712\cdot10^{-6}\omega^2+1}}{\sqrt{9.317075625\cdot10^{-9}\omega^4+0.000842180625\omega^2+1}}$
MATLAB:	$\frac{136.9695\sqrt{5.14924579718105\cdot10^{-10}\omega^4-8.7670403330771\cdot10^{-6}\omega^2+1}}{\sqrt{9.317075625\cdot10^{-9}\omega^4+0.000842180625\omega^2+1}}$

Un nodo particular de interés es el nodo de salida del circuito propuesto por Gielen [25] para medir el rendimiento de un simulador simbólico, el gráfico mostrado en la figura 4.8 sustenta la validación para el nodo número nueve correspondiente con la salida del circuito, el gráfico

mostrado representa el comportamiento en amplitud por ambos simuladores SCAM y SCAPy, mientras que el cuadro número 4.4 reafirma la salida simbólica de ambos simuladores.

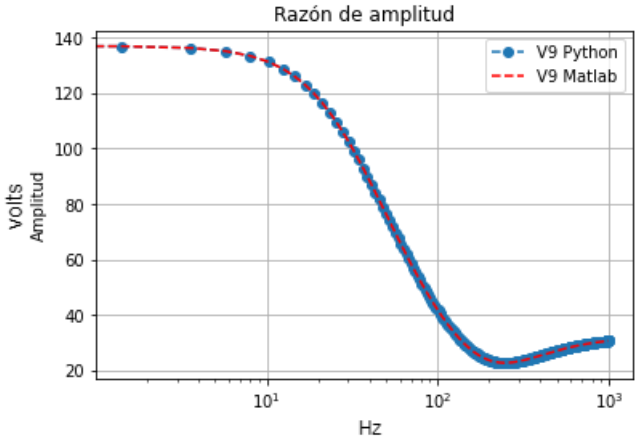


Figura 4.8: Frecuencia en nodo 9 semi-simbólico

Cuadro 4.5: Nodo 3

Python:	$\frac{1.0135125\|\omega\|}{\sqrt{9.317075625\cdot 10^{-9}\omega^4+0.000842180625\omega^2+1}}$
MATLAB:	$\frac{1.0135125\|\omega\|}{\sqrt{9.317075625\cdot 10^{-9}\omega^4+0.000842180625\omega^2+1}}$

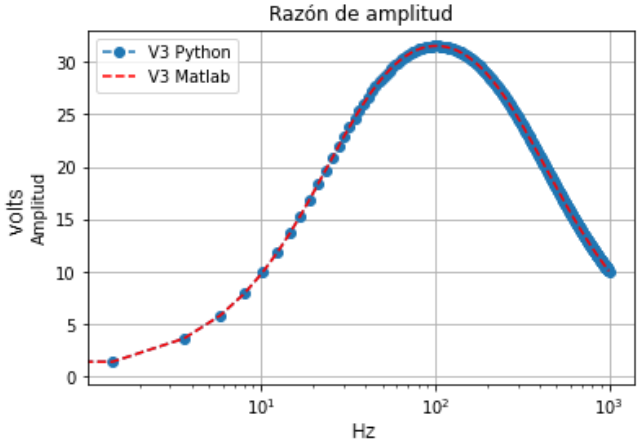


Figura 4.9: Frecuencia en nodo 3 semi-simbólico

Cuadro 4.6: Nodo 5

Python:	$\frac{1.0135125\|\omega\|}{\sqrt{9.317075625\cdot10^{-9}\omega^4+0.000842180625\omega^2+1}}$
MATLAB:	$\frac{1.0135125\|\omega\|}{\sqrt{9.317075625\cdot10^{-9}\omega^4+0.000842180625\omega^2+1}}$

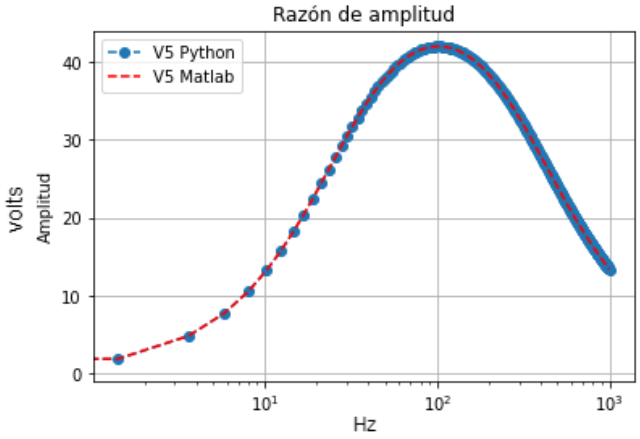


Figura 4.10: Frecuencia en nodo 5 semi-simbólico

Cuadro 4.7: Nodo 7

Python:	$\frac{34.65}{\sqrt{9.317075625\cdot10^{-9}\omega^4+0.000842180625\omega^2+1}}$
MATLAB:	$\frac{34.65}{\sqrt{9.317075625\cdot10^{-9}\omega^4+0.000842180625\omega^2+1}}$

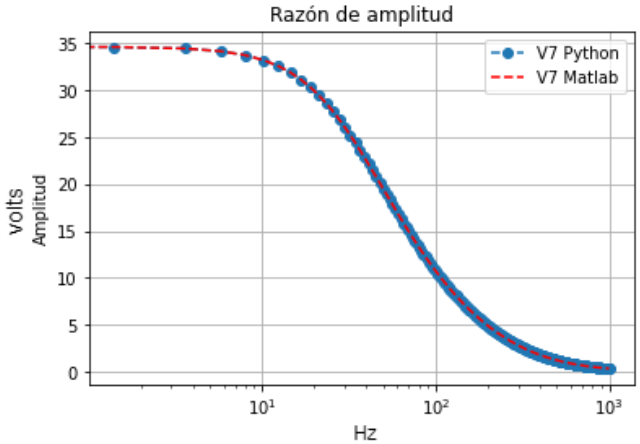


Figura 4.11: Frecuencia en nodo 7 semi-simbólico

La graficas 4.9 4.10 y 4.11 muestran los resultados obtenidos para los nodos tres, cinco y siete

respectivamente devueltos por SCAPy en color azul y SCAM en color rojo, además los cuadros 4.5 4.6 y 4.7 permiten observar los resultados simbólicos obtenidos en ambos simuladores, con estos voltajes obtenidos para cada nodo en el circuito se valida la confiabilidad de SCAPy.

4.2. Memoria

Uno de los aspectos importantes a tener en consideración es la memoria RAM del hardware para la validación de este algoritmo además de verificar que los resultados simbólicos sean los correctos, también se midió la memoria RAM consumida por este sistema de $15x15$ de 134.6 Mebibytes (MiB) equivalente a 141.1383 Megabytes, la función que se usó para medir la memoria se llama Memory Profiler, es una herramienta de Python, el tiempo que le tomó al algoritmo sin la medición de memoria fueron 16.2 segundos resolver todo el sistema.

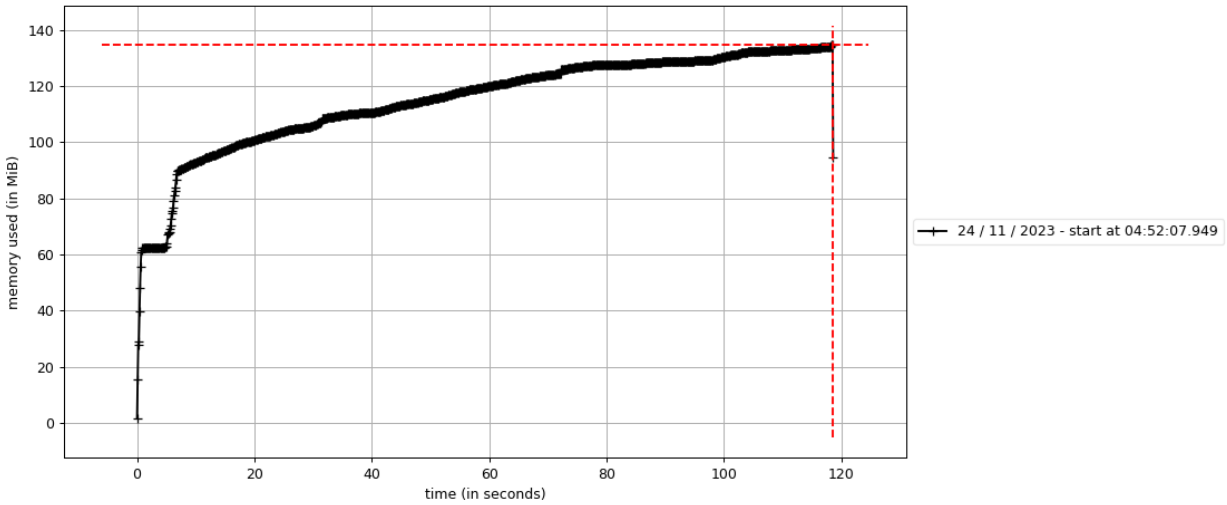


Figura 4.12: Memoria gastada en cálculos del circuito

4.3. Tiempo de computo

A continuación se muestra una tabla comparativa en los tiempos que le tomó a DDD en serie, DDD en paralelo, MATLAB 2023 y SymPy resolviendo determinantes simbólicos llenos, las consideraciones a tomar en cuenta es que en cada computadora pueden resultar tiempos diferentes, la computadora en que se resolvieron los determinantes simbólicos fue en arquitectura ryzen 7 serie 4000H con sistema operativo Windows 11

Cuadro 4.8: Tabla comparativa de tiempo

Tamaño lleno	5	6	7	8	9	10
DDD serie	0.471658 s	0.489248 s	0.667794 s	2.230921 s	16.744515 s	2min 8012 s
DDD paralelo	1.479939 s	1.580352 s	4.741123 s	5.747280 s	7.963706 s	30.636130 s
MATLAB	0.051301 s	0.334005 s	1.076124 s	23.063503 s	34min 5744 s	—
SymPy	3.28 s	4min 48s	—	—	—	—

4.4. Circuitos analógicos de prueba para simulación simbólica y semisimbólica en SCAPy

4.4.1. Integrador Fraccional

Este circuito mostrado en la figura 4.13 resulta tener un alto costo computacional y fue propuesto por [26] los nodos de interés resultan ser el nodo 4 y nodo 9 con las salidas PDA y PAA despectivamente.

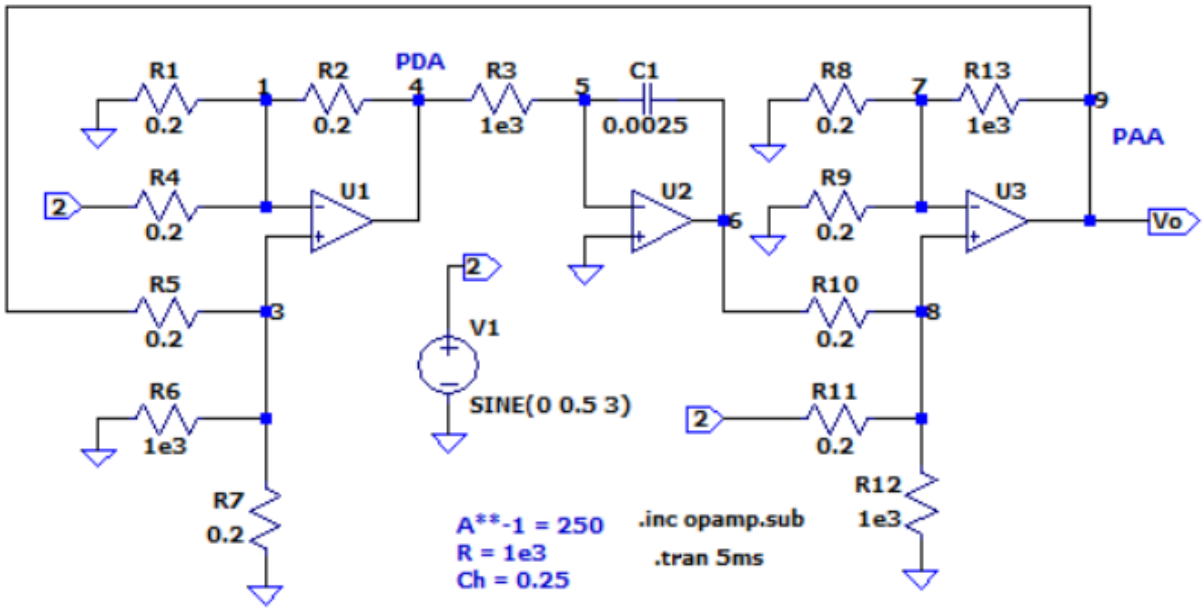


Figura 4.13: Integrador Fraccional

Para este caso particular se obtuvo la salida en el dominio de la frecuencia, pero dándole un enfoque hacia una evaluación en el dominio del tiempo y haciendo una comparativa en el rendimiento entre SCAPy con el algoritmo de solución DDD serie y SCAM, por último haciendo una simulación en LTSpice se obtuvo lo siguiente la presente salida para los nodos 4 y 9

$$v4 = -C1*R3*V1*s*(R1*R2*R8*R9*(R10*R11+R10*R12+R11*R12)*(R5*R6+R5*R7 +R6*R7)-R10*R12*R6*R7*(R1*R2+R1*R4+R2*R4)*(R13*R8+R13*R9+R8*R9))/(C1 *R1*R3*R4*R8*R9*s*(R10*R11+R10*R12+R11*R12)*(R5*R6+R5*R7+R6*R7)+R11*R12 * R6 * R7 * (R1 * R2 + R1 * R4 + R2 * R4) * (R13 * R8 + R13 * R9 + R8 * R9))$$

$$v9 = R1*R12*V1*(C1*R10*R3*R4*s-R11*R2)*(R13*R8+R13*R9+R8*R9)*(R5*R6+ R5*R7+R6*R7)/(C1*R1*R3*R4*R8*R9*s*(R10*R11+R10*R12+R11*R12)*(R5*R6+ R5*R7+R6*R7)+R11*R12*R6*R7*(R1*R2+R1*R4+R2*R4)*(R13*R8+R13*R9+R8*R9))$$

Con un tiempo de cómputo medido: 2.3268 segundos en el modo de cálculo, serie este cálculo se efectuó en una CPU ryzen 7 400H. Del las salidas simbólicas obtenidas en los nodos 4 y 9 en el dominio de la frecuencia se obtienen las funciones de transferencia correspondientes con H4 y H9 respectivamente:

$$H4 = -C1*R3*s*(R1*R2*R8*R9*(R10*R11+R10*R12+R11*R12)*(R5*R6+R5*R7 + R6 * R7) - R10 * R12 * R6 * R7 * (R1 * R2 + R1 * R4 + R2 * R4) * (R13 * R8 + R13 * R9 + R8 * R9))/(C1 * R1 * R3 * R4 * R8 * R9 * s * (R10 * R11 + R10 * R12 + R11 * R12) * (R5 * R6 + R5 * R7 + R6 * R7) + R11 * R12 * R6 * R7 * (R1 * R2 + R1 * R4 + R2 * R4) * (R13 * R8+ R13 * R9 + R8 * R9))$$

$$H9 = R1*R12*(C1*R10*R3*R4*s-R11*R2)*(R13*R8+R13*R9+R8*R9)*(R5*R6 + R5*R7+R6*R7)/(C1*R1*R3*R4*R8*R9*s*(R10*R11+R10*R12+R11*R12)* (R5*R6+R5*R7+R6*R7)+R11*R12*R6*R7*(R1*R2+R1*R4+R2*R4)*(R13 * R8 + R13 * R9 + R8 * R9))$$

Con las funciones de transferencia se multiplica por la entrada de voltaje en el dominio de Laplace, si la entrada es: $0.5 * sen(wt) * u(t)$ donde $w = 2 * \pi * 3$ entonces la entrada en el dominio de Laplace:

$$vin = 5305678314004791/(562949953421312 * (s^2 + 281502223 71700721605604770953681/7922816251426433759354395033 6))$$

El último paso es multiplicar la entrada en el dominio de Laplace por las respectivas funciones de transferencia y devolverlas al dominio del tiempo con la transformada inversa de Laplace:

v4=(550 569 441 892 372 854 727 459 811 440 664 530 075 928 974 118 041 992 187 5* cos ((530 567 831 400 479 1*t)/ 281 474 976 710 656))/ 233 709 319 597 453 954 80 733 094 448 174 049 718 084 080 494 817 053 900 8-(550 569 441 892 372 854 727 459 811 440 664 530 0759 289 741 180 419 921 875 * exp(-(322 154 759 454 720 078 125* t)/ 107 395 658 310 221 824))/ 233 709 319 597 453 954 807 330 944 481 740 497 180 840 804 948 170 539 008 +(906 933 242 652 845 586 156 679 348 020 328 668 582 324 942 669 047 730 351 562 5*sin ((530 567 831 400 479 1*t) / 281 474 976 710 656))/ 612 654 958 7655 496 952 901 296 311 022 138 089 297 433 197 233 321 777 771 315 2

v9=(112 054 721 334 606 710 823 665 370 814 369 599 072 221 520 592 896 000 * cos((530 567 831 400 479 1*t)/ 281 474 976 710 656))/ 713 224 241 935 589 461 692 294 142 095 155 325 869 265 151 819 368 1-(112 054 721 334 606 710 823 665 370 814 369 599 072 221 520 592 896 000 * exp(-(322 154 759 454 720 078 125 *t)/ 107 395 658 310 221 824))/ 713 224 241 935 589 461 692 294 142 095 155 325 869 265 151 819 368 1-(167 352 089 645 389 849 140 586 570 709 986 211 161 645 157 975 00 *sin((530 567 831 400 479 1*t) /281 474 976 710 656))/ 713 224 241 935 589 461 692 294 142 095 155 325 869 265 151 819 3681

El resultado es mostrado en la figura 4.14

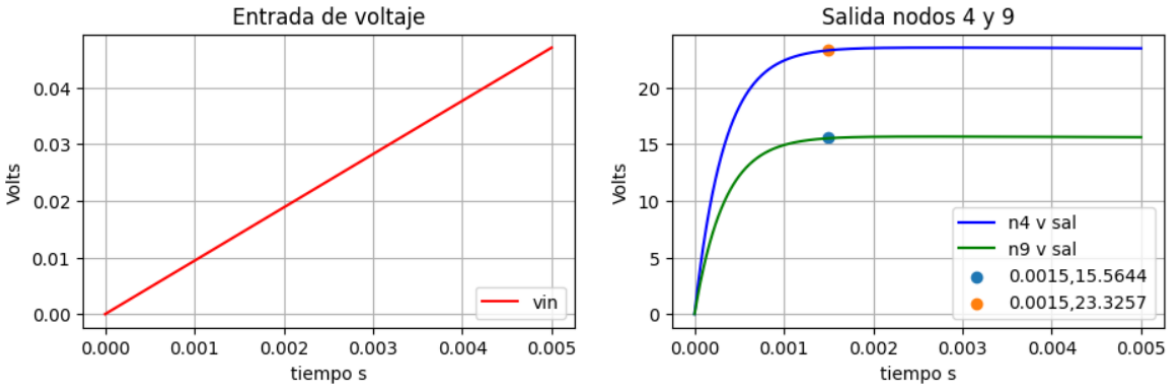


Figura 4.14: Integrador Fraccional nodos 4 y 9 simbólico

El resultado puede ser comparado con LTspice como se muestra en el gráfico 4.15, si se hace la comparativa seleccionando un punto en el vector del tiempo, para este caso se selecciona un punto en 1.5 ms para el nodo 4 y 9 respectivamente se puede observar una misma amplitud de voltaje para ambas simulaciones correspondientes con 23.32 y 15.56 volts para ambos nodos.

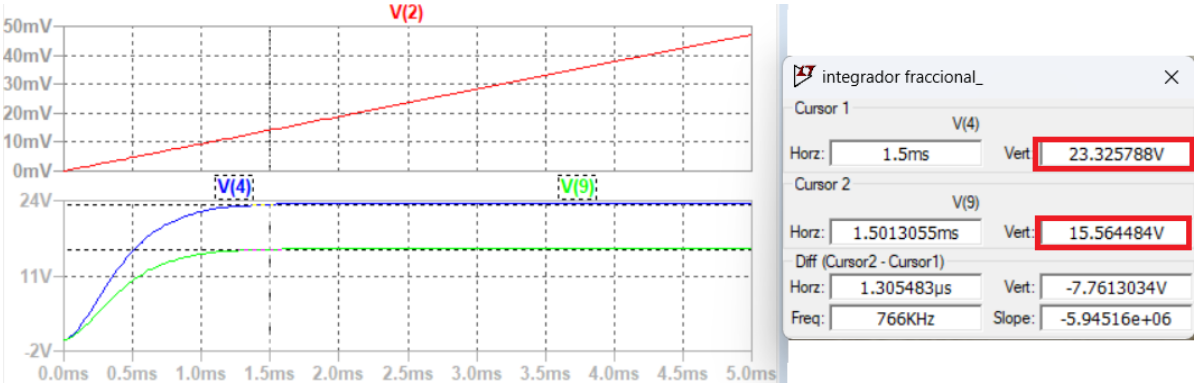


Figura 4.15: Integrador Fraccional nodos 4 y 9 numérico

4.4.2. Filtro WTA LTA

Los filtros WTA (winer Take All) y LTA (Loser Take All) son filtros de estadística esto quiere decir que en un gráfico con diferentes entradas, amplitudes y frecuencias, cada filtro se quedará con la mayor o menor amplitud de señal, se ha preparado un circuito que permite incorporar elementos eléctricos y compatible con SCAPy, el circuito en cuestión retomando la idea de [27] implementar dicho filtro reemplazando el transistor por un modelo PI de pequeña señal, y el más apto para esta configuración que permite representar disparos con la amplificación de las fuentes [28], ajustando las ganancias para las fuentes de corriente controladas por voltaje de dado por las mismas recomendaciones de [28] y reemplazando las fuentes de Ibias por resistencias de $1e12$ ohms se planteó el siguiente circuito mostrado en la figura 4.16 b y 4.17 b:

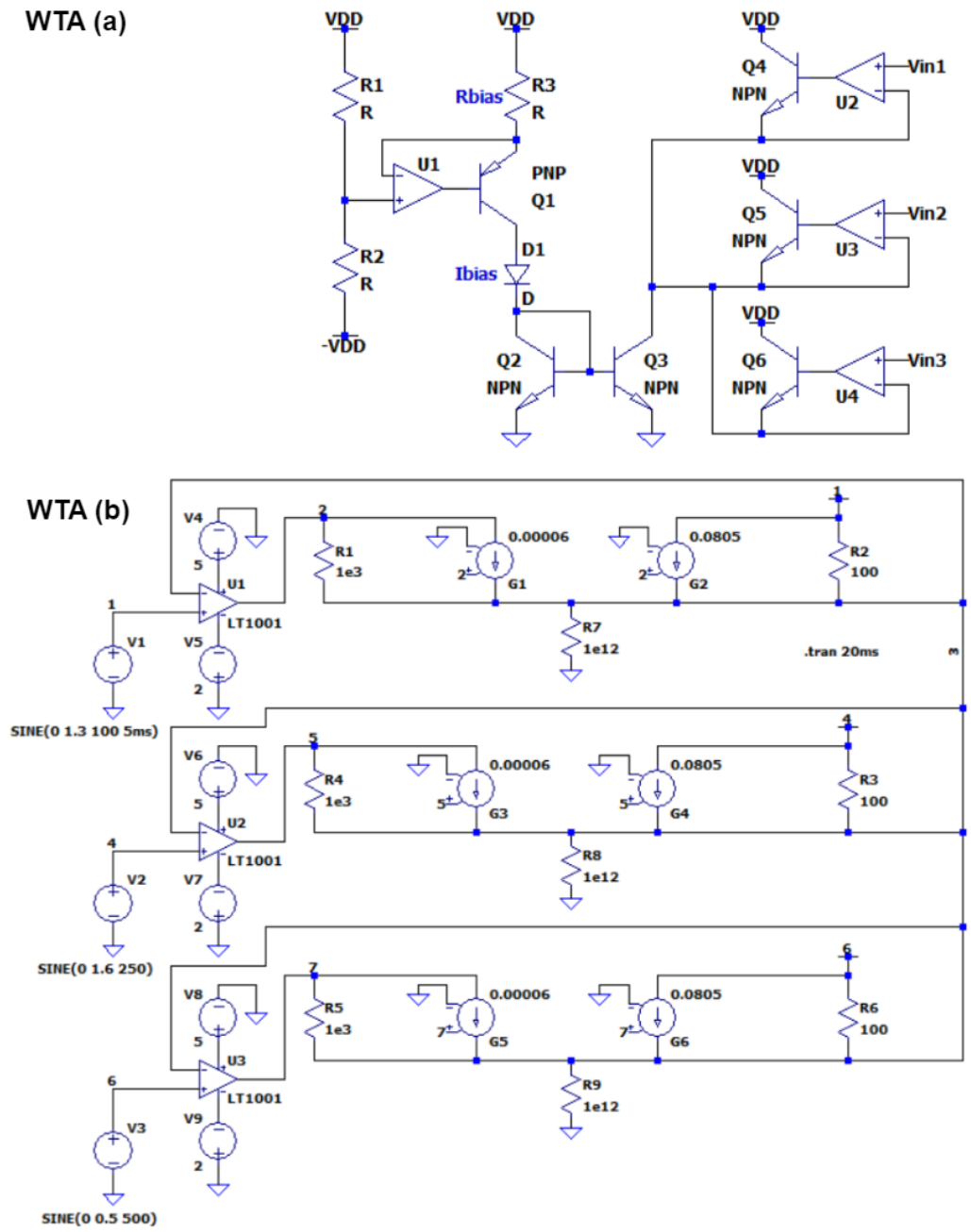


Figura 4.16: Filtro WTA (a) muestra la configuración eléctrica no lineal de donde se tomó el filtro, mientras que (b) muestra la adaptación que se hizo para el filtro WTA con elementos lineales

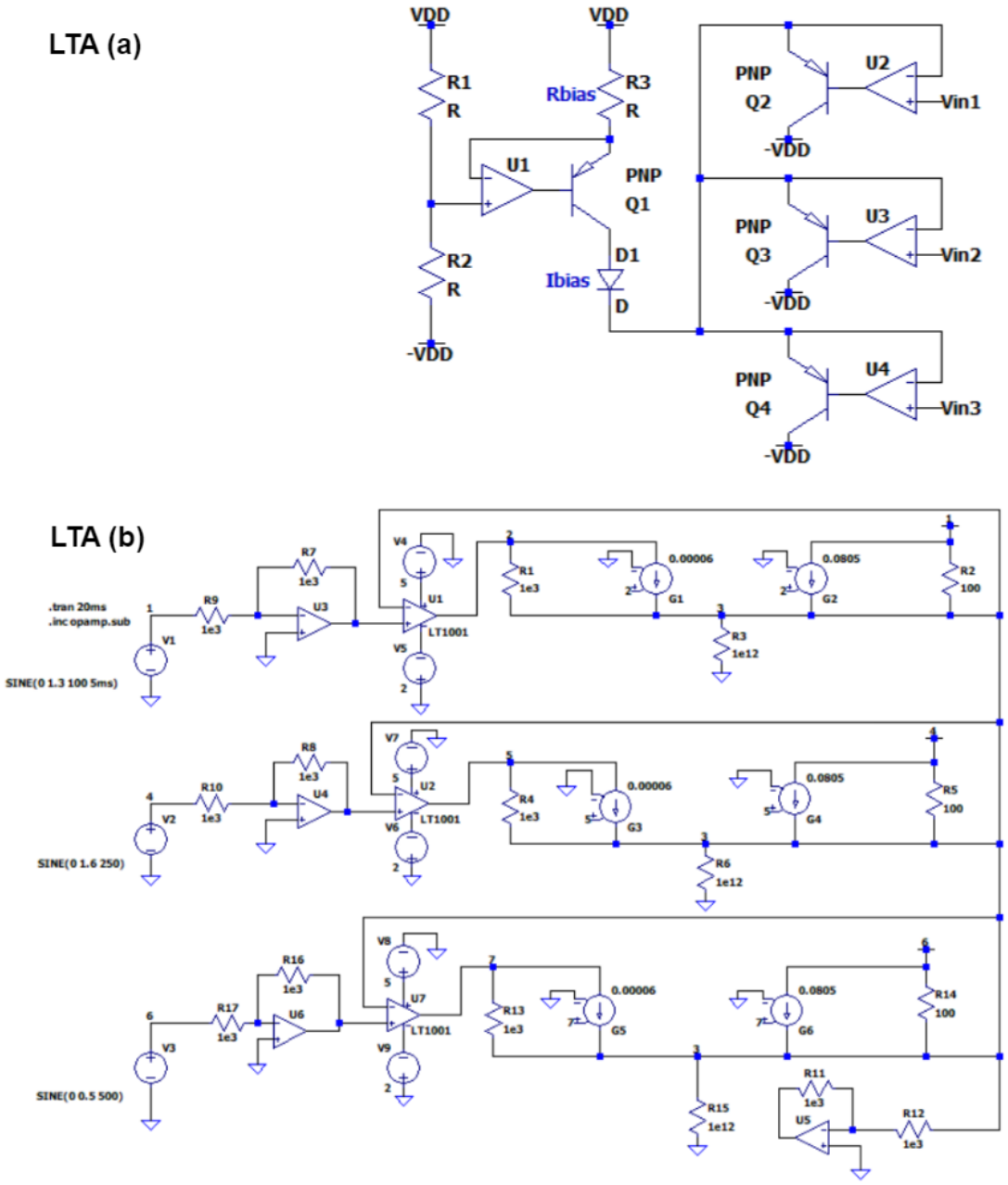


Figura 4.17: Filtro LTA (a) muestra la configuración eléctrica no lineal de donde se tomó el filtro, mientras que (b) muestra la adaptación que se hizo para el filtro LTA con elementos lineales

Simulando el presente circuito de las figuras 4.16 (b) y 4.17 (b) se puede observar el acoplamiento de tres señales a diferentes amplitudes y diferentes frecuencias, también se pueden observar las resistencias $R7$, $R8$ y $R9$ del WTA como resistencias de inhibición al igual que las resistencias $R3$, $R6$ y $R15$ del LTA, la salida de voltaje se encuentra dado en el nodo 3 como

se muestra en la figura 4.18.

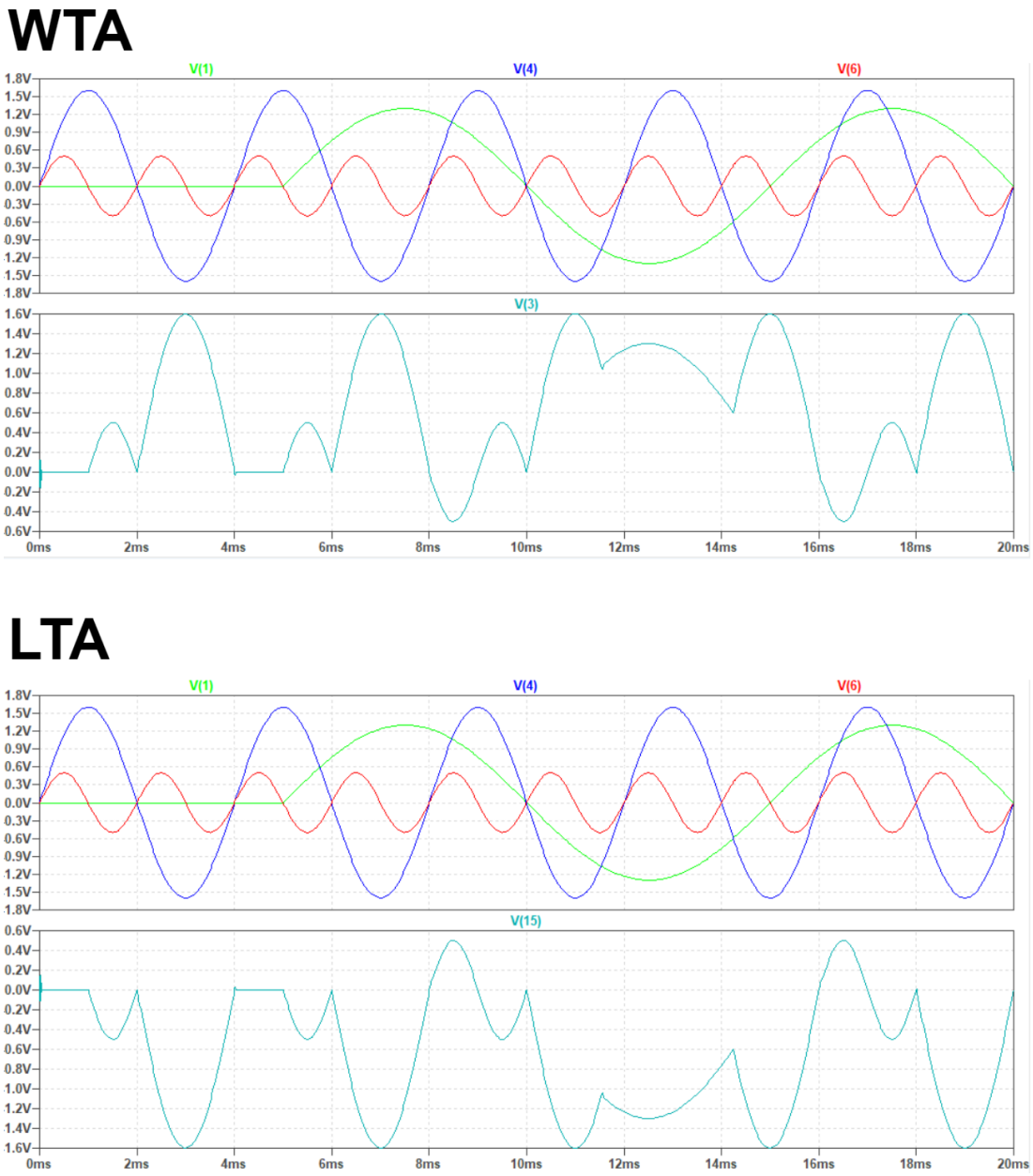


Figura 4.18: Salidas WTA LTA

El cálculo en SCAPy a partir de la netlist generada por ambos filtros WTA y LTA resulta en una matriz indeterminada que no tiene solución:

$$A = \begin{bmatrix}
\frac{1}{R_2} & G_2 & -\frac{1}{R_2} & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & G_1+\frac{1}{R_1} & -\frac{1}{R_1} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
-\frac{1}{R_2} & -G_1-G_2-\frac{1}{R_1} & \frac{1}{R_9}+\frac{1}{R_8}+\frac{1}{R_7}+\frac{1}{R_6}+\frac{1}{R_5}+\frac{1}{R_4}+\frac{1}{R_3}+\frac{1}{R_2}+\frac{1}{R_1} & -\frac{1}{R_3} & -G_3-G_4-\frac{1}{R_4} & -\frac{1}{R_6} & -G_5-G_6-\frac{1}{R_5} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -\frac{1}{R_3} & \frac{1}{R_3} & G_4 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & -\frac{1}{R_4} & 0 & G_3+\frac{1}{R_4} & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & -\frac{1}{R_6} & 0 & 0 & \frac{1}{R_6} & G_6 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & -\frac{1}{R_5} & 0 & 0 & 0 & G_5+\frac{1}{R_5} & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

De acuerdo con la formulación de Roldán [4] la matriz D en conjunto con la matriz G, B y C hacen que DDD no encuentre una diagonal diferente de cero, haciendo que el sistema sea indeterminado cuando se encuentran todas las entradas conectadas al nodo 3, sin embargo, cuando solo se toma en cuenta una entrada de voltaje la salida en el nodo 3 es igual a la entrada para el circuito WTA y menos la entrada para el circuito LTA.

$$\begin{bmatrix}
\frac{1}{R_2} & G_2 & -\frac{1}{R_2} & 1 & 0 \\
0 & G_1+\frac{1}{R_1} & -\frac{1}{R_1} & 0 & 1 \\
-\frac{1}{R_2} & -G_1-G_2-\frac{1}{R_1} & \frac{1}{R_7}+\frac{1}{R_2}+\frac{1}{R_1} & 0 & 0 \\
1 & 0 & 0 & 0 & 0 \\
-1 & 0 & 1 & 0 & 0
\end{bmatrix}
\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ Iv_1 \\ I_{OAmp1} \end{bmatrix} =
\begin{bmatrix} 0 \\ 0 \\ 0 \\ V_1 \\ 0 \end{bmatrix}$$

$$x = \begin{bmatrix}
V_1 \\
\frac{V_1(R_1+R_7)}{R_7(G_1R_1+G_2R_1+1)} \\
V_1 \\
\frac{G_2V_1(-R_1-R_7)}{R_7(G_1R_1+G_2R_1+1)} \\
\frac{V_1(-G_1R_1+G_2R_7-1)}{R_7(G_1R_1+G_2R_1+1)}
\end{bmatrix}$$

Tomando en cuenta que si se conectan dos o más entradas el sistema se hace indefinido, entonces solo se toma el valor más alto y más bajo del voltaje, como se muestra en el gráfico de la figura 4.19

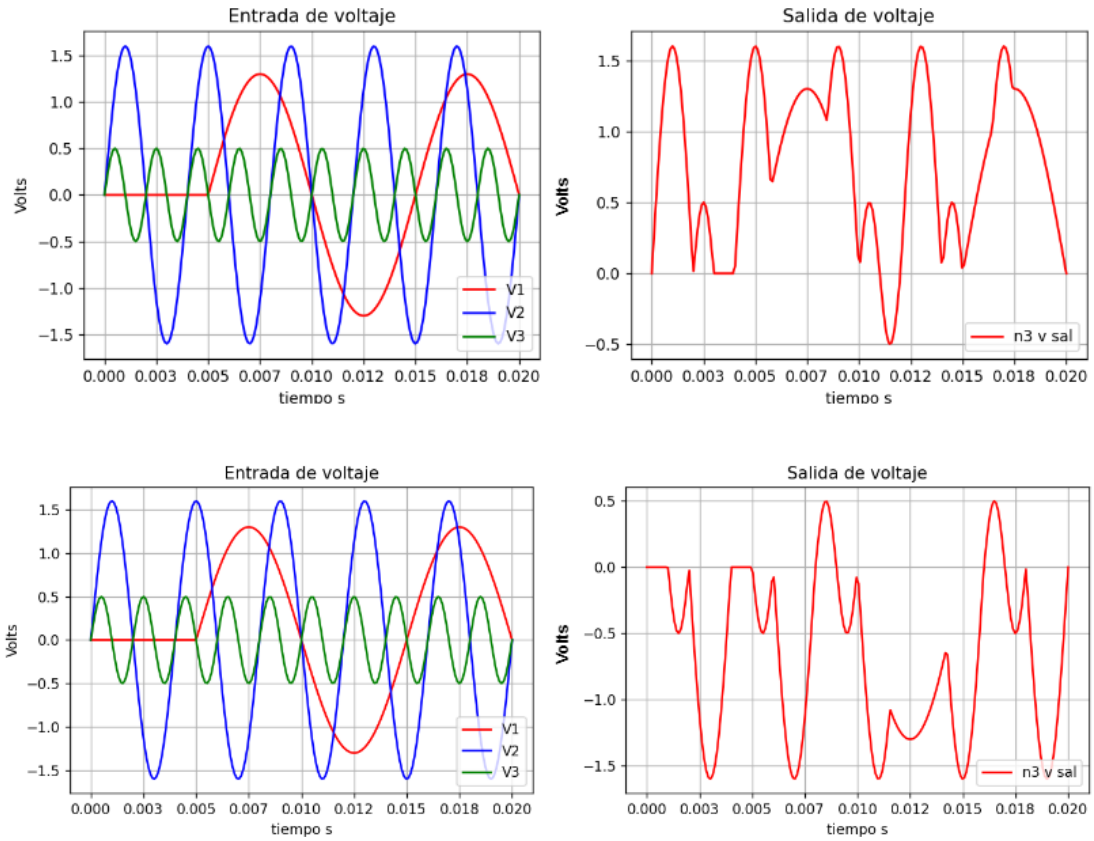


Figura 4.19: Salidas WTA LTA en Python

4.4.3. Memristor

El último circuito que se presenta es el memristor, El principio fundamental es su capacidad para recordar la cantidad de carga que ha pasado a través de él, lo que implica que su resistencia depende de la historia de corriente que ha circulado. Esto significa que, una vez que la corriente cesa, el memristor conserva su último estado de resistencia, permitiendo almacenar información sin necesidad de energía continua. Este comportamiento se asemeja a la memoria biológica.

El circuito que se ha preparado está basado en [29] mostrado en la figura 4.20

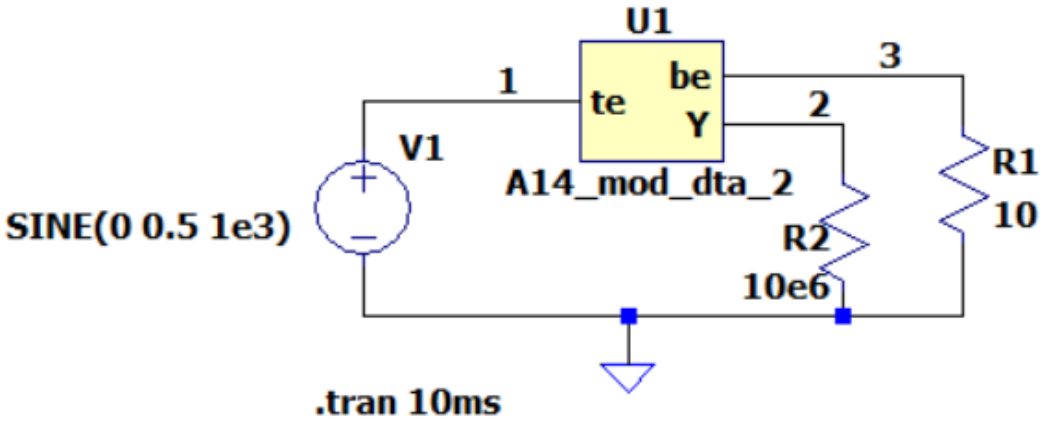


Figura 4.20: Memristor

El elemento U1 en cuestion consiste en un circuito por la siguiente netlist:

```
.subckt A14_mod_dta_2 te be Y
.params ron=100.7 roff=12.6e3 k=17803.4 Cint=1 m=3
.ic V(Y) = 0.6V
.FUNC FG1() {V(te)*((1/(ron*(V(Y))+roff*(-V(Y)+1))))}
.FUNC FG2() {(k*pow(V(te),m)*(pow(sin(pi*V(Y)),2)))}
.FUNC Fn() {V(Y)}
C1 Y be 1
R1 Y be 100G
*G1 te be te be 0.001
*G2 Y be te be 9000
G1 te be value={FG1()}
G2 Y be value={FG2()}
.ends A14_mod_dta
* For Citation:
*V. Mladenov and S. Kirilov, "An Improved Memristor Model and Applications,"
*2023 12th International Conference
*on Modern Circuits and Systems Technologies (MOCAST),
*Athens, Greece, 2023, pp. 1-4, doi: 10.1109/MOCAST57943.2023.10176507.
```

Como puede observarse es un netlist exótico, ya que hace una iteración de funciones internas en las fuentes de corriente controladas por voltaje estas funciones refieren a $FG1()$ y $FG2()$ respectivamente entonces como puede verse en los comentarios $*G1$ *te be te be* 0.001 y $*G2$ *Y be te be* 9000 se hizo con la intención de sacar un circuito compatible que no dependiera temporalmente de las funciones y poder graficar su comportamiento, el circuito de la figura 4.20 dadas las instrucciones presentadas en el netlist corresponde con el siguiente comportamiento mostrado en la figura 4.21

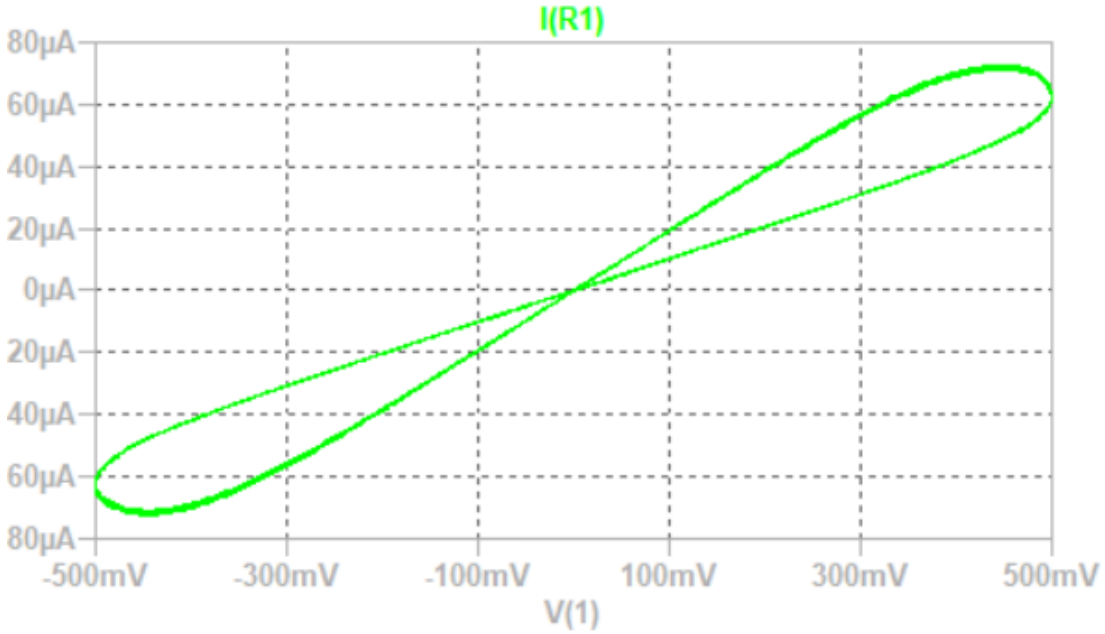


Figura 4.21: Histéresis del memristor

A continuación se muestra el circuito equivalente, con el que se estuvieron haciendo pruebas, asumiendo que ambas expresiones son equivalentes entre $G1$ y $G2$

```
*G1 te be te be k1}
*G2 Y be te be k2}
*G1 te be value={(V(te)-V(be))*k1}
*G2 Y be value={(V(te)-V(be))*k2}
```

Donde $k1$ y $k2$ toman el valor de $FG3$ y $FG4$:

```
.FUNC FG3() {(V(te)*((1/(ron*(V(Y))+roff*(-V(Y)+1)))))/(V(te)-V(be))}
.FUNC FG4() {((k*pow(V(te),m)*(pow(sin(pi*V(Y)),2))))/(V(te)-V(be))}
```

El circuito del memristor equivalente de la figura 4.22 no podría ser computado en LTspice por la forma de trabajo del mismo simulador, es por eso que se manda a instrucciones para tomar los valores de las funciones; sin embargo, en SCAPy interesa obtener el modelo simbólico obteniendo el siguiente netlist:

```
R1 2 3 100e9
R2 2 0 10e6
R3 3 0 10
C1 2 3 1
G1 1 3 1 3 F3
G2 2 3 1 3 F4
V1 1 0 V
```

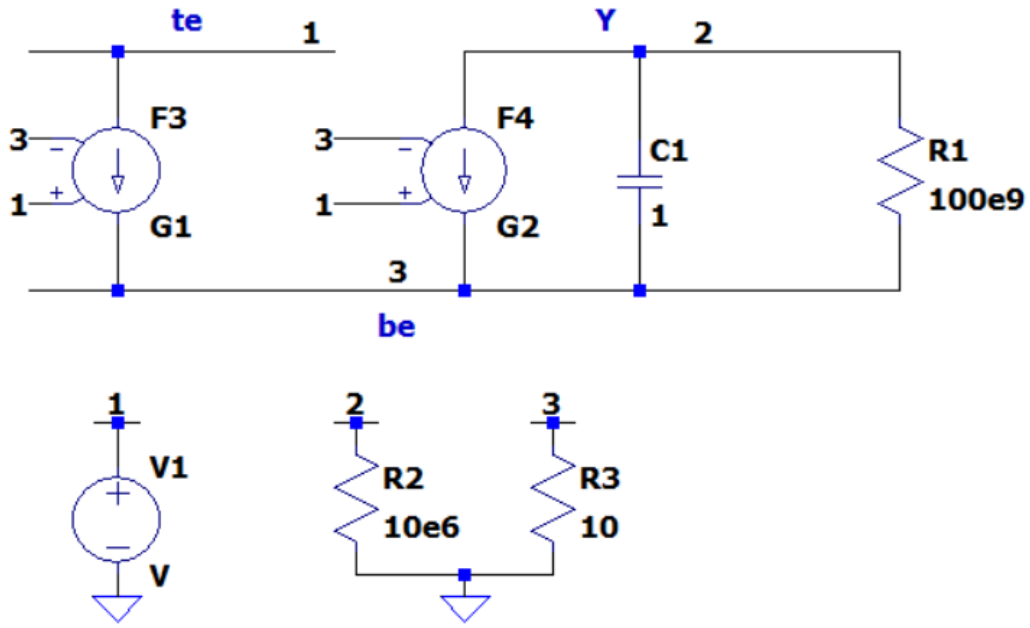


Figura 4.22: Memristor equivalente

Resolviendo el sistema:

$$\begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ Iv_1 \end{bmatrix} = \begin{bmatrix} V_1 \\ \frac{R_2 V_1 (C_1 G_1 R_1 R_3 s + G_1 R_3 - G_2 R_1)}{C_1 G_1 R_1 R_2 R_3 s + C_1 R_1 R_2 s + C_1 R_1 R_3 s + G_1 R_1 R_3 + G_1 R_2 R_3 + G_2 R_1 R_3 + R_1 + R_2 + R_3} \\ \frac{R_3 V_1 (C_1 G_1 R_1 R_2 s + G_1 R_1 + G_1 R_2 + G_2 R_1)}{C_1 G_1 R_1 R_2 R_3 s + C_1 R_1 R_2 s + C_1 R_1 R_3 s + G_1 R_1 R_3 + G_1 R_2 R_3 + G_2 R_1 R_3 + R_1 + R_2 + R_3} \\ -G_1 V_1 \end{bmatrix}$$

Interesa conocer el valor de voltaje en el nodo 2 y respectivamente, entonces el procedimiento realizado para conocer el voltaje en cada nodo, fue obteniendo la función de transferencia dividiendo la salida sobre la entrada, quedando la siguiente función de transferencia para dichos nodos (H_2, H_3) posteriormente se toma la entrada dada por $0.5 * sin(w * t)$ donde $w = 2 * pi * 1e3$: entonces la excitación del sistema en el dominio de la frecuencia queda como lo indica el voltaje uno:

$$H_2 = \frac{R_3 \left(C_1 G_1 R_1 R_2 s + G_1 R_2 - G_2 R_1\right)}{C_1 G_1 R_1 R_2 R_3 s - C_1 R_1 R_2 s - C_1 R_1 R_3 s + G_1 R_1 R_2 + G_1 R_2 R_3 + G_2 R_1 R_2 - R_1 - R_2 - R_3}$$

$$H_3 = \frac{R_2 \left(C_1 G_1 R_1 R_3 s + G_1 R_1 + G_1 R_3 + G_2 R_1\right)}{C_1 G_1 R_1 R_2 R_3 s - C_1 R_1 R_2 s - C_1 R_1 R_3 s + G_1 R_1 R_2 + G_1 R_2 R_3 + G_2 R_1 R_2 - R_1 - R_2 - R_3}$$

$$v_i = \frac{15708}{(5 * (s * *2 + 986965056/25))}$$

Multiplicando v_i por H_2 y sustituyendo los valores de la resistencias y capacitancia, y haciendo la transformada inversa de Laplace:

v2 = (3141 6000 000 0000 0000 000 000 0000 000 0000 0000 000* exp(-(t*(100 010 000000 * G1 + 1000000 00000 * G2 - 100 010 00 001)) / (100 0000 0000 000 00000 * G1 - 100 00010 0000 000000))*(1000 0000 * G1 - 1000001) * (G1 + 1000001 * G2 - 10 * G1**2 - 1000 0010 *G1*G2))/((1000 000 000 000 000 000 * G1 - 10000 01000 0000 0000)*(394786 02240 0000 000 00010 002 000 1000 0000 0000 000 * G1**2 + 200 0200 0000 0000 00000 000 * G1*G2 - 789 5728 34372 044 800 0002 0004 000 202 0002 0000 000 * G1 + 1000 000 0000 0000 0000 0000 * G2**2 - 2000 2000 002 0000 000 0000 * G2 + 3947 86811 97243 95860 22500 02000 10200 0200 0001)) - (5000 0000 0000 0000 0000 0000 0000 * G2**2* sin((31416*t)/5) - 19739 301120 00000 000 000 0000 50005 000 0000 000000 * G1**2 *sin((31416*t)/5) - 31416 000 000 0000 0000 000 0000 0000 000 * G1**2 * cos((31416*t)/5) + 3141 6000 0000 0000 000 00 0000 0000 000 * G1* cos((31416*t)/5) + 3141 6031 416 0000 00000 0000 0000 00000 00000 *G2*cos((31416*t)/5) + 197 3932 0859 30112 0000 00000 05000 50000 0500 0000 * G1 *sin((31416*t)/5) - 5000 500 000 5000 0000 0000 00 000 *G2 * sin((31416*t)/5) - 31416 03141 6000 00000 00000 0000 00000 000 000 * G1 * G2*cos((31416*t)/5) + 50004 9999 95000 00000 0000 00000 * G1 * G2 * sin((31416*t)/5))/(3947 8602 24000 0000 00001 0002 0001 0000 0000 0000 00 * G1**2 + 20002 00000 00000 0000 0000 * G1 * G2 - 7895 72834 372 044 8000 00200 040 002 02000 20000 000 * G1 + 1000 0000 000 0000 000 000 00 * G2**2 - 20002 0000 020 0000 0000 00 * G2 + 39478 681 1972 4395 8602 2500 0200 010200 020 000 01)

multiplicando v_i po H_3 se obtiene:

v3 = (1973930 112000 0000 0000 050010 0005 000 0000 0000 00 * G1**2 * sin((31416*t)/5) + 5000 0000 0000 0000 000 000 * G2**2 * sin((31416*t)/5) + 3141 6000 0000 0000 00000 000 00 * G1*cos((31416*t)/5) + 31416 03141 60000 00000 00000 000 0000 * G2*cos((31416*t)/5) - 1973 93208 59301 12000 00050 01000 05050 00 5000 000 * G1*sin((31416*t)/5) - 5000 50000 05000 0000 000 * G2*sin((31416*t)/5) + 10001 00000 000000 0000 000 * G1 * G2*sin((31416*t)/5))/(394 78602 2400 0000 0000 0100 0200 01000 0000 0000 000 * G1**2 + 2000 20000 000 0000 0000 000 * G1*G2 - 7895 72834 372044 80000 020004 0002020 0020 000 000 * G1 + 1000 0000 000 0000 00000 000 * G2**2 - 20002 000002 000000 00000 * G2 + 3947 86811 9724 3958 6022 50002 0001 0200 0200 0001) - (3141 60000 00000 000 00000 0000 000 000 00000 * exp(-(t*(1000 1000 0000 * G1 + 1000 00 000 000 * G2 - 1000 1000 001))/(10000 0000 0000 000 000 * G1 - 1000 0010 000 0000 000)) * (100 00000 * G1 - 100 0001) * (G1 + 100 0001 * G2))/((1000 0000 0000 0000 000 * G1 - 1000 00100 00000 0000) * (39478 6022 40000 0000 00010 0020 00100 0000 0000 0000 * G1**2 + 2000 2000 00000 0000 0000 00 * G1*G2 - 7895 7283 4372 04480 0000 20004 0002 02000 200 00000*G1 + 1000 0000 0000 0000 0000 000 * G2**2 - 2000 2000 00200 00000 0000 * G2 + 3947 86811 97243 95860 2250 0020 0010 2000 2000 001))

Una forma de validar el circuito es proponiendo valores y observar su gráfico comparando entre lo que se obtiene en LTspice y los valores al graficar en Python, del circuito de la figura 4.20 se muestra la unidad U1, esta unidad contiene la información de G_1 y G_2 para validar el modelo, se habilitan los siguientes valores:

- G_1 *be te value* $= \{(V(te) - V(be)) * 0.0008\}$
- G_2 *be Y value* $= \{(V(te) - V(be)) * 1000\}$
- $.ic\ V(Y) = 0.0V$

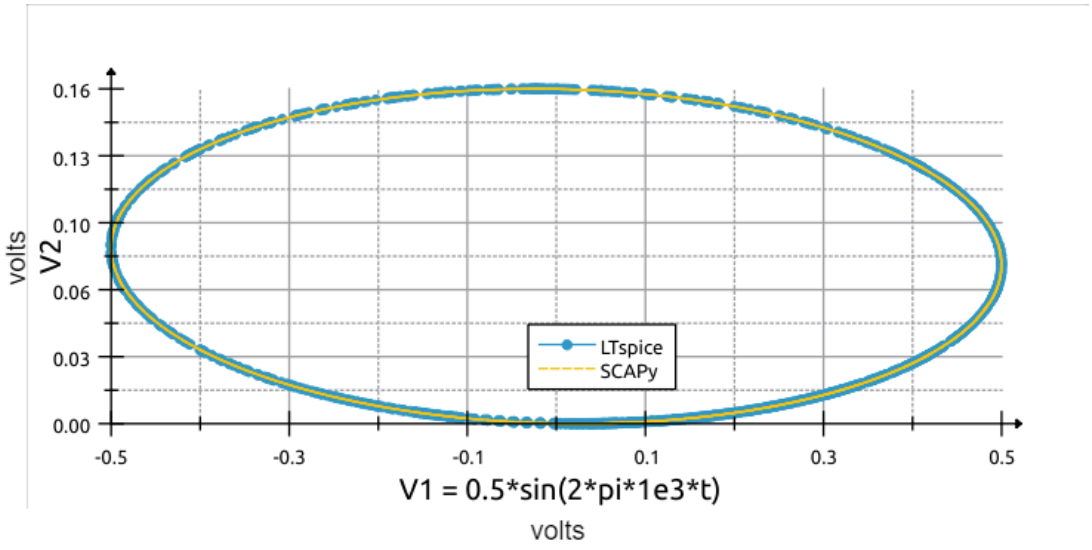


Figura 4.23: Resultados en LTspice y SCAPy para el nodo 2 (validación 1)

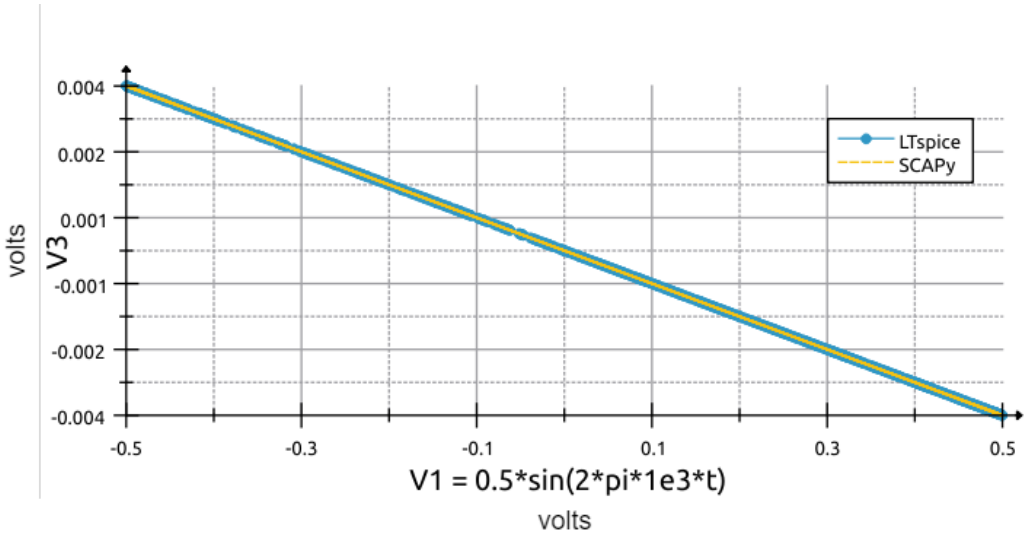


Figura 4.24: Resultados en LTspice y SCAPy para el nodo 3 (validación 1)

Los gráficos mostrados en las figuras: 4.23 y 4.24 muestran el comportamiento de los voltajes dos y tres del circuito mostrado en la figura 4.22 para SCAPy y LTspice 4.20 proponiendo los valores señalados en la parte superior.

- $G_1\ be\ te\ value = \{(V(te) - V(be)) * -0.0008\}$
- $G_2\ be\ Y\ value = \{(V(te) - V(be)) * 30\}$
- $.ic\ V(Y) = 0.0V$

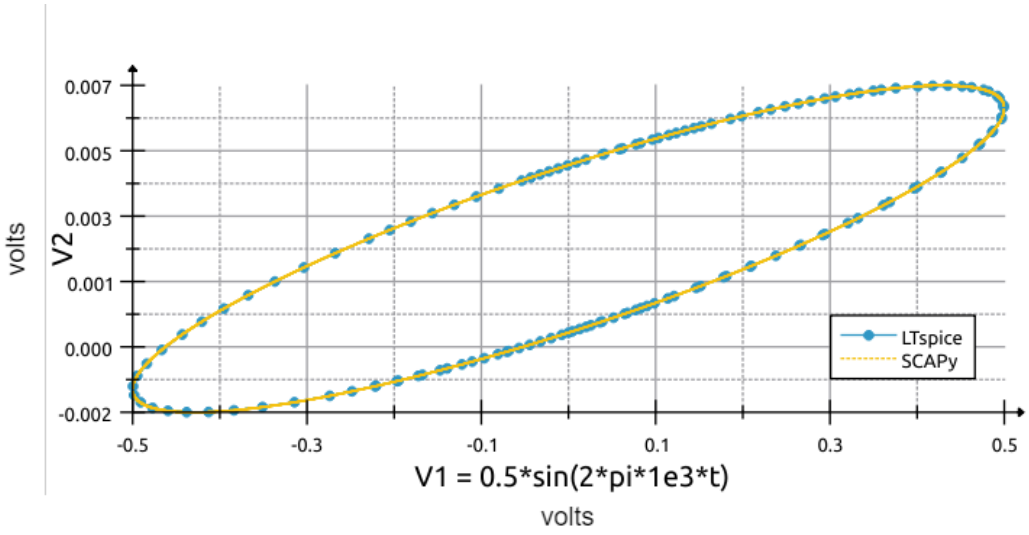


Figura 4.25: Resultados en LTspice y SCAPy para el nodo 2 (validación 2)

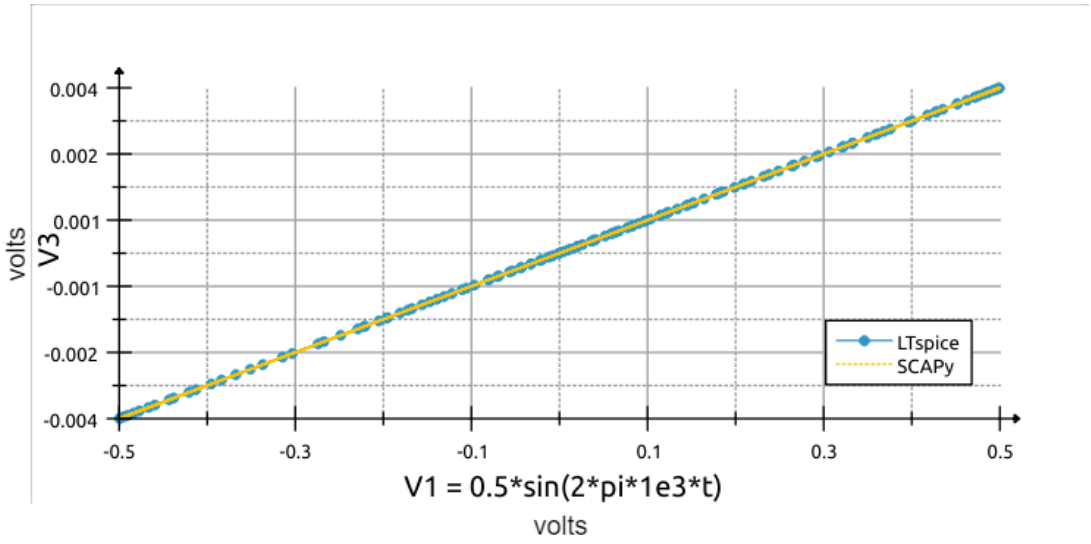


Figura 4.26: Resultados en LTspice y SCAPy para el nodo 3 (validación 2)

Los gráficos mostrados en las figuras 4.25 y 4.26 muestran el comportamiento entre ambos simuladores LTspice y SCAPy, pero ahora con distintos valores en las ganancias de las fuentes, observando un mismo comportamiento, recordando que el circuito computado en SCAPy es un circuito equivalente, ya que en SCAPy se simula el circuito que equivale a la unidad U1 del memristor mostrado en la figura 4.20.

- $.FUNC\ FG3()\ \{(V(te)*((1/(ron*(V(Y))+roff*(-V(Y)+1)))))/(V(te)-V(be))\}$
- $G_1\ be\ te\ value = \{(V(te)-V(be))*FG3()\}$
- $G_2\ be\ Y\ value = \{(V(te)-V(be))*1800\}$
- $.ic\ V(Y) = 0.0V$

A continuación se muestra los resultados obtenidos en los gráficos mostrados en las figuras 4.27 y 4.28, pero esta vez reemplazando los valores de las ganancias G_1 y G_2 por los mostrados, donde el valor de G_1 corresponde con el valor devuelto por la función $FG3$ añadida a la unidad U1 del circuito del memristor mostrado en la figura 4.20 y añadido en la iteración para la obtención del gráfico en Python, para más detalle acerca del procedimiento en Python con SCAPy se puede consultar en los ejemplos preparados para esta Tesis y que se encuentran disponibles en el repositorio de GitHub https://github.com/luisCorl/E2SCAPy-examples/tree/main/memristor.

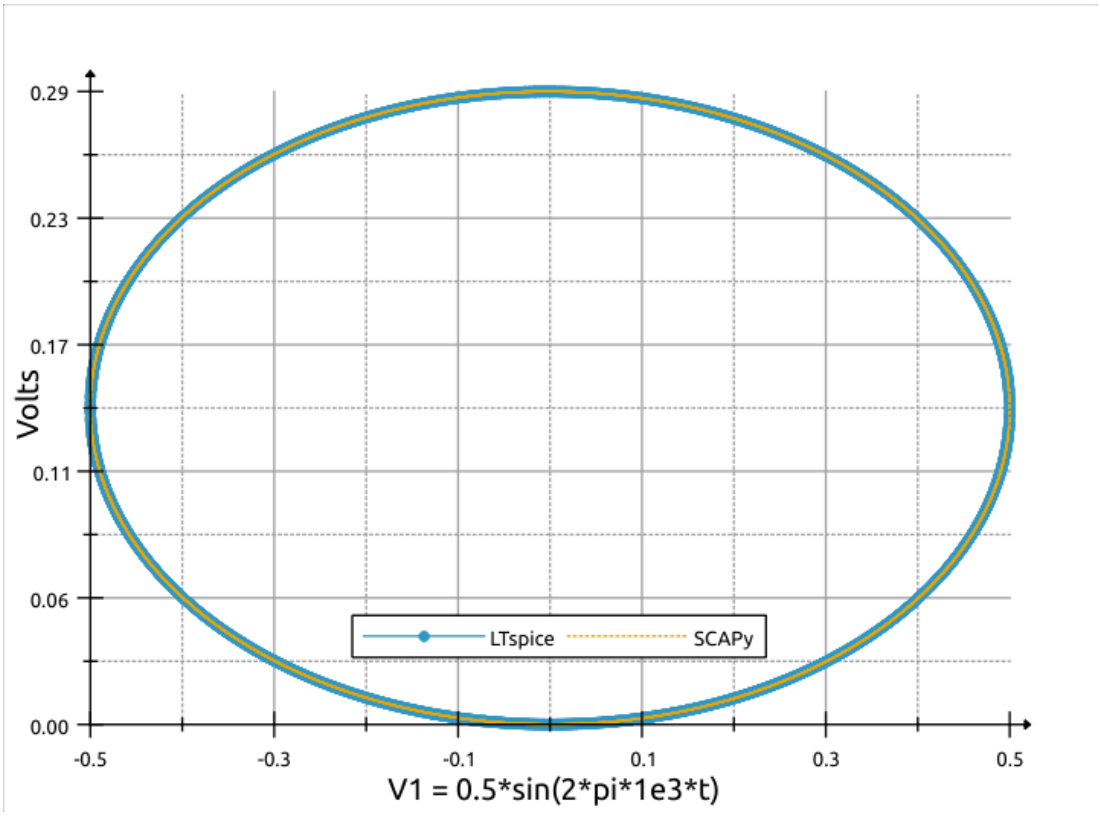


Figura 4.27: Resultados en LTspice y SCAPy validacion 3 nodo 2

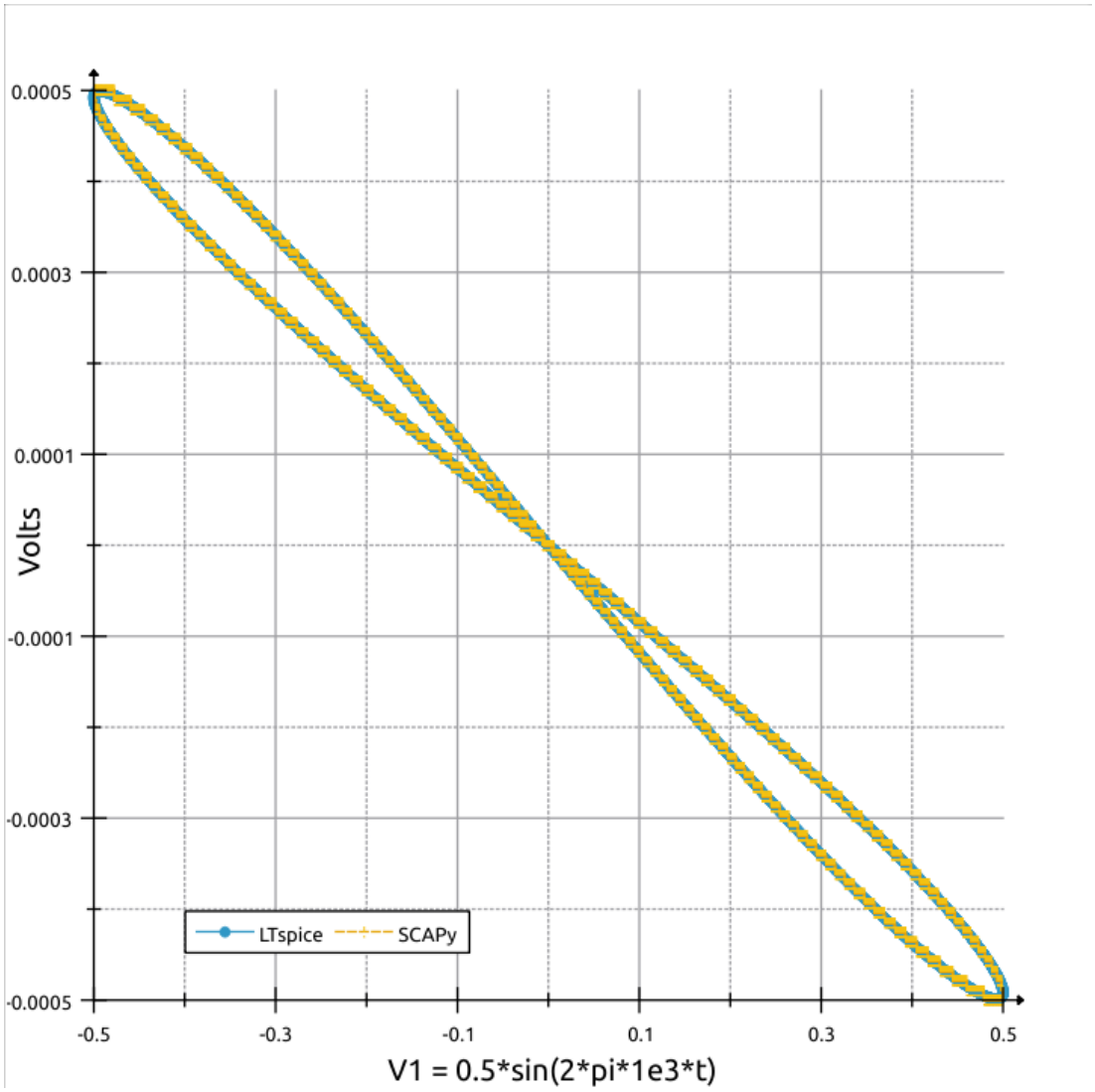


Figura 4.28: Resultados en LTspice y SCAPy validacion 3 nodo 3

De esta manera se hace válido el modelo de memristor computado en SCAPy y también se muestra como parar del dominio de la frecuencia al dominio del tiempo, esperando ser de utilidad para el lector todo este trabajo realizado.

A continuación se muestra una comparativa de tiempos entre SCAPy con el método simbólico que oferta SymPy, el método DDD desarrollado en esta tesis y SCAM de Matlab, cabe mencionar que MATLAB ha optimizado mucho sus algoritmos en estos años es por eso que consideramos medir el tiempo de cómputo entre estas tres opciones para obtener el cálculo simbólico y observar el rendimiento, para los circuitos en los que se estuvo trabajando.

	DDD serie	SymPy	MATLAB 2023b SCAM
Gielen	13.2178 s	///////	2.82814 s
Integrador fraccional	2.1871 s	///////	4.59351 s
WTA	0.022 s	0.4396 s	1.3346 s
LTA	0.6124 s	1.8091 s	1.8605 s
Memristor	0.0001 s	0.9133 s	1.4 s

Cuadro 4.9: Tiempos de computo

4.5. Conclusiones

El capítulo cuatro expone de manera detallada los resultados derivados del arduo trabajo en el desarrollo e implementación del simulador simbólico SCAPy. Estos resultados se obtuvieron a través de la validación sistemática de diferentes circuitos eléctricos y electrónicos, demostrando la robustez y eficacia del simulador. La diversidad de circuitos analizados no solo reafirma la versatilidad de SCAPy en el manejo de diversas configuraciones y complejidades de circuitos, sino que también destaca su capacidad para proporcionar soluciones simbólicas precisas. Esta validación minuciosa subraya la contribución significativa de SCAPy al campo de la ingeniería eléctrica electrónica, ofreciendo una herramienta de simulación confiable y efectiva para el análisis de circuitos.

Los resultados observados a través de SCAPy muestran una alta coincidencia con los obtenidos mediante el simulador simbólico LTSPICE, un estándar reconocido en la industria. Esta concordancia es crucial, ya que proporciona una validación independiente de la precisión y confiabilidad de SCAPy. La comparación sistemática con LTSPICE permite a los usuarios confiar plenamente en los resultados generados por SCAPy para el análisis de circuitos complejos. Es importante destacar que la resolución de los sistemas de ecuaciones en SCAPy se lleva a cabo utilizando el algoritmo DDD por renglones, un método que optimiza el proceso de cálculo simbólico y asegura una solución eficiente y exacta. Este enfoque innovador no solo garantiza la precisión de los resultados, sino que también mejora significativamente la velocidad de procesamiento, posicionando a SCAPy como una herramienta avanzada y confiable en el ámbito de la simulación de circuitos.

Capítulo 5

Conclusiones

5.1. Sumario

En el presente trabajo de tesis se pudo estudiar el estado del arte destacando herramientas gratuitas y de paga que permiten analizar de manera simbólica circuitos eléctricos lineales, entre las más destacadas se encontró SCAM, ESCAM y Lcapy, una vez contempladas las herramientas existentes se analizaron sus alcances y limitaciones de cada una, también se observaron los resultados obtenidos y al momento de las pruebas se encontraron algunas incongruencias en los resultados por Lcapy, estas se presentaron principalmente en la formulación matemática de los circuitos devuelta por el algoritmo de parseo, siendo una razón más para continuar con el desarrollo de SCAPy.

Después se analizó una comparativa entre MATLAB y Python, se encontraron cualidades muy buenas descritas en el capítulo 2.1; sin embargo, para este punto era un hecho que se requería un algoritmo de solución, ya que al momento solo se contaba con los métodos de solución por algunas librerías de cálculo simbólico como sympy, estos métodos son muy buenos y precisos; sin embargo, el tiempo de cómputo para converger a una solución es muy extenso incluso para procesadores de gama alta, siendo computacionalmente costos

Teniendo un punto de partida en el que se requería un algoritmo de parseo y uno de solución, se presenta toda la metodología del capítulo 2.3, para este punto se hace mención que el núcleo del parser se fundamenta en el método MNA (Análisis Nodal Modificado) por sus siglas en inglés, y partiendo de esta formulación se puede hacer un resumen de llenado presentado como método de los STAMPS, se hicieron muchos ejemplos del llenado de la matriz MNA y los más destacados fueron descritos en la sección 2.3.3 de este presente trabajo

Con un algoritmo de parseo sólido se hicieron pruebas con sympy para obtener los primeros resultados simbólicos y observar si el llenado por el algoritmo que permite formular el modelo matemático en la matriz MNA (parseo) es correcto se hicieron algunas pruebas y se decidió que SCAPy también tuviera implementado los métodos de solución simbólica de sympy, posteriormente se presenta DDD (Diagrama de Decisión a Determinantes), y dos variantes (Greedy y capas) pasando por la idea principal BDD para desarrollar DDD, de acuerdo con el número de operaciones presentado en la sección 3.2 se presenta la implementación del algoritmo DDD por capas en la sección 3.2.3 para la solución a la formulación MNA

Después de tener un algoritmo de solución se presenta la validación con ejemplos de la

sección 4 de este trabajo, midiendo el tiempo de cómputo, memoria y trabajando con circuitos poco convencionales como el integrador fraccional, filtros LTA-WTA y el memristor

Con este trabajo de tesis se espera aportar una herramienta gratuita y poderosa para el análisis de circuitos eléctricos, principalmente a alumnos que se encuentran cursando materias del área.

Como pudo observarse obtener el modelo de ciertos circuitos es una tarea complicada de obtener manualmente, como es el caso del memristor, con esta herramienta se pudo obtener un modelo matemático que describe perfectamente el funcionamiento de este componente, este modelo puede implementarse en diferentes áreas de investigación como el cómputo neuromórfico, también permite conocer que pasa con otros modelos que pueden ser complicados de analizar como es el integrador fraccional o los filtros WTA-LTA.

5.2. Contribuciones originales

Como pudo observarse, los resultados de simulación son muy buenos, cuando se hizo la comparativa entre simulaciones numéricas y los resultados obtenidos por SCAPy, prácticamente el gráfico es el mismo entre las diferentes simulaciones; sin embargo, si hay algo importante a destacar es que todos los elementos de SCAPy son ideales, y esto implica soluciones simbólicas ideales, a diferencia de los simuladores numéricos donde se tienen modelos exclusivos para ciertos elementos, permitiendo obtener resultados más finos; sin embargo, el trabajo realizado hasta ahora es de gran utilidad para una de las metas planteadas que refiere el ofrecer una herramienta gratuita de libre acceso para la comunidad estudiantil, todos aquellos estudiantes que desean aprender, entender el análisis de circuitos eléctricos electrónicos lineales

Los resultados simbólicos se pueden ir mejorando implementando dispositivos con una descripción específica para cada modelo de cada dispositivo comercial, entonces las salidas simbólicas serían más reales para cada dispositivo comercial que se desee computar

Los resultados obtenidos presentan una gran importancia en diferentes aspectos como:

- Formulación matemática devuelta por el algoritmo de parseo resulta ser esencial para la obtención de resultados fiables y certeros.
- El algoritmo de solución DDD obtuvo un excelente rendimiento en tiempo, computacionalmente es muy barato en comparación con las soluciones simbólicas disponibles en Python.
- Los circuitos no convencionales resultaron interesantes para su análisis y comprensión en el rendimiento de SCAPy, en los resultados simbólicos como el modelo del memristor.

5.3. Trabajo futuro

Posibles trabajos futuros podrían implementar elementos no lineales como el diodo y el transistor esta área de investigación aún queda abierta para futuros interesados que quieran implementar más herramientas como la incorporación del análisis en el tiempo o análisis en el dominio de la frecuencia de manera automatizada con la propuesta de algoritmos, y la implementación de elementos no lineales.

Hasta ahora se puede enriquecer en un trabajo futuro, la implementación de modelos específicos para dispositivos comerciales en donde se incluyan parámetros propios del modelo de cada dispositivo comercial, otra manera de enriquecer este trabajo es dándolo a conocer a estudiantes de licenciatura y obtener una retroalimentación para implementar recomendaciones que pudieran ser valiosas y medir la eficiencia en la curva de aprendizaje de circuitos eléctricos y electrónicos.

Apéndice A: Repositorios

5.4. SCAPy

Todo el código de SACAPy puede encontrarse en el repositorio https://github.com/luisCorl/e2scapy desde github, adicionalmente el paquete disponible desde https://pypi.org/project/e2scapy/

5.5. DDD

Este algoritmo se encuentra disponible en el repositorio de github: https://github.com/luisCorl/ddd_layer y el paquete para instalar se encuentra disponible directamente en pypi: https://pypi.org/project/ddd-layer/

5.6. Ejemplos

Se han preparado ejemplos para visualizar el funcionamiento de SCAPy en los que se muestra el computo del memristor e integrador fraccional que se pueden encontrar directamente en el repositorio https://github.com/luisCorl/E2SCAPy-examples, un ejemplo más sobre un horno por arco eléctrico computado en SCAPy se encuentra disponible en: https://github.com/luisCorl/eaf-simbolico

Apéndice B: Manual de usuario

¡Bienvenido a SCAPy!. En el presente manual se borda:

- Introducción.
- Instalación de SCAPy.
- Consideraciones a tener en cuenta, si se desea obtener un netlist a partir de LTSPICE.
- Obtener un Netlist desde LTSPICE.
- Computar un circuito con SCAPy a partir del método DDD por capas.
- Computar un circuito con SCAPy a partir del método simbólico de SymPy.
- Recomendaciones si se desea obtener el análisis en el dominio del tiempo con una ejemplificación con el circuito bicuadrado de Tow-Thomas.

SCAPy cuenta con una licencia GNU GPLV3 que puede ser consultada desde el repositorio: https://github.com/luisCorl/e2scapy/blob/main/LICENSE.txt y que no se coloca en este manual por exención de hojas, pero se invita al lector a revisar si así lo desea. También se invita al lector ponerse en contacto por recomendaciones, bugs detectados, preguntas sobre el funcionamiento y o colaboraciones a través del contacto de GitHub o directamente en el correo: lui.corl.ing@hotmail.com.

5.7. Introducción

SCAPy es una librería innovadora y potente diseñada para el lenguaje de programación Python, que permite la resolución simbólica de circuitos lineales. A diferencia de los métodos tradicionales que proporcionan valores numéricos para el voltaje o la corriente en un elemento o nodo específico, SCAPy ofrece expresiones matemáticas detalladas que describen estas magnitudes en el dominio de la frecuencia. Esto proporciona una comprensión más profunda y completa del comportamiento del circuito. Además, se presenta el método de transformar las expresiones del dominio de la frecuencia al dominio del tiempo, lo que amplía su utilidad y aplicabilidad en el análisis de sistemas dinámicos y en el estudio de respuestas transitorias.

En este manual se incluyen una serie de scripts cuidadosamente seleccionados que ilustran diversos ejemplos de cálculo y métodos de visualización de resultados, facilitando así el proceso de aprendizaje y aplicación de SCAPy. Estos ejemplos no solo demuestran las capacidades de la librería en términos de resolución simbólica y análisis de circuitos, sino que también

muestran cómo graficar las respuestas del sistema de manera efectiva. Además de los ejemplos proporcionados aquí, se ha preparado una colección más extensa de ejemplos prácticos que están disponibles para los usuarios en el repositorio de GitHub, que se puede encontrar en GitHub.https://github.com/luisCorl Este repositorio es un recurso valioso para cualquier lector que desee explorar y utilizar plenamente las capacidades de SCAPy en diversos escenarios de análisis de circuitos.

Además en el repositorio dado se pueden encontrar los ejemplos de los resultados mostrados en esta Tesis y algunos ejemplos adicionales como el modelo EAF.

5.8. Instalación

Para poder usar SCAPy se puede obtener directamente de las siguientes URL: https://pypi.org/project/ddd-layer/ para acceder a la descarga del algoritmo DDD, y la siguiente URL es para acceder al algoritmo SCAPy: https://pypi.org/project/e2scapy/, la forma de hacer la instalación es por el comando pip, adicionalmente si es la primera vez que se instala se recomienda instalar o revisar que se tengan las siguientes paqueterías previamente instalas:

- symengine
- memory_profiler
- sympy
- pandas
- numpy
- matplotlib

Todas estas librerías también pueden ser obtenidas a través del comando *pip*, en la figura 5.1 se muestra el procedimiento para descargar varios paquetes incluyendo ddd_layer y SCAPy, Una vez completada la instalación de todas las librerías requeridas se puede hacer una prueba de importación para comprobar que todo esté en orden la figura 5.1 muestra la correcta importación o invocación de la librería sin ningún error:

```
[7]  pip install e2scapy

Requirement already satisfied: e2scapy in /usr/local/lib/python3.10/dist-packages (0.0.2)

[8]  pip install symengine

Requirement already satisfied: symengine in /usr/local/lib/python3.10/dist-packages (0.11.0)

     pip install ddd_layer

Requirement already satisfied: ddd_layer in /usr/local/lib/python3.10/dist-packages (0.0.2)

[12] pip install memory_profiler

Requirement already satisfied: memory_profiler in /usr/local/lib/python3.10/dist-packages (0.61.0)
Requirement already satisfied: psutil in /usr/local/lib/python3.10/dist-packages (from memory_profiler) (5.9.5)

[14] from e2scapy import e2scapy as sc
     #no olvides conectar con Google drive o asegurarte de que
     #se lee correctamente el archivo filtroRC.cir
     #/content/drive/MyDrive/Colab Notebooks/filtroRC.cir

     sc.MNAf("/content/drive/MyDrive/Colab Notebooks/filtroRC.cir")
     A,x,z = sc.formula_sympy()

DDD15V3
estamos verificando..
  elemento  nodo1 +  nodo2 - nodo3 + nodo4 -  miu miu2 miu3   t1   t2
0       R1        1        2     NaN     NaN  NaN  NaN  NaN  NaN  NaN
1       C1        2        0     NaN     NaN  NaN  NaN  NaN  NaN  NaN
2       V1        1        0     NaN     NaN  NaN  NaN  NaN  NaN  NaN
```

Figura 5.1: Descarga de paquetes mediante el comando pip

5.9. Como dibujar un circuito

Una vez que se tiene todo lo necesario para poder trabajar, se recomienda tener instalado LTspice o bien obtenerlo desde la siguiente URL: https://www.analog.com/en/resources/design-tools-and-calculators/ltspice-simulator.html ahora se muestra como obtener las ecuaciones simbólicas del presente circuito que es un amplificador sumador inversor:

Paso 1 dibujar el circuito en LTSpice como se muestra en la figura 5.2 muestra el inciso (a) como debe dibujarse un circuito contemplando el número de nodos y recordando que todos los nodos conectados a tierra corresponden con el nodo número ”0”

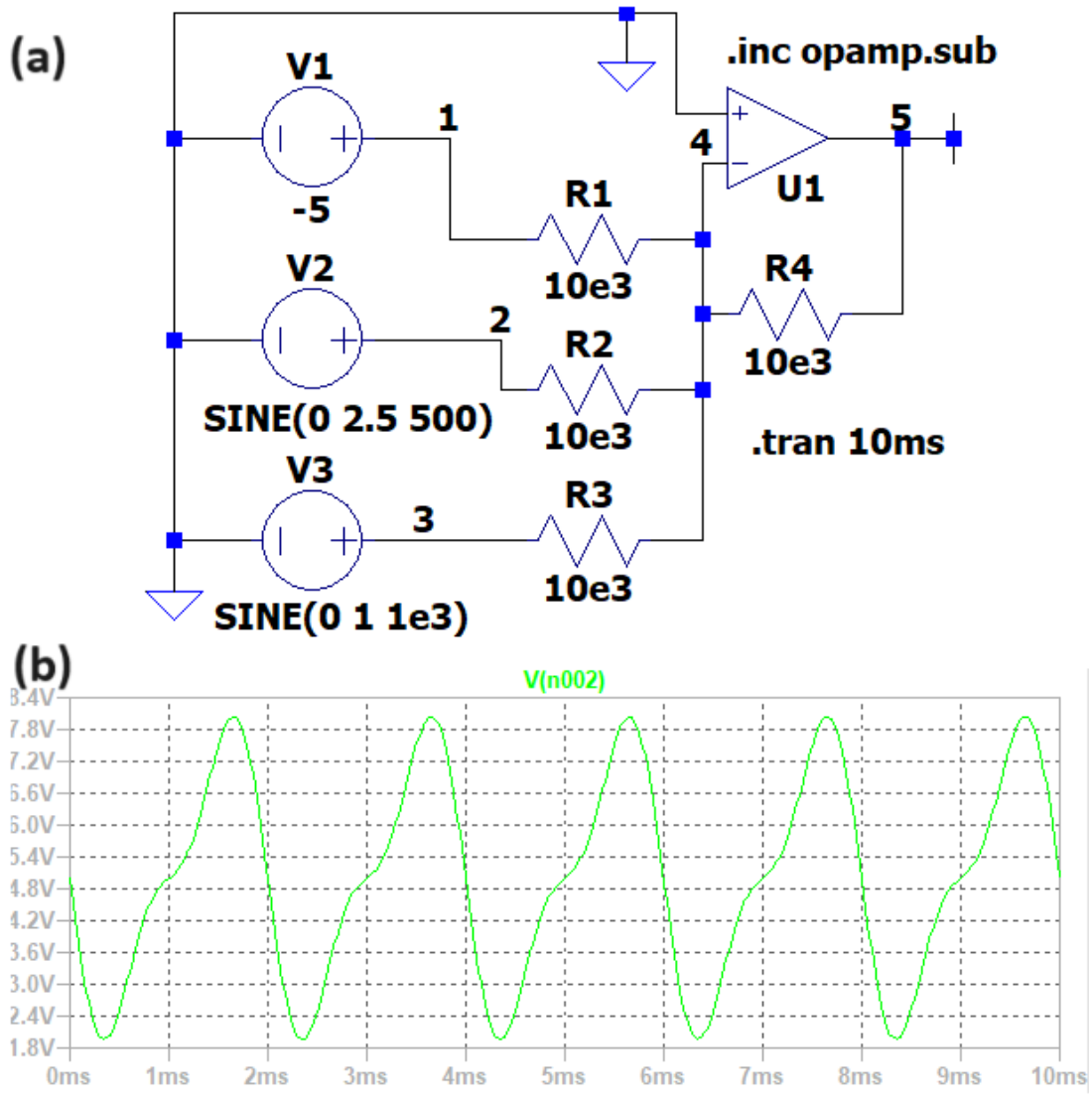


Figura 5.2: Circuito sumador inversor de tensión, (a) muestra el circuito dibujado en LTspice, (b) muestra el grafico de salida en el nodo numero 5

5.10. Como obtener un netlist de LTspice

Para obtener el netlist a partir de la figura 5.2 (a) para obtener el netlist de un circuito en LTspice recordando tendrán que estar enumerados todos los nodos a los que se encuentren conectados los elementos pasivos, entonces estando en el esquemático (no en el gráfico) dar clic en view/SPICE Netlist como se muestra en la figura 5.3

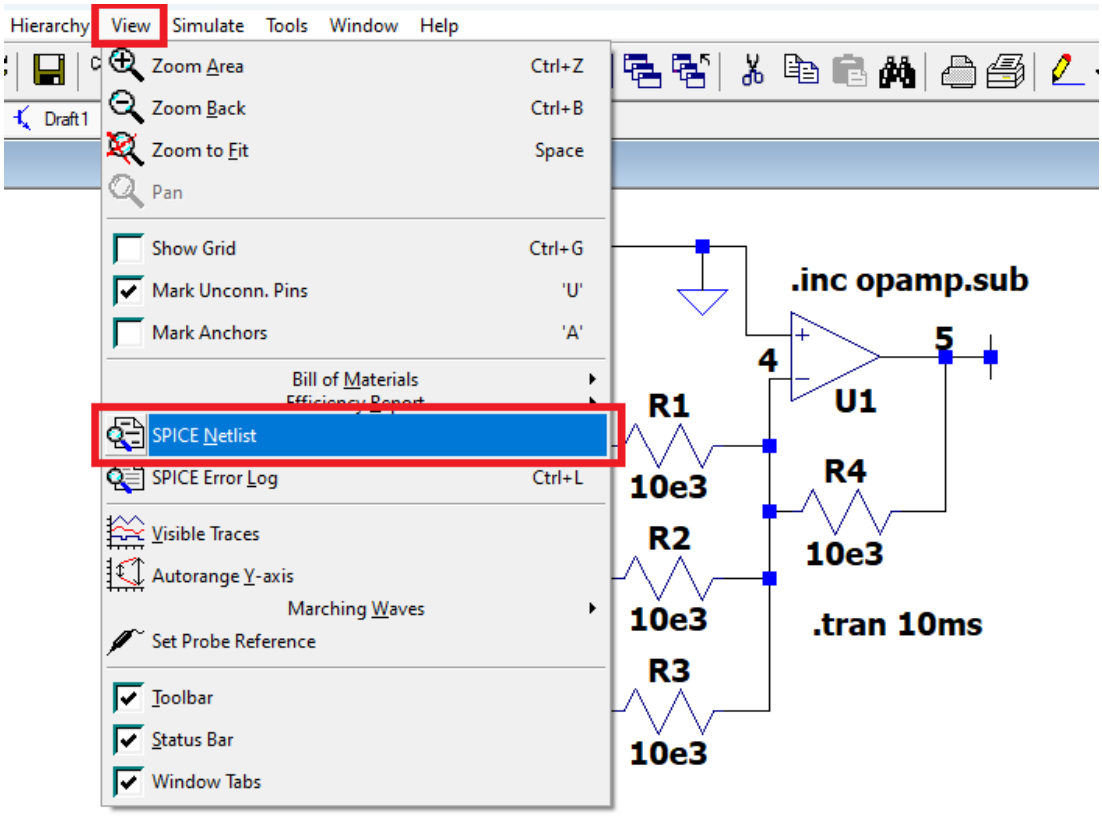


Figura 5.3: Obtención de netlist

A continuación se muestra la siguiente ventana que devuelve el netlist, sin embargo solo se seleccionan todos los elementos lineales que conforman el circuito omitiendo .inc .tran .backanno .end que son instrucciones adicionales para LTspice.

```
XU1 4 0 5 opamp Aol=100K GBW=10Meg
R1 4 1 10e3
R2 4 2 10e3
R3 4 3 10e3
R4 5 4 10e3
V1 1 0 -5
V2 2 0 SINE(0 2.5 500)
V3 3 0 SINE(0 1 1e3)
.inc opamp.sub
.tran 10ms
.backanno
.end
```

Figura 5.4: Netlist LTspice

5.11. Cómo preparar un netlist para SCAPy

Este paso consiste en guardar el netlist en formato .cir y asignar el nombre de los elementos con el formato compatible, recordando que si se quiere:

- OpAmp = O
- Resistor = R
- Inductor = L
- Capacitor = C
- Fuente de voltaje = V
- Fuente de corriente = I
- Fuente de voltaje controlada por voltaje (VCVS)= E
- Fuente de corriente controlada por voltaje (VCCS)= G
- Fuente de voltaje controlada por corriente (CCVS)= H
- Fuente de corriente controlada por corriente (CCCS) = F
- Transformador = L1, L2, k1, k2
- Girador = J

De este modo se ordena el netlist de la manera siguiente figura 5.5:

```
R1 4 1 10e3
R2 4 2 10e3
R3 4 3 10e3
R4 5 4 10e3
O 4 0 5
V1 1 0 -5
V2 2 0 SINE(0 2.5 500)
V3 3 0 SINE(0 1 1e3)
```

Figura 5.5: Netlist para SCAPy

Se recomienda siempre ordenar los elementos pasivos (R,L,C) y siempre ponerlos en orden ascendente como se muestra en la figura 5.5 y guardar el archivo con formato .cir

5.12. Cómo computar un circuito en SCAPy

Se recomienda tomar el siguiente script como base para hacer ele cálculo simbólico usando DDD en donde el algoritmo hace la formulación matemática y resuelve el sistema de ecuaciones lineales

```
#importamos la librer a:
from e2scapy import e2scapy as sim

#leemos el archivo .cir con el respectivo
#nombre asignado
sim.MNAf("sum_inv.cir")
#se procede a hacer la formulaci n matematica
#con DDD por capas
A,x,z = sim.formula_DDD()
#se le pide al algoritmo DDD que resuelva
#el sistema de ecuaciones lineales
X = sim.resuelve_serie_DDD(A,x,z)
X = sim.simplifica(X)
```

El nodo de interés es el nodo 5 con respecto al circuito eléctrico de la figura 5.2 por tanto, que en el terminal se puede solicitar la información del nodo 5 ubicado en el vector X[4] como se muestra en la figura 5.6

```
tiempo de calculo en serie por DDD:  0.03325080871582031
>>> X[4]
R1·R2·V3 + R1·R3·V2 + R2·R3·V1
-------------------------------
    R1·R2 + R1·R3 + R2·R3

>>>
```

Figura 5.6: Salida nodo 5 mediante el método DDD de la figura 5.2

Se recomienda tomar el siguiente script como base para hacer ele cálculo simbólico usando SymPy en donde el algoritmo hace la formulación matemática y resuelve el sistema de ecuaciones lineales

```
#importamos la librer a:
from e2scapy import e2scapy as sim

#leemos el archivo .cir con el respectivo
#nombre asignado
sim.MNAf("sum_inv.cir")
#se procede a hacer la formulaci n matematica
#con DDD por capas
A,x,z = sim.formula_sympy()
#se le pide al algoritmo GE que resuelva
#el sistema de ecuaciones lineales
X = sim.resuelve_LU(A,x,z)
#soluciones adicionales:
#   X = sim.resuelve_ADJ(A,x,z)
#   X = sim.resuelve_GE(A,x,z)
X = sim.simplifica(X)
```

```
tiempo de calculo en sympy LU:  5.371159791946411
>>> X[4]
R1·R2·V3 + R1·R3·V2 + R2·R3·V1
─────────────────────────────────
    R1·R2 + R1·R3 + R2·R3
>>>
```

Figura 5.7: Salida nodo 5 mediante el método LU de la figura 5.2

5.13. Análisis en el dominio del tiempo

En el caso de tener elementos que contengan información en el dominio de la frecuencia directamente desde la formulación MNA como es el caso del capacitor C_n y el inductor L_n el resultado será propiamente en el dominio de la frecuencia con lo se puede evaluar propiamente en dicho dominio; sin embargo, en muchos análisis importa conocer el resultado simbólico, semi-simbólico o numérico en el dominio del tiempo. Para ello se va a analizar un circuito bicuadrado de Tow Thomas en el dominio del tiempo. Para ello se prepara el siguiente circuito:

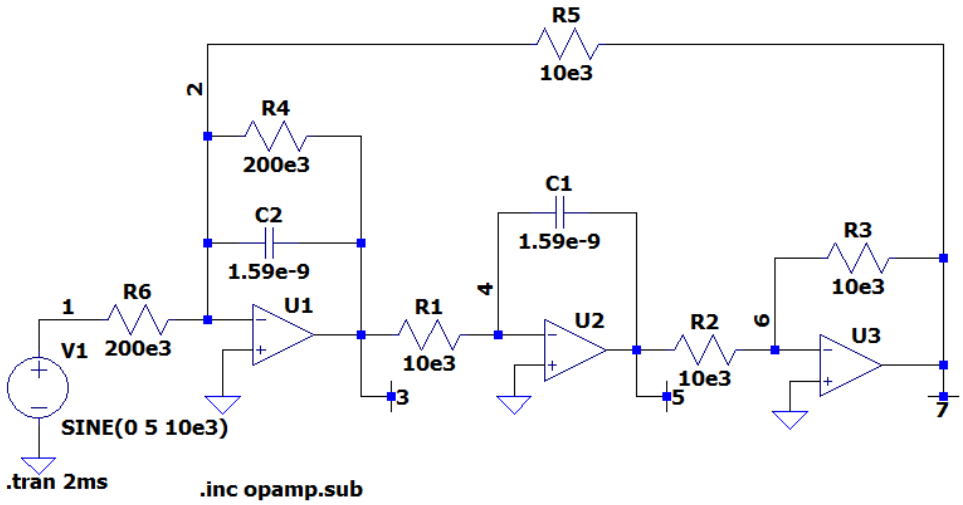


Figura 5.8: Circuito bicuadrado de Tow-Thomas

Los nodos de interés son los nodos 3, 5, 7 se prepara el netlist de la siguiente forma:

```
O1 2 0 3
O2 4 0 5
O3 6 0 7
R1 4 3 10e3
R2 6 5 10e3
R3 7 6 10e3
C1 5 4 1.59e-9
C2 3 2 1.59e-9
R4 3 2 200e3
R5 7 2 10e3
R6 2 1 200e3
V1 1 0 SINE(0 5 10e3)
```

Se guarda el presente netlist en formato.cir y se prepara el script eligiendo el método de solución por DDD:

```
#importamos la librer a:
from escapy import scapy as sim

#leemos el archivo .cir con el respectivo
#nombre asignado
sim.MNAf("TOW THOMAS.cir")
#se procede a hacer la formulaci n matematica
#con DDD por capas
A,x,z = sim.formula_DDD()
#se le pide al algoritmo DDD que resuelva
#el sistema de ecuaciones lineales
X = sim.resuelve_serie_DDD(A,x,z)
X = sim.simplifica(X)
```

En la terminal se preparan las funciones de transferencia dividiendo la información del vector $H3 = X[2]/V1$, $H5 = X[4]/V1$ y $H7 = X[6]/V1$ como se muestra en la figura:

```
tiempo de calculo en serie por DDD:  0.5699453353881836
>>> x
    [[V_1]
     [V_2]
     [V_3]
     [V_4]
     [V_5]
     [V_6]
     [V_7]
    [Iv_1]
   [I_OAmp_1]
   [I_OAmp_2]
   [I_OAmp_3]]
>>> print(X[2]/X[0])
C1*R1*R2*R4*R5*s/(R6*(-C1*R1*R2*R5*s*(C2*R4*s + 1) - R3*R4))
>>> print(X[4]/X[0])
R2*R4*R5/(R6*(C1*R1*R2*R5*s*(C2*R4*s + 1) + R3*R4))
>>> print(X[6]/X[0])
R3*R4*R5/(R6*(-C1*R1*R2*R5*s*(C2*R4*s + 1) - R3*R4))
```

H3, H5, H7

Figura 5.9: Funciones de transferencia para los nodos H3, H5 y H7

A partir de las funciones de transferencia se recomienda usar las transformada de Laplace y la transformada inversa de Laplace de MATLAB, hasta presenta mejor convergencia cuando son ecuaciones algebraicas grandes.

```
clc; clear all;
syms s t

R1 = 10e3
R2 = 10e3
R3 = 10e3
R4 = 200e3
R5 = 10e3
R6 = 200e3
C1 = 1.59e-9
C2 = 1.59e-9
```

```
12
13 H3 = C1*R1*R2*R4*R5*s/(R6*(-C1*R1*R2*R5*s*(C2*R4*s + 1) - R3*R4))
14 H5 = R2*R4*R5/(R6*(C1*R1*R2*R5*s*(C2*R4*s + 1) + R3*R4))
15 H7 = R3*R4*R5/(R6*(-C1*R1*R2*R5*s*(C2*R4*s + 1) - R3*R4))
16 w = 2*3.1416*10e3
17 vin = 5*sin(w*t)
18 vin = laplace(vin,t,s)
19 v3s = vin*H3;
20 v5s = vin*H5;
21 v7s = vin*H7;
22 v3t = ilaplace(v3s,s,t)
23 v5t = ilaplace(v5s,s,t)
24 v7t = ilaplace(v7s,s,t)
```

Cómo puede observarse es importante transformar la fuente de excitación al dominio de la frecuencia para posteriormente obtener el voltaje total en el dominio de la frecuencia y después calcular la transformada inversa de Laplace

Para graficar todo esto en Python se requiere se puede seguir el siguiente procedimiento de acuerdo con el script que se presenta a continuación:

```
1  from matplotlib import pyplot as plt
2  import numpy as np
3
4  # Generar se al de entrada V1
5  T = np.linspace(0, 0.002, 600)
6  V_1 = []
7  ampl = 0.5
8  f = 10e3
9  w = 2 * np.pi * f
10 for t in T:
11     tempo = ampl * np.sin(w * t)
12     V_1.append(tempo)
13
14
15 from numpy import sin, cos, exp, pi
16
17 def fv3(t):
18     v3 = (25177073981803590908441542337484240471164190720*exp
       (-(4611686018427387904*t)/2933032307719819)*(cos((68719476736*159**(1/2)
       *1144999672862302388011**(1/2)*t)/466352136927451221) +
       (9662018248046651403629057812725 76*159**(1/2)*1144999672862302388011**(1/2)
       *sin((68719476736*159**(1/2)*1144999672862302388011**(1/2)*t)
       /466352136927451221))/160181680102909130250247911627102837954882 01))
       /12975111684348524948804390054269758351775182933 7 -
       (64777702983050847111369040606228680223204835328 0*sin(62832*t))
       /12975111684348524948804390054269758351775182933 7 -
       (25177073981803590908441542337484240471164190720*cos(62832*t))
       /12975111684348524948804390054269758351775182933 7
19     return v3
20
21 def fv5(t):
22     v5 = (25201549726898354590419758006460114534400000000*sin(62832*t))
       /12975111684348524948804390054269758351775182933 7 -
       (64840676247822732549220843889814468034560000000 0*cos(62832*t))
       /12975111684348524948804390054269758351775182933 7 +
```

```
    (648406762478227325492208438898144680345600000000*exp
    (-(4611686018427387904*t)/2933032307719819)*(cos((68719476736*159**(1/2)
    *1144999672862302388011**(1/2)*t)/466352136927451221) -
    (146137487871216261357315111992138518 81*159**(1/2)
    *1144999672862302388011**(1/2)*sin((68719476736*159**(1/2)
    *1144999672862302388011**(1/2)*t)/466352136927451221))
    /450746218921560165635904159781495586339100454551552))
    /129751116843485249488043900542697583517751829337
    return v5
def fv7(t):
    v7 = (648406762478227325492208438898144680345600000000*cos(62832*t))
    /129751116843485249488043900542697583517751829337 -
    (252015497268983545904197580064601145344000000000*sin(62832*t))
    /129751116843485249488043900542697583517751829337 -
    (648406762478227325492208438898144680345600000000*exp
    (-(4611686018427387904*t)/2933032307719819)*(cos((68719476736*159**(1/2)
    *1144999672862302388011**(1/2)*t)/466352136927451221) -
    (146137487871216261357315111992138518 81*159**(1/2)
    *1144999672862302388011**(1/2)*sin((68719476736*159**(1/2)
    *1144999672862302388011**(1/2)*t)/466352136927451221))
    /450746218921560165635904159781495586339100454551552))
    /129751116843485249488043900542697583517751829337
    return v7

# Inicializar listas para V3, V5 y V7
V3 = []
V5 = []
V7 = []
# C lculo del modelo
for t in T:
    v3 = fv3(t)
    v5 = fv5(t)
    v7 = fv7(t)

    V3.append(v3)
    V5.append(v5)
    V7.append(v7)
# Graficar resultados
fig, axs = plt.subplots(1, 1, figsize=(11, 3))
axs.xaxis.set_major_formatter(plt.FormatStrFormatter('%.3f'))
axs.plot(T, V3, color="green", label="v3")
axs.plot(T, V5, color="blue", label="v5")
axs.plot(T, V7, color="red", label="v7")
axs.set_title("Input voltage")
axs.set_xlabel("time s")
axs.set_ylabel("Volts")
axs.legend(loc="lower right")
axs.grid()

plt.show()
```

A continuación se muestra el gráfico de salida

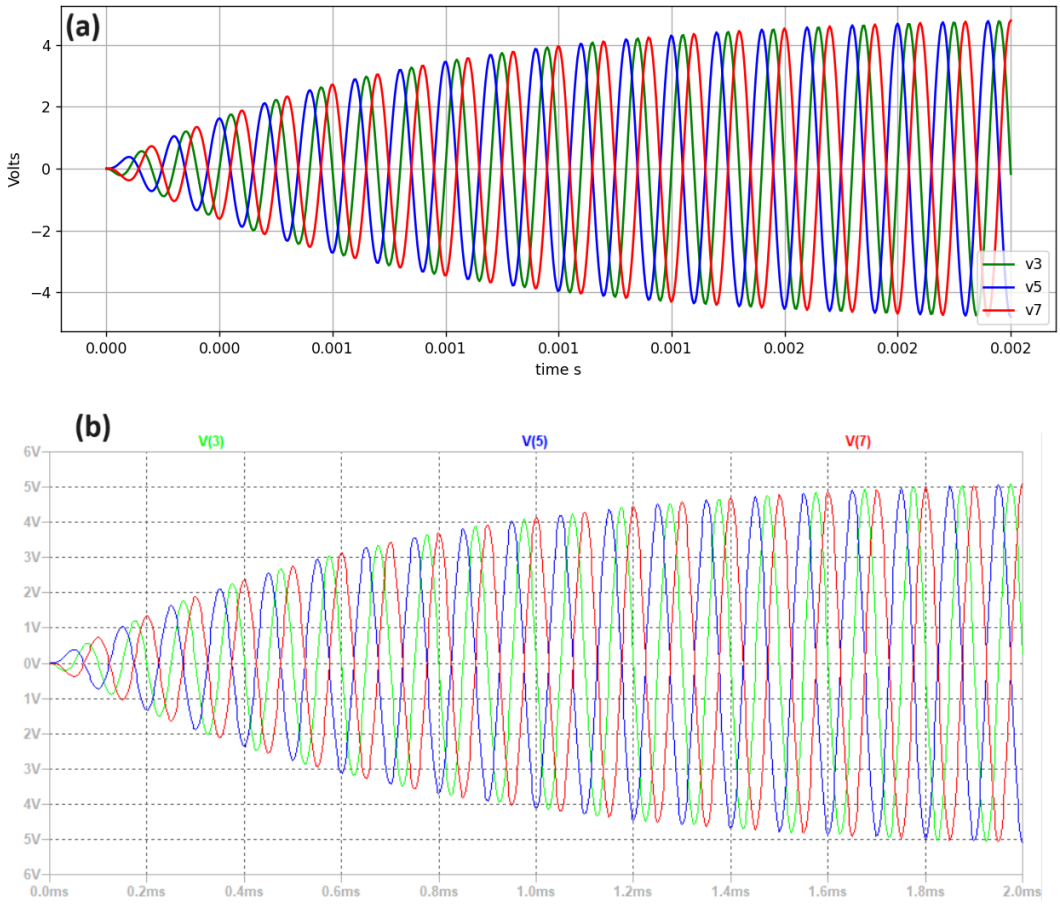


Figura 5.10: Gráfico comparativo entre el gráfico obtenido con el script de Python (a) y el gráfico obtenido en LTspice (b)

Apéndice C

Operating System	Windows 11 Windows 10 (version 20H2 or higher) Windows Server 2019 Windows Server 2022
Processor	**Minimum:** Any Intel or AMD x86–64 processor **Recommended:** Any Intel or AMD x86–64 processor with four logical cores and AVX2 instruction set support **Note:** A future release of MATLAB will require a processor with AVX2 instruction set support
RAM	**Minimum**: 4 GB **Recommended**: 8 GB
Storage	3.8 GB for just MATLAB 4-6 GB for a typical installation 23 GB for an all products installation An SSD is strongly recommended
Graphics	No specific graphics card is required, but a hardware accelerated graphics card supporting OpenGL 3.3 with 1GB GPU memory is recommended. GPU acceleration using Parallel Computing Toolbox requires a GPU with a specific range of compute capability. For more information, see GPU Computing Requirements.

Figura 5.11: Hardware requerido para MATLAB

Bibliografía

[1] Python. Python. https://www.python.org/, 20 de Septiembre de 2022, 2022.

[2] Numpy. Numpy. https://numpy.org/, 24 de Septiembre de 2022, 2022.

[3] Sympy. Sympy. https://www.sympy.org/en/index.html, 20 de Septiembre de 2022, 2021.

[4] A Roldán Aranda and JB Roldán Aranda. Álisis simbólico de circuitos mediante técnicas de análisis nodal modificado. 1970.

[5] year=2012 Francisco V. Fernandez, Angel Rodriguez-Vazquez, Jose L. Huertas, Georges G. E. Gielen. Symbolic analysis techniques.

[6] Mourad Fakhfakh. Design of analog circuits through symbolic analysis. 2012.

[7] Ralf Sommer, Eckhard Hennig, Manfred Thole, Thomas Halfmann, and Tim Wichmann. Analog insydes 2–new features and applications in circuit design. *Proc. SMACD*, 2000.

[8] Arturo Corona Nieva. *EI SCAM, Herramienta EDA para el análisis simbólico y de sensibilidad de circuitos analógicos y de RF así como la evaluación de ruido en circuitos de interfaz.* PhD thesis, UNIVERSIDAD POLITÉCNICA DE PUEBLA, 2016.

[9] Antonio Luchetta, Stefano Manetti, and Alberto Reatti. Sapwin-a symbolic simulator as a support in electrical engineering education. *IEEE Transactions on Education*, 44(2):9, 2001.

[10] Luis A Sánchez-Gaspariano, Israel Vivaldo-de-la Cruz, Jesús M Muñoz-Pacheco, Luz del Carmen Gómez-Pavón, and Arnulfo Luis Ramos. Ei-scam as a teaching tool in an undergraduate course in analog and high-frequency circuits. *Computer Applications in Engineering Education*, 30(4):1022–1035, 2022.

[11] Circuit Magic. `https://circuit-magic.com/`,24 de Septiembre de 2022, 2003.

[12] Lcapy. Lcapy. https://lcapy.readthedocs.io/en/latest/, 24 de Septiembre de 2022, 2014–2022.

[13] Erik Cheever. Scam: Symbolic circuit analysis in matlab, 2005 to 2022. Accessed on 2024.

[14] Cadence, Eda software. `http://www.cadence.com/en/default.aspx`,20 de Septiembre de 2022, 2016l.

[15] T. Tanaka. Ceyhun ozgur;taylor colliau;grace rogers;zachariah hughes;elyse bennie myertyson. *Journal of Data Science*, pages 355–371, 2017.

[16] fangohr@soton.ac.uk Hans Fangohr. A comparison of c, matlab, and python as teaching languages in engineering. 2012.

[17] circuitnav.pythonanywhere.com. `circuitnav.pythonanywhere.com`,20 de Septiembre de 2022, 2021.

[18] T. Tanaka. Parsing electronic circuits in a logic grammar. *IEEE Transactions on Knowledge and Data Engineering*, 5(2):225–239, 1993.

[19] Analog.com. `https://www.analog.com/media/en/simulation-models/spice-models/ltspicegettingstartedguide.pdf?modelType=spice-models`,24 de Sabril de 2023, 2011.

[20] Esteban Tlelo Cuautle Guoyong Shi. Sheldon X.-D.Tan. *Advanced Symbolic Analysis for VLSI Systems Methods and Aplications*.

[21] C-J Richard Shi and Xiangdong Tan. Symbolic analysis of large analog circuits with determinant decision diagrams. In *Proceedings of the 1997 IEEE/ACM international conference on Computer-aided design*, pages 366–373, 1997.

[22] W. Verhaegen and G. Gielen. Efficient ddd-based symbolic analysis of linear analog circuits. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 49(7):474–487, 2002.

[23] Efraín Rodríguez López. *Análisis simbólico de sistemas LPV basado en diagramas de desisión para determinantes: una aplicación de control predictivo.* PhD thesis, Universidad Michoacana de San Nicolás de Hidalgo, 2020.

[24] Per Lindgren, Mikael Kerttu, Mitch Thornton, and Rolf Drechsler. Low power optimization technique for bdd mapped circuits. In *Proceedings of the 2001 Asia and South Pacific Design Automation Conference*, pages 615–621, 2001.

[25] G. Gielen, P. Wambacq, and W.M. Sansen. Symbolic analysis methods and applications for analog circuits: a tutorial overview. *Proceedings of the IEEE*, 82(2):287–304, 1994.

[26] Carlos Muñiz-Montero, Luis V García-Jiménez, Luis A Sánchez-Gaspariano, Carlos Sánchez-López, Víctor R González-Díaz, and Esteban Tlelo-Cuautle. New alternatives for analog implementation of fractional-order integrators, differentiators and pid controllers based on integer-order integrators. *Nonlinear Dynamics*, 90:241–256, 2017.

[27] Luis Abraham Sánchez Gaspariano, Carlos Muñiz Montero, Efraín G Cuautle Zacatelco, Francisco R Trejo Macotela, Fernando Osvaldo González Manzanilla, Mario Espinosa Tlaxcaltecatl, et al. Filtros no lineales de máxima y mánima en modo voltaje basados en seguidores de voltaje anidados. *REPOSITORIO NACIONAL CONACYT*, 2012.

[28] CH Phang, YT Yeow, RA Barham, and PJ Allen. Measurement of hybrid pi equivalent circuit parameters of bipolar junction transistors in undergraduate laboratories. *IEEE Transactions on Education*, 40(3):213–218, 1997.

[29] V. Mladenov and S. Kirilov. An improved memristor model and applications. In *2023 12th International Conference on Modern Circuits and Systems Technologies (MOCAST)*, pages 1–4, Athens, Greece, 2023. IEEE.

TOEFL ITP Score Report

Name of Institution: CETEC

Name: CORTES RAMIREZ LUIS Student Number: 108042

DOB: 08/01/1994 Sex: M Degree: Times Taken TOEFL: Two

Native Country: Mexico
Native Language: Spanish

Scaled Scores:

Listening Comprehension:	57
Structure & Written Expression:	56
Reading Comprehension:	57
Total Score:	567

Test Date: 07/08/2024
Form: TOEFL ITP

ETS TOEFL.ITP

Student's File Copy
Do Not Copy

The face of this document has a security background. The back contains a watermark. Hold at an angle to view.

The *TOEFL.ITP* Assessment Series is designed to be used for placement, progress monitoring, and exit purposes. TOEFL.ITP scores can also be used for admissions to programs and institutions where English is not the dominant language of instruction for content courses. Learn more at **www. ets.org/toefl_itp/use.**

97824-16573• FB313R200• Printed in U.S.A. I.N. 770462

ACADEMIA JOURNALS
OPUS PRO SCIENTIA ET STUDIUM

SAN ANTONIO, TX
CONTACTO@ACADEMIAJOURNALS.COM
ACADEMIAJOURNALS.COM

11 abril 2024

AUTORES: Ing. Luis Cortés Ramírez
Dr. Luis Abraham Sánchez Gaspariano
Dr. Carlos Leopoldo Pando Lambruschini

ARTÍCULO: **Análisis de Horno EAF Usando ESCAP y una Herramienta EDA para el Análisis Simbólico**

ARTÍCULO Núm: FRS055

Estimados autores,

Con agrado les informamos que, con fecha de hoy, el artículo arriba citado ha sido aprobado para su presentación en el Congreso International de Investigación Academia Journals Abril 2024, *Avances Multidisciplinarios para una Sociedad Innovadora*. El congreso evento totalmente virtual tendrá lugar los días 22 y 23 de abril en el portal www.academiajournals.com.

El artículo será incluido en las publicaciones del congreso, que incluyen modalidades ISBN, ISSN, e indización en *Fuente Académica Plus* de EBSCOHost.

1. Volúmenes online con *ISSN 1946-5351, Vol. 16, No. 03 online,* e indexación en Fuente Académica Plus de EBSCOhost en Estados Unidos.
2. Libros ebook compilados por área temática con números ISBN online.

Saludos cordiales.

Dr. Rafael Moras, P.E.
Editor
Academia Journals
contacto@academiajournals.com

ACADEMIA JOURNALS

OPUS PRO SCIENTIA ET STUDIUM

Tomo 06

Ingenierías

ARTÍCULOS DEL CONGRESO INTERNACIONAL DE INVESTIGACIÓN ACADEMIA JOURNALS ABRIL 2024

ELIBRO ONLINE CON ISSN 1946-5351 | VOL. 16, NO. 03, 2024

Análisis de Horno EAF Usando ESCAP y una Herramienta EDA para el Análisis Simbólico

Ing. Luis Cortés Ramírez[1], Dr. Luis Abraham Sánchez Gaspariano[2],
Dr. Carlos Leopoldo Pando Lambruschini[3]

***Resumen*—**En el presente trabajo de investigación se evalúa la formulación matemática de ESCAPy (Análisis de Circuitos Simbólicos Eléctricos en Python) por sus siglas en inglés, una herramienta EDA (Automatización de Diseño Electrónico) por sus siglas en inglés, para el análisis de cálculo simbólico a partir de un circuito eléctrico descrito en una netlist como entrada al simulador, el circuito de elección es un modelo para horno de arco eléctrico (EAF) para observar y discutir el comportamiento de este en la fundición de acero, y además ejemplificar el uso de esta herramienta desarrollada en el programa de maestría en Ciencias de la Electrónica opción Automatización y que se espera que sea de ayuda a alumnos que cursan materias de teoría de circuitos como refuerzo a la comprensión de estos.

***Palabras clave*—**EDA, MNA, EAF, ESCAPy, Simulación.

Introducción

Horno de fundición por arco eléctrico.

Un horno de fundición por arco eléctrico (EAF) consta de tres electrodos que se introducen en una cámara sellada que contiene chatarra en su interior, para sellar dicha cámara se desliza una tapa con tres orificios para la introducción de electrodos en cada electrodo [Bekker] se inyecta una fase de voltaje proveniente de un transformador, los electrodos generan un arco eléctrico entre la conductancia de la chatarra y los electrodos que funde dicha misma [Odenthal].

Cada electrodo tiene un control de subida y bajada de altura para modular el arco eléctrico, y usualmente la subida y bajada de cada electrodo es modulada por un control hidráulico [Logar], dependiendo del modelo de horno cuando se tiene la fundición en estado líquido es común que haya un proceso de vaciado de escoria, para posteriormente un nuevo recalentamiento del acero líquido y finamente un vaciado del acero líquido en donde usualmente es tratado en aditamento de materiales para propiedades especificas entre otros procesos cómo vacío y colado. En todo momento se tiene una extracción de humos y polvos que resultan ser tóxicos por tal motivo es muy común encontrar un extractor en cada horno de fundición por arco eléctrico como se muestra en la figura 1 [Bekker].

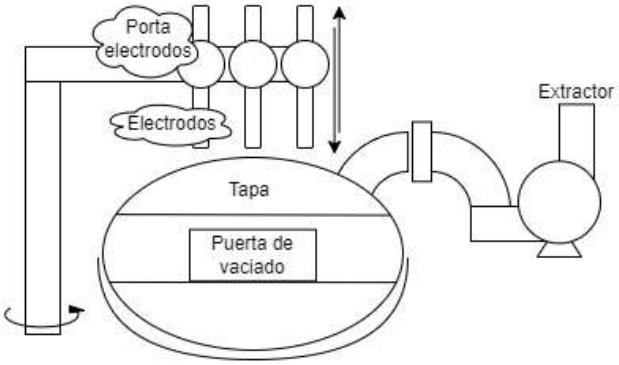


Figura 1. Esquemático general de un EAF.

Cálculo simbólico.

Por otro lado se tiene la introducción a la computación simbólica que hace posible computar operaciones elementales como sumar, restar, multiplicar, dividir y potenciar símbolos desde el punto de vista de objetos, en realidad un símbolo es un objeto [SymPy], existen diferentes librerías que permiten crear objetos con propiedades elementales y más aún, como es el caso de sympy (una librería de cálculo simbólico para Python), sin embargo sympy no es la

[1] Luis Cortés Ramírez es estudiante del programa de maestría por la Benemérita Universidad Autónoma de Puebla. luis.cortesr@alumno.buap.mx **(autor corresponsal)**
[2] El Dr. Luis Abraham Sánchez Gaspariano es profesor investigador por la Benemérita Universidad Autónoma de Puebla perteneciente al sistema nacional de investigadoras e investigadores SNI 1. luis.sanchezgas@correo.buap.mx
[3] El Dr. Carlos Leopoldo Pando Lambruschini es profesor investigador por la Benemérita Universidad Autónoma de Puebla perteneciente al sistema nacional de investigadoras e investigadores SNI 2. carlos@ifuap.buap.mx

única librería disponible, pero en general todo el computo simbólico se fundamenta en hacer operaciones con objetos. Una de las operaciones que mas trabajo le cuesta computacionalmente hablando al procesador por tomar varios ciclos de reloj es la multiplicación y la división, aunque en la actualidad cada CPU cuenta con hardware especializado para operar multiplicaciones y divisiones con la finalidad de no sacrificar tantos ciclos de reloj, hacer una multiplicación o división de objectos es una tarea muy estresante para el procesador, haciendo que el cálculo simbólico sea costoso especialmente cuando se tienen sistemas algebraicos por resolver en donde encontrar la solución por un método conocido como Gauss Jordan o descomposición de matrices triangular inferior y superior (LU), hace que cada objeto se acumule en un área de memoria que a su vez en cada ciclo es nuevamente computada.

Formulación MNA.

La formulación MNA tomando como punto de partida la ley de corrientes de Kirchhoff es bien conocido que para analizar un conjunto de resistencias conectadas a una fuente independiente que puede ser de voltaje o de corriente, siempre se obtiene un sistema de ecuaciones, al ser la resistencia un elemento pasivo lineal, el sistema de ecuaciones resultante es lineal, respetando las propiedades de superposición y escalaridad en el sistema de ecuaciones.

A lo largo del tiempo se implementó esta idea en un método llamado NA (Nodal Analisis) [Fakhfakh] por sus siglas en inglés, este método trabaja con el inverso de la resistencia en elementos pasivos (R, L y C) siendo la conductancia para el resistor y la suceptancia en el capacitor e inductor, que no es otra cosa más que el inverso de la reactancia en dichos elementos, esta idea de trabajar con las conductancias y equivalentes permitió hacer que el sistema de ecuaciones fuera siempre una matriz cuadrada al incorporar un vector extra en la matriz.

Sin embargo nuevas investigaciones permitieron incorporar en dicho sistema de ecuaciones lineales a elementos que no son pasivos pero si lineales como el Amplificador Operacional (OpAmp) por sus siglas en inglés, el transformador, fuentes de voltaje que dependen de una ganancia en el voltaje o corriente de un nodo dado (VCVS, CCVS) respectivamente, fuentes de corriente que dependen de una ganancia en el voltaje o corriente de un nodo dado (VCCS, CCCS) respectivamente, transformadores y algunos elementos ideales como el girador. Esta técnica fue nombrada: Análisis Nodal Modificado [Fakhfakh] [Fernandez], y la matriz resultante de estos elementos es una matriz MNA refiriendo a esta técnica y es llenada bajo un concepto interesante de incorporar cuatro tipos de matrices resultantes de cada elemento como se muestra en la figura 2.

$$MNA = \begin{bmatrix} G & B \\ C & D \end{bmatrix} \quad x = \begin{bmatrix} v \\ j \end{bmatrix} \quad z = \begin{bmatrix} i \\ e \end{bmatrix}$$

Figura 2. Construcción de sistema lineal MNA.

Donde la matriz MNA se consta de una matriz cuadrada G en donde se encuentra la susceptancia y la conductancia de los elementos pasivos, en la matriz C y B se encuentra la información de las fuentes independientes de voltaje traspuestas una de la otra con un uno y también contiene información del amplificador operacional, fuentes dependientes, finalmente en la matriz D es una matriz cuadrada que contiene la información del transformador y girador, a este método se le suma una simplificación por nullators y norators que no se discutirá en este articulo pero que sirven para llenar la matriz ya que la naturaleza del método MNA es una matriz con muchos ceros en los triángulos inferior y superior por la matriz G.

Se tiene el vector de incógnitas x en que se desconoce a “v” (los valores desconocidos de las tenciones nodales) y “j” (el valor desconocido de las corrientes que fluyen en el circuito) como vector de entrada se tiene a z que corresponde al vector “i” (los valores conocidos de las fuentes de corriente dependientes) y vector “e” (los valores conocidos de las fuentes de voltaje dependientes), la figura 3 muestra un integrador para la formulación del método MNA.

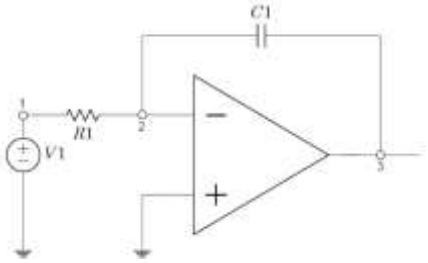


Figura 3. Circuito integrador.

El método consta de poner un número a cada nodo existente iniciando siempre en cero con la tierra, y de 1 … n para los nodos restantes para este caso se tienen dos elementos pasivos correspondientes al resistor y capacitor respectivamente, también se tiene un amplificador operacional y una fuente de voltaje independiente, la formulación para el circuito integrador es la siguiente figura 4.

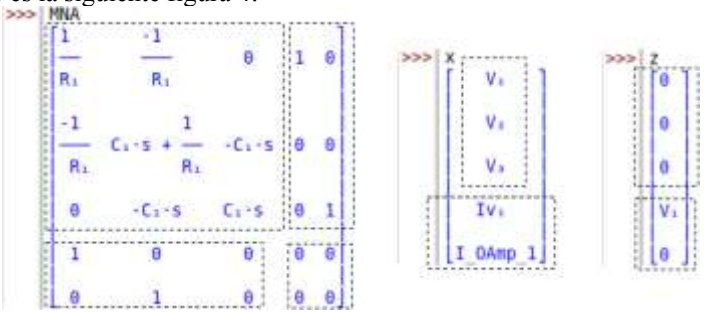


Figura 4. Sistema de ecuaciones lineales.

Comparando la formulación MNA de la figura 4 con la formulación planteada en la figura 2, se puede comparar el orden de las sub matrices que componen el sistema de ecuaciones. Este articulo no pretende abordar el método de solución simbólica, pero si se hace mención que ESCAPy cuenta con técnicas novedosas para la solución de sistemas lineales simbólicos.

Metodología

Procedimiento.

Retomando el modelo eléctrico de [Boulet] como se muestra en la figura 5 dicho modelo comprende las resistencias R1, R2, R3, R4, R5, R6 representando la conductividad eléctrica de la chatarra que se encuentra dentro del horno EAF, las fuentes de voltaje representan los electrodos que a su vez inyectan un voltaje senoidal desfasado a 120° uno con respecto del otro, voltaje que es subministrado por un transformador de alta potencia el voltaje es representativo al igual que la resistencia eléctrica entre la chatarra y que puede variar dependiendo de la homogeneidad de la misma, altura de los electrodos o gases del ambiente que pueden ser o no controlados.

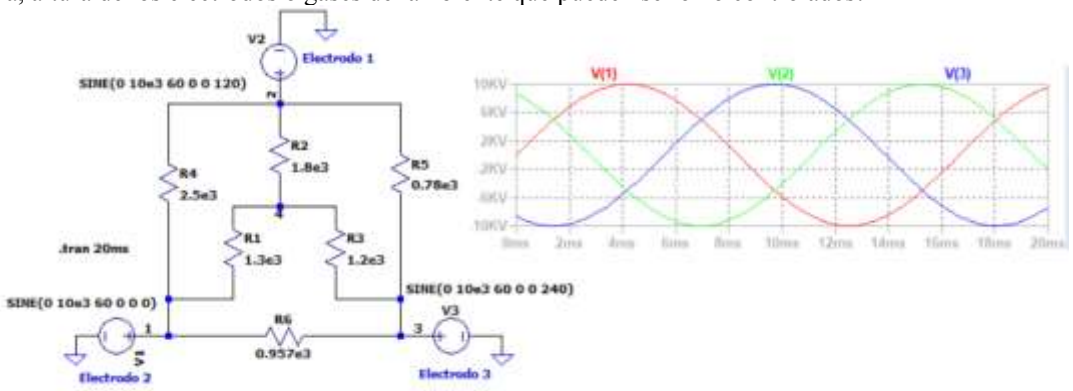


Figura 5. Modelo eléctrico de electrodos.

A su vez el circuito modelo tiene cuatro nodos siendo el nodo numero 4 una incógnita ya que en los nodos 1-3 corresponde con el voltaje de la fuente que representa cada electrodo, figura 6.

R1	4	1
R2	2	4
R3	4	3
R4	2	1
R5	2	3
R6	3	1
V1	1	0
V2	2	0
V3	3	0

Figura 6. Netlist.

El netlist de la figura 6 es guardado con formato ".cir" o bien puede obtenerse directamente de LTspice, a continuación la figura 7 muestra cómo computar el circuito en ESCAPy asumiendo que el nombre del netlist es: "EAF.cir".

```
File  Edit  Format  Run  Options  Window  Help
# Se importa la librería scapy de escapy:
from escapy import scapy

# Se le indica a scapy que se desea leer el
# archivo EAF.cir:
scapy.MNAf('EAF.cir')

# A continuación se formula en A,x,z
# y resyelve en el vector X
A,x,z = scapy.formula_sympy()
X = scapy.resuelve_GE(A,x,z)
X = scapy.simplifica(X)
```

Figura 7. Sintaxis de Python-ESCAPy.

La figura 7 muestra como computar un circuito eléctrico en ESCAPy a partir de un netlist, mientras la figura 6 muestra el netlist a partir de un circuito eléctrico, a su vez el circuito de la figura 5 muestra como nombrar los nodos a analizar en el circuito, adicionalmente se pueden poner valores numéricos en el netlist.

Resultados

De acuerdo con el vector de incógnitas "x" la información de interés puede imprimirse de acuerdo a la posición correspondiente como se muestra en la figura 8.

```
>>> X
[V1]
[V2]
[V3]
[V4]
[Iv1]
[Iv2]
[Iv3]
>>> print('voltaje en el nodo 4:',X[3])
voltaje en el nodo 4: (R1*R2*V3 + R1*R3*V2 + R2*R3*V1)/(R1*R2 + R1*R3 + R2*R3)
>>> print('corriente del nodo 1:',X[4])
corriente del nodo 1: (R4*V3*(R1*R2 + R1*R3 + R2*R3 + R2*R6) + R6*V2*(R1*R2 + R1*R3 + R2*R3 + R3*R4) - V1*(R1*R2*R4
+ R1*R2*R6 + R1*R3*R4 + R1*R3*R6 + R2*R3*R4 + R2*R3*R6 + R2*R4*R6 + R3*R4*R6))/(R4*R6*(R1*R2 + R1*R3 + R2*R3))
>>> print('corriente del nodo 2:',X[5])
corriente del nodo 2: (R4*V3*(R1*R2 + R1*R3 + R1*R5 + R2*R3) + R5*V1*(R1*R2 + R1*R3 + R2*R3 + R3*R4) - V2*(R1*R2*R4
+ R1*R2*R5 + R1*R3*R4 + R1*R3*R5 + R1*R4*R5 + R2*R3*R4 + R2*R3*R5 + R3*R4*R5))/(R4*R5*(R1*R2 + R1*R3 + R2*R3))
>>> print('corriente del nodo 3:',X[6])
corriente del nodo 3: (R5*V1*(R1*R2 + R1*R3 + R2*R3 + R2*R6) + R6*V2*(R1*R2 + R1*R3 + R1*R5 + R2*R3) - V3*(R1*R2*R5
+ R1*R2*R6 + R1*R3*R5 + R1*R3*R6 + R1*R5*R6 + R2*R3*R5 + R2*R3*R6 + R2*R5*R6))/(R5*R6*(R1*R2 + R1*R3 + R2*R3))
```

Figura 8. Resultados simbólicos.

De acuerdo con [Boulet] la conductancia es dada por: $G = C_i * x_i + G_s$ donde C_i es el coeficiente de conductancia dado en $\frac{siemens}{m}$, mientras que x_i es la inmersión del electrodo en la chatarra dado en m, y por último G_s es la conductancia de la chatarra con los electrodos posicionados de este modo se sume la siguiente relación de inmersión para los electrodos.

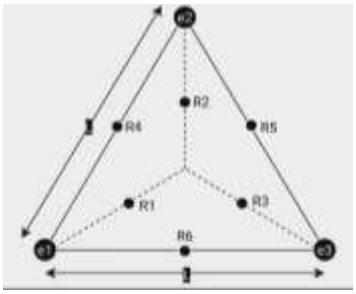


Figura 9. Distribución de electrodos.

De modo que la resistencia es dada por: $R_i = \frac{1}{C_i x_{i+} G_s P}$ donde "P" indica el estado "0" para electrodos fuera de posición y "1" para electrodos en posición, con esta ecuación cada resistencia queda en función de la inmersión del electrodo, y por tanto la corriente de consumo también está en función de la inmersión del de cada resistencia, cada relación de inmersión correspondiente a cada resistencia es mostrada en el cuadro 1:

Inmersión para cada resistencia	parámetros a, b, c, d, e, f
$x_{R1} = \frac{x_{e1}(a+\frac{b}{2})}{a+b} + \frac{x_{e2}(e-f)}{e} + \frac{x_{e3}(e-f)}{e}$	$a = \frac{L}{2}\tan(30)$

$x_{R2} = \frac{x_{e2}(a+\frac{b}{2})}{a+b} + \frac{x_{e1}(e-f)}{e} + \frac{x_{e3}(e-f)}{e}$	$b = sqrt\left(\left(\frac{L}{2}\right)^2 + a^2\right)$
$x_{R3} = \frac{x_{e3}(a+\frac{b}{2})}{a+b} + \frac{x_{e1}(e-f)}{e} + \frac{x_{e2}(e-f)}{e}$	$c = 60 - tan^{-1}(\frac{2a+b}{L})$
$x_{R4} = \frac{x_{e1}(\frac{L}{2})}{L} + \frac{x_{e2}(\frac{L}{2})}{L}$	$d = 120 - c$
$x_{R5} = \frac{x_{e2}(\frac{L}{2})}{L} + \frac{x_{e3}(\frac{L}{2})}{L}$	$e = \frac{L \sin(60)}{\sin(d)}$
$x_{R6} = \frac{x_{e1}(\frac{L}{2})}{L} + \frac{x_{e3}(\frac{L}{2})}{L}$	$f = sqrt(\left(\frac{L}{2}\right)^2 + \left(a+\frac{b}{2}\right)^2)$

Cuadro 1. Relación de inmersión para resistencias.

Las figuras 10 y 11 muestran una comparativa de resultados entre Qucs 0.0.19 en simulación numérica y los resultados obtenidos en ESCAPy y graficados con pyplot, dependiendo de la potencia de cómputo los tiempos de cálculo pueden variar pero se estima que el cálculo simbólico siempre será más lento en converger comparado con un método numérico.

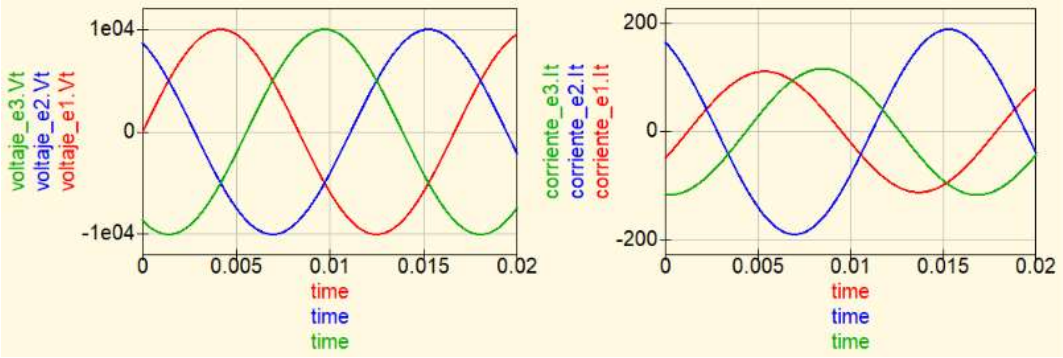


Figura 10. Voltaje y corriente de inmersión.

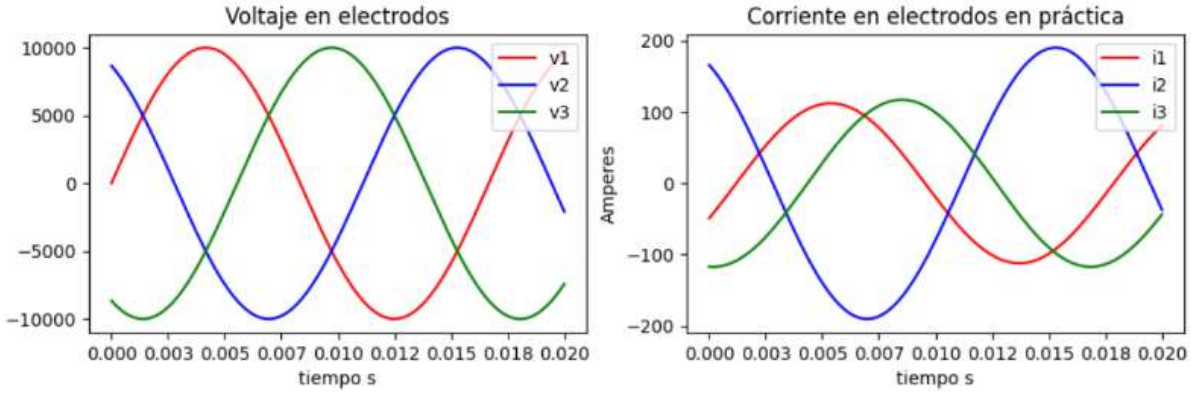


Figura 11. Voltaje y corriente de inmersión.

Con el modelo simbólico se puede proponer una función de transferencia para las corrientes i_1, i_2 e i_3 para el sistema que depende exclusivamente de la inmersión de los electrodos, proponer una ley de control que controle las corrientes a partir de las inmersiones recordando que en la práctica se debe contemplar el desgaste de los electrodos para tenerlo en cuenta en los cálculos de inmersión, la figura 9 muestra un desbalance en la corriente consumida en cada electrodo tomando en cuenda una inmersión desproporcionada. La simulación completa se puede encontrar en el repositorio: https://github.com/luisCorl/eaf-simbolico/blob/main/EAF_simbolico.ipynb en donde lo ideal es tener un balance de corriente en las fases o líneas del transformador, así como tener un control de un máximo consumo de corriente.

Conclusiones

Los resultados que se obtienen con el simulador simbólico ESCAPy son interesantes y confiables ya que permiten conocer el comportamiento del modelo desde el punto de vista simbólico para el caso del modelo EAF los resultados indican el comportamiento de la corriente y como puede tener un desfasamiento de onda dependiente de las resistencias vecinas y como estas pueden variar según su inmersión, además se demuestra la funcionalidad de la herramienta ESCAPy como alternativa al refuerzo de análisis en circuitos eléctricos para alumnos.

Una de las principales contribuciones es presentar un software de acceso libre a la comunidad académica estudiantil que cursa materias de análisis de circuitos, para alumnos ajenos a carreras de electrónica y mecatrónica sin un amplio conocimiento en metodologías de análisis en circuitos eléctricos y desarrollan proyectos con algún circuito eléctrico que requieren del análisis simbólico ESCAPy es una alternativa, sin embargo esta herramienta también contribuye al análisis de circuitos complejos y que comúnmente se analizan en simuladores numéricos pero que se conoce poco sobre las ecuaciones que describen el circuito eléctrico, un ejemplo el modelo del horno de fundición por arco eléctrico (EAF), filtros WTA (Winner Take All) LTA (Loser Take All) son circuitos no comunes en al análisis de circuitos eléctricos que requieren en ocasiones de cierta experiencia para planteralos y analizarlos

Limitaciones.

El ejemplo puede replicarse en Google Colab, o Python IDLE o Spyder, sin embargo, algo que podría implicar en no obtener los resultados es la instalación previa de las librerías sympy, memory profiler, symengine, pandas, numpy, multiprocesing, ddd_layer y por supuesto ESCAPy.

ESCAPy trabaja con la susceptancia en inductores y capacitores, por lo que es importante recordar que el análisis que devuelve es en el dominio de la frecuencia, para hacer el análisis en el tiempo es recomendable usar el análisis con números complejos. Es importante monitorear la memoria ram, si se observa que la memoria ram está por arriba del 98% se aconseja abortar inmediatamente la simulación, cada computadora podrá operar diferentes tamaños de circuitos dependiendo del hardware.

Recomendaciones.

Los investigadores interesados en continuar nuestra investigación podrían concentrarse en el diseño de algoritmos y su influencia en la contribución académica. Podríamos sugerir que hay un abundante campo todavía por explorarse en lo que se refiere a procesamiento en paralelo y mejoras en los algoritmos de ESCAPy.

Referencias

Bekker, Johannes Gerhardt, Ian Keith Craig, and Petrus Christiaan Pistorius. "Modeling and simulation of an electric arc furnace process." *ISIJ international* 39.1 (1999): 23-32.

Boulet, Benoit, Gino Lalli, and Mark Ajersch. "Modeling and control of an electric arc furnace." *Proceedings of the 2003 American Control Conference, 2003.*. Vol. 4. IEEE, 2003.

Fathi, Amirhossein, et al. "Comprehensive Electric Arc Furnace Model for Simulation Purposes and Model-Based Control." *steel research international* 88.3 (2017): 1600083.

Fakhfakh, Mourad, Esteban Tlelo-Cuautle, and Francisco V. Fernández, eds. *Design of analog circuits through symbolic analysis*. Bentham Science Publishers, 2012.

Fernandez, Francisco V. "Symbolic analysis techniques: applications to analog design automation." *(No Title)* (1998).

Logar, Vito, Dejan Dovžan, and Igor Škrjanc. "Mathematical modeling and experimental validation of an electric arc furnace." *ISIJ international* 51.3 (2011): 382-391.

Odenthal, Hans-Jürgen, et al. "Review on modeling and simulation of the electric arc furnace (EAF)." *steel research international* 89.1 (2018): 1700098.

SymPy. "sympy/core/symbol.py - Implementación del módulo Symbol en el repositorio SymPy" [Código fuente]. GitHub, s.f. Recuperado de \url{https://github.com/sympy/sympy/blob/master/sympy/core/symbol.py}. Fecha de acceso: 3 de abril de 2024.

Notas Biográficas

El **Ing. Luis Cortés Ramírez** es estudiante de posgrado de maestría en ciencias de la electrónica opción Automatización por la Benemérita Universidad Autónoma de Puebla, con experiencia en la automatización de la industria metalúrgica desde 2018.

La **Dr. Luis Abraham Sánchez Gaspariano** es profesor investigador en áreas diseño de circuitos integrados, telecomunicaciones, procesamiento de señales, sistemas automotrices y energías por la Benemérita Universidad Autónoma de Puebla perteneciente al SNI 1.

El **Dr. Carlos Leopoldo Pando Lambruschini** es profesor investigador en áreas de dinámica no lineal, óptica no lineal, y sistemas hamiltonianos por la Benemérita Universidad Autónoma de Puebla perteneciente al SNI 2.

ACADEMIA JOURNALS

OPUS PRO SCIENTIA ET STUDIUM

Congreso de Investigación Academia Journals Abril 2024:
Avances Multidisciplinarios para una Sociedad Innovadora

RECONOCIMIENTO

A:

Ing. Luis Cortés Ramírez

Por su destacada participación como Jefe de Sesión en la Presentación de Ponencias Virtuales en Vivo a través de Zoom, el día 23 de abril de 2024, en el contexto del Congreso Academia Journals Abril 2024: *Avances Multidisciplinarios para una Sociedad Innovadora*

Dr. Rafael Moras
Director General,
Academia Journals

Mtra. Analucia Alegre
Directora de Desarrollo,
Academia Journals

Congreso Academia Journals Abril 2024
Lunes y Martes, 22 y 23 de Abril
Horario de Ponencias Online, Presentadas en Vivo, Vía Zoom

Autor		Ponencia	Título de la Ponencia	Área Temática	Institución
Urlina Adriana Hernández Oliveros	Lunes 16:00 - 17:00 Hora CDMX	FRS020	Percepción del Acoso Sexual Callejero en Jóvenes: Una Comparación entre Colombia y México	Humanidades y ciencias sociales, bellas artes	Centro de Estudios Universitarios Vizcaya de las Américas, Uruapan
Karen Anif Jiménez Cuevas		FRS080	Principales Aspectos del Modelado y Simulación de un Biorreactor	Ingenierías	TecNM, Instituto Tecnológico de Misantla
Gabriel Adrián Romero		FRS005	Procedimiento para la Exploración de la Zona de Menor Aprovechamiento Energético de un Aspa Eólica Aplicado a la Turbina Mexicana MEM-B30	Ingenierías	Centro de Ingeniería y Desarrollo Industrial, Querétaro
Miguel Villagómez Galindo		FRS075	Análisis Bibliométrico sobre la Responsabilidad Social Empresarial en el Periodo 2013-2023	Ciencias administrativas	Universidad del Centro del Bajío Universidad de Guanajuato Universidad Michoacana de San Nicolás de Hidalgo
Isma Sandoval Galaviz		FRS069	Egreso en la Licenciatura en Derecho	Educación	Universidad Autónoma de Nayarit (UAN) Universidad Michoacana de San Nicolás de Hidalgo
Leidy Erandy Hernández Magaña	Martes 9:00 - 10:00 Hora CDMX	FRS044	Caracterización molecular del papel de E4ORF1 de Adenovirus 36 en el proceso de Autofagia	Ciencias	Universidad Juárez Autónoma de Tabasco (UJAT) Instituto Nacional de Perinatología
Helida Nancy Chávez Pérez		FRS027	Validez y Confiabilidad de la Escala Miedo a la Muerte en Población Adolescente Mexicana	Ciencias de la salud	Fac. C. Conducta, Universidad Autónoma del Estado de México (UAEM)
Luis Cortés Ramírez		FRS055	Análisis de Horno EAF Usando ESCAPy una Herramienta EDA para el Análisis Simbólico	Ingenierías	Benemérita Universidad Autónoma de Puebla (BUAP)
José Alfredo Díaz López		FRS028	Estabilidad Asintótica Global del PID Lineal Usando una Función Estricta de Lyapunov	Ingenierías	Lab. Robótica y Control, Fac. C. de la Electrónica, Benemérita Universidad Autónoma de Puebla (BUAP)

BENEMÉRITA UNIVERSIDAD
AUTÓNOMA DE PUEBLA