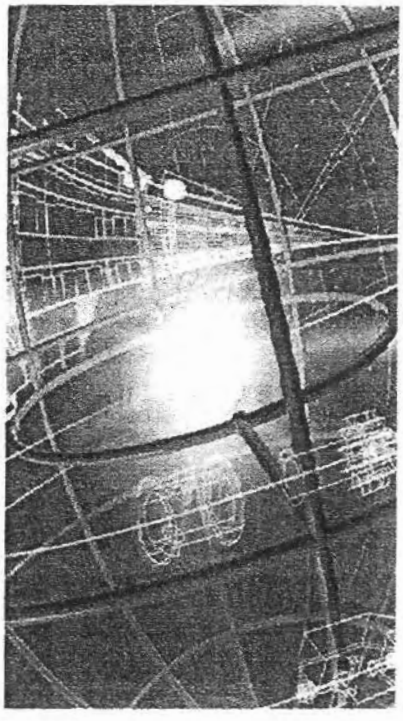
UNIVERSIDAD TECNOLÓGICA DE LA MIXTECA
INGENIERÍA EN ELECTRÓNICA

SISTEMA CAD PARA ANÁLISIS BÁSICO DE CIRCUITOS ELÉCTRICOS



TESIS PROFESIONAL QUE PARA OBTENER EL TITULO DE
INGENIERO EN ELECTRÓNICA

PRESENTA:

LANCELOT GARCÍA LEYVA

HUAJUAPAN DE LEÓN, OAX. JULIO DEL 2001

D EDICATORIAS

Primeramente a DIOS ya que sin su ayuda
nada de esto, hubiera sido posible.

A mis padres, por haber tenido el tiempo y
paciencia suficientes durante mi formación
de hombre y ahora de profesionista.

A mis hermanos Gabriel, Judith y Eliel por ser
más que unos simples hermanos durante los sucesos buenos
Y malos de mi vida.

A mi cuñada Tere y a sus hermosos hijos Gabito,
Kevin y Kevy, por alegrar mis momentos de tristeza
Y dolor.

A Susy por tener la paciencia ya que
a pesar de no esperar nada siempre me apoyo
en mis decisiones.

De forma muy especial al Ing. José Antonio M, por ser quien me apoyo
durante mi formación académica y asesoró en este trabajo.
Gracias Ingeniero..

Al Dr. José Javier B. Por el impulso que siempre representa en mí,
pra seguir adelante y ser la inspiración en el desarrollo de este trabajo.

A mi Universidad, por todo el apoyo recibido durante
m estancia de cinco años.

A las personas que laboran de forma ardua en la
Universidad por que a través de su trabajo coadyuvan a hacernos
profesionistas. Por mencionar algunas: Ing.Gerardo García
Lic. Carlos Santibañez.....

Índice General

1

Introducción

Con la aparición de los microprocesadores y la integración a gran escala (VLSI), que son circuitos de alta densidad de elementos, los cuales en estos días superan el millón de transistores cuyo análisis y diseño de estos llega ser tan complicado que sin la ayuda de un ordenador y un software de apoyo seria casi imposible de realizarlo.

El presente trabajo consiste en desarrollar un sistema de análisis para la verificación de circuitos eléctricos, básicamente consiste en tener un software de análisis básico y que sea propiedad de la Universidad Tecnológica de la Mixteca y sobre todo que sea una herramienta útil a los alumnos que cursan las materias de circuitos eléctricos.

El sistema es desarrollado bajo el lenguaje de programación Maple utilizando técnicas en el Diseño Asistido por Computadora (CAD) de la forma más adecuada y eficiente posible (Cáp. 2 y 4).

En el marco teórico, se dan las bases necesarias de cómo formar este sistema básico de análisis, además se da un repaso rápido sobre las herramientas de verificación de circuitos y su clasificación.

El desarrollo del sistema es una parte importante del presente trabajo, en el cual se describe paso a paso, desde que partes constituyen el programa principal y cuales son las funciones vitales para el funcionamiento correcto de éste, las cuales fueron llevadas a la práctica (teoría cáp. 3, práctico cáp. 4).

El diseño asistido por computadora es un área que se ha desarrollado rápidamente en los últimos años. Uno de sus pasos fundamentales es la formulación de las ecuaciones del circuito.

- *Primero:* La formulación de las ecuaciones de análisis esté basada en una mínima cantidad de información, relacionada a la configuración y a los valores de los elementos.

- *Segundo:* Las ecuaciones deben ser formuladas de tal forma que permitan su solución a través de una computadora digital.

La aplicación de la teoría de grafos, permite a la computadora formular las ecuaciones para el análisis, basado en una mínima cantidad de información proveniente de la red del circuito. La computadora, puede ser entonces programada, para utilizar la información relacionada a la topología del circuito, así como los nodos a los cuales esta conectado cada elemento. Las matrices topológicas son entonces generadas por la computadora, permitiendo la formulación eficiente de las ecuaciones de interés.

1.1. Antecedentes de la Electrónica

El termino *electrónica* se deriva de la palabra electrón, que se emplea para denotar una cantidad muy pequeña e invisible de electricidad que esta presente en los materiales. La electrónica puede definirse, en cuanto a sus muchas aplicaciones, de manera que incluya todos aquellos campos en los que es necesario el control de la electricidad, el ejemplo más claro, es en materiales semiconductores sólidos que se emplean en la fabricación de circuitos integrados.

La electrónica tuvo sus comienzos a la par de las comunicaciones por radio. Los primeros experimentos sobre electricidad y magnetismo sirvieron como base para el desarrollo de la radio. El comienzo de la transmisión inalámbrica, para fines de comunicación por radio, puede situarse en el trabajo del físico alemán Heinrich Hertz, que fue el primero en demostrar en 1887, los efectos de la radiación electromagnética en el espacio. En esa ocasión, la distancia de la transmisión fue de algunos pies; sin embargo, el experimento demostró que las ondas de radio pueden viajar de un lugar a otro sin necesidad de conectar por medio de alambres, los equipos de transmisión y recepción.

Hertz también demostró que las ondas de radio, a pesar de que no son visibles, viajan con la misma velocidad con que lo hacen las ondas de luz, he hecho tantos las ondas de radio como las de luz son dos ejemplos de radiación electromagnética. En esta forma de energía se combinan los efectos de la electricidad y el magnetismo. Una onda electromagnética transmite energía eléctrica a través del espacio.

El trabajo de Hertz fue precedido por varios experimentos en electricidad y magnetismo. En 1820, el físico danés H.O. Oersted demostró que una corriente eléctrica provoca efectos magnéticos. Después, en 1831, el físico Inglés Michael Faraday descubrió que un imán en movimiento genera electricidad y que, además, el movimiento impone el requisito de un cambio en el campo magnético. En 1864, el físico inglés James Clerk Maxwell, basándose en los trabajos previos en electricidad y magnetismo, predijo la existencia de las ondas electromagnéticas, hecho que fue confirmado experimentalmente por Hertz.

Puede juzgarse la importancia de estos trabajos por el hecho de que las unidades básicas de medición empleadas en electricidad y magnetismo llevan el nombre de muchos de estos científicos. El Maxwell (Mx) y el oersted (Oe) son unidades de medición para el magnetismo. El Hertz (Hz) es igual a un ciclo por segundo y la unidad con la que se mide la frecuencia de cualquier corriente o voltaje alterno. El farad (F) es la unidad con la que se mide la capacitancia e indica la cantidad de carga eléctrica que puede almacenarse en un capacitor.

En 1985, Guglielmo Marconi utilizó un alambre como antena con el propósito de desarrollar un sistema de radio para grandes distancias. La antena es necesaria, ya que con ella se logra una radiación más eficiente. Marconi tuvo éxito al lograr la primera comunicación inalámbrica a través del Océano Atlántico en 1901.

Después de este hecho, se sucedieron grandes avances, primero, gracias a la introducción y después a su mejora, de la válvula electrónica como amplificador de señales eléctricas. Lee DeForest con la invención en 1906 de una válvula electrónica de audio, fue uno de los líderes en este campo. Conforme avanzó el diseño de válvulas electrónicas, la transmisión por radio destinada al publico en general progresó en forma acelerada. En 1920, la estación KDKA transmitía, en forma regular, programas

de radio en la banda de amplitud modulada (AM) estándar,. La transmisión comercial por audio en frecuencia modulada (FM) comenzó en 1939, mientras que la transmisión estéreo en FM se inició en 1961.

La televisión comercial comenzó, de manera oficial, en 1941, pero se popularizó hasta 1945. El sistema de televisión en color que se emplea en el presente se adoptó en 1953.

El transistor se inventó en 1948 en los laboratorios Bell y, desde entonces, los dispositivos de estado sólido han remplazado a las válvulas electrónicas en la mayor parte de las aplicaciones de la electrónica, la radio y la televisión. El transistor es una aplicación del flujo controlado de electrones en materiales semiconductores, como el silicio (Si) y el Germanio (Ge). Los transistores y las válvulas electrónicas tienen usos similares para el control del flujo de electrones y la amplificación de señales. Sin embargo, el transistor es mucho más pequeño y eficiente, ya que, a diferencia de las válvulas electrónica, no necesita un filamento caliente.

Los dispositivos electrónicos de estado sólido incluyen transistores, diodos y circuitos integrados. Los dispositivos de estado sólido han permitido llevar a la práctica muchas aplicaciones novedosas, debido a su pequeño tamaño y al bajo precio de los encapsulados de circuitos integrados. Un ejemplo de lo anterior lo constituye el acelerado crecimiento de la electrónica digital y de sus aplicaciones, entre las que se incluyen las calculadoras electrónicas, las computadoras personales y otros dispositivos más.

2

Estructura de diseño de una herramienta CAD para verificar circuitos eléctricos.

Para el desarrollo y elaboración de este trabajo se necesitan de conocimientos básicos de ingeniería de Software y simuladores. En las siguientes secciones se da una breve historia de los sistemas CAD (Diseño Asistido por Computadora), así como una breve explicación de los requerimientos necesarios para construir un sistema CAD y una rama que es la verificación del diseño de circuitos.

2.1 Herramientas CAD

El diseño asistido por computadora (CAD) es una metodología en la cual el proceso de diseño y análisis tiene como soporte una computadora. En donde se tienen componentes de software y hardware los cuales deben de interactuar entre si para desarrollar sus objetivos de diseño y análisis [8].

El diseño asistido por computadora tuvo en sus inicios (en los años 60), algunos problemas debido a:

- La no aceptación inicial de CAD.
- La limitación tecnológica (la más importante en esos días).

Con el paso del tiempo estas limitantes han sido resueltas, y en nuestros días CAD ha llegado a ser una herramienta poderosa en el diseño, que en la mayoría de las áreas del conocimiento se encuentra presente.

Las ventajas que ofrecen los sistemas CAD son fáciles de visualizar así como notorias ya que incrementan la productividad debido a la reducción de los tiempos en la fase de desarrollo de un producto.

El sistema CAD (*CAD framework*), es conocido como la integración de varias herramientas CAD de verificación de circuitos, Un sistema CAD debe satisfacer ciertos requerimientos para su desarrollo [10]:

- El sistema CAD debe ser abierto: esto es que el sistema no sólo debe ser restringido a realizar la aplicación con la que fue diseñado. El sistema debe tener la capacidad de aceptar que otras nuevas herramientas u otros sistemas sean incorporados con mínimas modificaciones así como con un mínimo esfuerzo.

- Debe ofrecer alto grado de confiabilidad de ambiente.

- Debe de tener simplicidad de conceptos así como de implementación, ya que este punto tiene como finalidad la flexibilidad de una forma tal que sea lo más fácil posible de entenderlo como de implementarlo.

2.2 Fases de un sistema CAD

Las fases del sistema se refieren a las actividades de análisis, diseño, implementación y mantenimiento.

- ***Análisis:*** es el proceso donde se estudia como resolver un determinado problema para satisfacer las necesidades de un usuario.

- ***Diseño:*** es el proceso de resolver el problema, que puede ser visto como una cadena de transformaciones, en la cual los productos de varias actividades en el proceso se sobreponen en etapas sucesivas. Las actividades incluyen la definición de la arquitectura del sistema, componentes, datos e interfaces para que el sistema satisfaga los requerimientos especificados.

- ***Implementación:*** es el proceso donde se realiza un diseño en términos más concretos y particulares así como en función de hardware, software o ambos.

- ***Mantenimiento:*** Es el proceso que se refiere a la actividad de modificar un sistema o componente para la eliminación de sus errores, así como de desarrollar nuevas características o atributos que modifiquen su ambiente[10N].

2.3 Estructura de un sistema CAD.

Un sistema CAD se encuentra estructurado por: el módulo de comunicaciones, el módulo de métodos, el módulo de base de datos y el módulo de control. Como se muestra en la figura 2.1.

- *Módulo de comunicaciones*: es la interface entre el diseñador y el sistema CAD, el cual contiene comandos, controles y funciones gráficas de entrada - salida.

- *Módulo de métodos:* es el que se encarga de proveer los mecanismos de procedimientos adecuados para resolver un problema especifico.

- *Módulo de base de datos:* es el encargado de manejar los datos que serán utilizados por el sistema CAD.

- *Módulo de control:* Su función es la de coordinar todas las etapas para un funcionamiento correcto del sistema.

Los tres primeros módulos mencionados pueden desarrollarse en paralelo, ya que al poner el cuidado suficiente en los datos de entrada y salida de cada módulo para que al momento de ser interfazados no presenten incompatibilidad de datos.

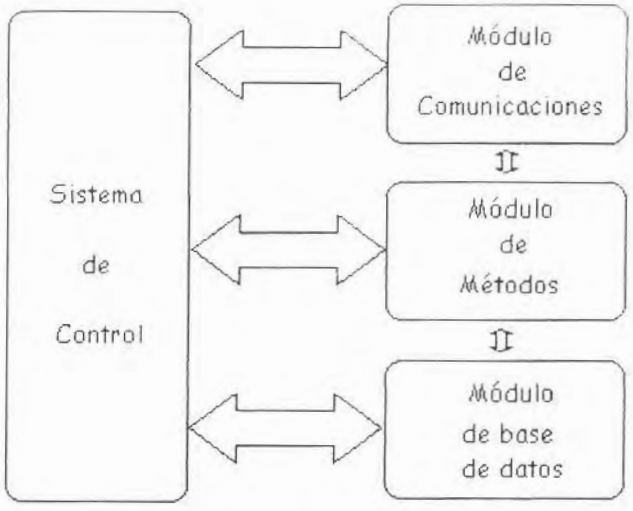


Figura 2.1 Estructura del sistema CAD desarrollado.

2.4 Verificación de circuitos

Una de las etapas cruciales para el diseño de circuitos integrados es la verificación. Es necesario verificar que el diseño de un circuito funcione de manera correcta, debido a que existen muchos factores que intervienen en su fabricación, como intervenciones humanas y modelos que no describen el comportamiento completo de los componentes en la síntesis del circuito.

La verificación de un circuito se puede dividir en dos partes:

- Simulación: La verificación se realiza antes de la construcción del circuito.
- *Pruebas:* Se realiza durante y después de la construcción.

2.4.1 Tipos de simuladores.

El avance en el nivel de integración en la tecnología de los circuitos integrados ha ido necesitando más y más de la ayuda de la simulación para la verificación del diseño de circuitos. Estos simuladores son empleados en todas las áreas del diseño de tal forma que se pueden clasificar en seis grupos [9]:

1. Simuladores funcionales.
2. Simuladores a nivel de registro de transferencia.
3. Simuladores lógicos.
4. Simuladores de circuitos.
5. Simuladores de dispositivos.
6. Simuladores de proceso.

Simuladores funcionales

Los simuladores funcionales han sido utilizados en dos áreas diferentes:

1. *Simuladores funcionales* (también conocidos y nombrados como algorítmicos o de comportamiento) los cuales son utilizados para simular algoritmos que van a ser implementados en hardware antes de ser fabricados. El punto principal de la simulación algorítmica es la capacidad de procesos recurrentes.

2. *Simuladores funcionales cuyas primitivas son macros,* se asemejan a los simuladores de registro de transferencia y sólo se restringen a la simulación de maquinas síncronas.

Simuladores de registro de transferencia

Los simuladores de registro de transferencia son la unión de dos actividades semejantes, la simulación a nivel arquitectura y a nivel de registro.

1. *Simulación a nivel registro.* Aquí el diseñador esta interesado en realizar funciones específicas. Estas operaciones son usualmente dadas como transferencia de información entre las diferentes etapas que se establecen en la arquitectura del diseño.

2. *Simulación a nivel arquitectura.* El diseñador está involucrado con toda la estructura del sistema, la cual se enfoca ha memorias, procesadores, etc.

Simuladores lógicos

Los simuladores lógicos son diseñados para verificar el funcionamiento de sistemas lógicos y detectar errores. Los simuladores de circuitos pueden ser utilizados para este propósito, pero el inconveniente que presentan al darles esta aplicación es que para circuitos digitales complejos tiene un alto costo en el tiempo de simulación.

En un simulador lógico los circuitos son modelados a nivel compuertas, como en estos simuladores solo es importante los niveles lógicos "0" y "1", los transistores que forman las compuertas son agrupados y substituidos siempre y cuando esto sea posible, así que el modelo a nivel compuertas se simplifica ha funciones booleanas. Por consecuencia de lo mencionado, los simuladores lógicos son más rápidos que otros simuladores, debido a que los simuladores no lógicos utilizan modelos más complicados que los lógicos para representar sus componentes circuitales.

Simulador lógico a nivel compuertas. Está constituido por una combinación de operaciones aritméticas, booleanas y unidades de retardo. La simulación se caracteriza por ser unidireccional para los modelos de compuertas lógicas con lo cual se implica que las entradas sólo afectan a las salidas y no viceversa; algunos de estos simuladores aceptan tres estados "1" (verdadero), "0" (falso) y "x" (indefinido), donde el estado "x" es usualmente para inicializar los tiempos de retardo, así como para direccionar ciclos en circuitos digitales secuenciales.

Simuladores de circuitos

La simulación de circuitos fue una de las primeras áreas en donde se desarrolló la simulación y se ha caracterizado por su constante refinamiento.

Los simuladores de circuitos se clasifican en: simuladores eléctricos (estos a su vez se clasifican en simuladores de propósito general y de dispositivos) y simuladores de retardo.

Simuladores eléctricos.

Los simuladores eléctricos son la unión de dos tareas de simulación, obtener las formas de onda que representa el comportamiento del circuito y aplicar modelos para la representación de dispositivos de tres o más terminales.

Simuladores de retardo.

En la simulación de retardo se estudia la propagación de señales, desde las entradas primarias hasta sus salidas, con esto para poder conocer las trayectorias de las señales. Cada bloque contribuye a un retardo de la señal que pasa a través de este.

Los simuladores de retardo al realizar el análisis de un circuito lo dividen en bloques funcionales en los cuales se estudia el tiempo de retardo y la conectividad del circuito. Los modelos que se emplean en este tipo de simulador son llamadas *ecuaciones de retardo*, las cuales son calculadas para cada bloque del circuito. La ecuación de retardo se obtiene al hacer una simulación eléctrica convencional a los bloques individuales que aparecen en la red.

Simuladores de circuitos generales.

Los simuladores más utilizados son los de este tipo, ya que se encuentran instalados en algunos cientos de computadoras alrededor del mundo, los cuales son usados cuando se realizan diseños con componentes y tecnologías comerciales, pero al realizar diseños específicos o con tecnologías especificas resultan ser inadecuados. Un ejemplo de este tipo de simuladores y familiar para la mayoría de diseñadores SPICE (*Simulation program for integraded circuits especially*) o para alta densidad HSPICE, CADENCE y el más utilizado en Europa APLAC.

Simuladores a nivel dispositivo.

Estos son los encargados de realizar simulaciones numéricas en dispositivos semiconductores o de otros tipos de dispositivos electrónicos usados en circuitos integrados, para poder conocer su comportamiento y así poder corregir errores de diseño.

De hecho las simulaciones están relacionadas con las soluciones numéricas de las ecuaciones diferenciales que se involucran con el comportamiento del dispositivo.

La importancia de la simulación de dispositivos es:

I. Proveer el comportamiento cualitativo del dispositivo para poder dar una solución al momento de analizar un efecto especifico como el fanout, el efecto capacitivo (capacitancias parásitas), etc.

II. Diseñar modelos de dispositivos para hacer simulaciones cuando éste sea utilizado como parte de un circuito.

III. Determinar y/o encontrar las limitaciones, así como la validación en las teorías de escalamiento.

Simuladores de procesos.

Éstos se involucran con la solución de ecuaciones diferenciales parciales que definen la fabricación de un dispositivo semiconductor (ATENA). Este tipo de simuladores su propósito es predecir los parámetros de fabricación tales como dopado, tiempos, temperaturas, etc., para la ayuda del control y calidad de los dispositivos a fabricar.

2.4.2 Pruebas de circuitos.

Esta etapa se hace presente cuando ya se tiene el circuito, entonces hay que hacer pruebas de éste con la finalidad de que se verifique el funcionamiento del mismo. Las pruebas de un circuito son hechas en todos los niveles de fabricación (a nivel prototipo y a nivel fabricación).

En el nivel prototipo se suprimen errores de los prototipos de circuito, en este punto se requiere de la localización de fallas para que esta se correlacione con la falla en el proceso o en el diseño. En este nivel no se toma en cuenta el costo del equipo que tiene que realizar las pruebas de detección de fallas.

El otro nivel de pruebas se realiza en la producción, donde se limita el costo del equipo y el tiempo para detectar fallas ya que aquí el tiempo de detección de fallas implica costos en el proceso de producción de un determinado producto.

Las pruebas se pueden clasificar en dos grupos: pruebas analógicas y pruebas digitales.

Prueba de fallas analógicas

La detección de errores es el primer paso del análisis de fallas, el cual consiste en usar un patrón de prueba para determinar si el sistema se comporta adecuadamente desde el punto de vista funcional.

El análisis analógico se encuentra atrasada con respecto al análisis digital, principalmente debido a su complejidad, dado que en el campo analógico se cuenta con un numero infinito de valores de voltaje o corriente, en comparación con el análisis digital que sólo importan dos estados: alto o bajo.

Los principales factores de dificultades en las pruebas analógicas pueden resumirse en:

- La complejidad de los circuitos analógicos de nuestros días ya que tienen una gran cantidad de parámetros que restringen su comportamiento, así como tener una cantidad de limitaciones de accesibilidad en sus componentes internos, restringe el uso de equipos de pruebas convencionales. Los equipos para realizar este tipo de pruebas resultan no tener la capacidad suficiente de almacenamiento, así como no tener la capacidad de almacenamiento ni la capacidad de computo requerida para realizar las pruebas.

- Los sistemas analógicos son frecuentemente no lineales, además de que tienen parámetros que varían grandemente, motivo por el cual los modelos determinísticos son ineficientes frecuentemente para modelar estos sistemas.

- Las relaciones que se tienen entre las señales de entrada y salida en circuitos analógicos son complejas comparadas con las digitales. Por lo tanto, modelar relaciones de entrada -salida analógicas es mucho más complejo que modelar digitales.

- La complejidad estadística de las fallas analógicas generalmente no se puede conocer de una manera completa. Por lo tanto, los métodos estadísticos o prababilisticos son ineficientes para realizar este tipo de pruebas.

2.5. Etapas de un simulador de circuitos eléctricos.

El análisis de circuitos eléctricos se pueden resolver en dos etapas las cuales están dadas por:

1. La primera consiste en la formulación de las ecuaciones del circuito a analizar utilizando las dos leyes de Kirchhoff así como las relaciones de los elementos que involucran sus propias variables eléctricas.

2. La segunda etapa consiste en resolver las ecuaciones de la etapa 1 mediante un método analítico o numérico.

En base a las dos etapas de un simulador de circuitos, se deben realizar las siguientes funciones:

1. Convertir el circuito dado por el usuario en una representación matemática.
2. Escoger y desarrollar los análisis funcionales.
3. Presentación de los resultados.

Los puntos antes mencionados conforman las etapas de desarrollo de un simulador de circuitos eléctricos, mostradas en la figura 2.5.1.

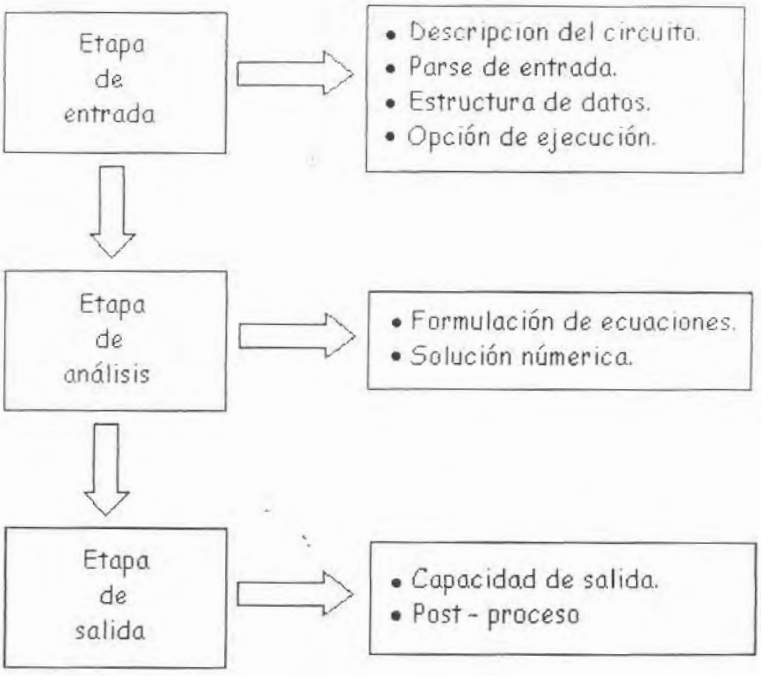


Figura 2.5.1 Etapas de un simulador.

La figura 2.5.1 nos muestra las etapas de entrada, análisis y la salida las cuales son las siguientes:

La etapa de entrada: su función es la de llevar a cabo la traducción del circuito a ser analizado en términos de la estructura de datos, requeridos para realizar el análisis en la segunda etapa. La etapa de entrada no sólo procesa la entrada de datos del diseñador o usuario, sino además debe proveer flexibilidad y algunas características orientadas al usuario para que el programa sea fácil de usar. En esta etapa deben hacerse presentes las técnicas de programación.

La etapa de análisis: esta es la parte más importante del simulador, debido a que en esta etapa los análisis requeridos por el diseñador son colocados en ecuaciones que modelan el comportamiento del circuito y se aplican métodos numéricos convenientes para resolver las ecuaciones.

En la etapa de análisis los simuladores pueden diferenciarse por realizar diferentes tipos de análisis, los cuales son:

1. Análisis en DC.
2. Análisis lineal en AC.
3. Análisis no lineal en AC.
4. Análisis lineal en el dominio del tiempo.
5. Análisis no lineal en el dominio del tiempo.

La etapa de salida: es la última, no por eso es menos importante en un simulador de circuitos, ella como su nombre indica es la que proporciona los datos de salida, así como una primera aproximación hacia cualquier otro postprocesamiento, puede ser hecho para que sea utilizado por cualquier otro procesamiento u otra herramienta en el ambiente CAD.

3

Bases Teóricas

Este capitulo cubre la parte teórica de las herramientas CAD que se desarrollaron. Sólo se hace mención de los puntos relevantes de cada una de las herramientas conforme al alcance de estas. En caso de que el lector quiera ampliar sobre un tema en particular se invita referirse a la bibliografía pertinente [4, 5, 6].

A continuación se describe a grandes rasgos el sistema de cálculo simbólico (SCS) Maple, utilizado para el desarrollo del sistema CAD.

MAPLE. Surgió como resultado de "Maple Project" a principio de los años 80's en la Universidad de Waterloo, Canadá. En 1992 la compañía Waterloo Maple Software comenzó a trabajar para comercializar el software. Es utilizado en la Universidad para la investigación y la enseñanza de alto nivel, en la empresa privada como aplicación de productividad, permite que sus numerosas funciones sean de ***manera simbólica y numérica*** a diferencia de Matlab, permite realizar diferentes tareas matemáticas como resolver ecuaciones así como sistemas lineales y no lineales, integrar, derivar, calcular límites, sumar series, resolver ecuaciones diferenciales, optimizar funciones, realizar cálculos estadísticos, manipular matrices, trabajar con polinomios, manejo de expresiones algebraicas, espacios vectoriales, aplicaciones lineales, formas cuadráticas, diagonalización de matrices, regresión simple y múltiple, programación lineal, teoría de grafos, lógica, variable compleja, etc. Maple ha sido diseñado de una manera inteligente, la cual contiene un pequeño *kernel* y una enorme librería. Maple

posee un lenguaje de programación amigable que ofrece al usuario un amplio rango de posibilidades para resolver problemas de ingeniería.

Las herramientas desarrolladas para el análisis de circuitos pueden ser tan variadas como el diseñador quiera que sean, con la ventaja que entre más características de análisis se tomen en cuenta en el sistema CAD, los resultados serán más aproximados a un circuito real, conllevando esto a que sea necesario más tiempo de diseño, así como mucho más tiempo de computo en el proceso de análisis y lo más complicado es que muchas de las características de análisis aun no son posibles de implementar.

La herramienta CAD desarrollada en este trabajo se enfoca al análisis en DC para circuitos lineales.

3.1 Conceptos fundamentales de teoría de circuitos.

Un circuito eléctrico es una entidad que está gobernada por las leyes físicas. Es bien conocido que el flujo de carga eléctrica en teoría de circuitos es un fenómeno discreto, sus unidades se encuentran dadas por unidades conocidas y determinadas por el número de electrones. Dadas las características de los componentes y de las interconexiones de un circuito físico, es posible considerar que el flujo de carga es un fenómeno continuo. Consecuentemente, la física cuántica no juega un papel importante en teoría de redes.

A continuación se mencionan las leyes e hipótesis de la teoría de circuitos.

3.1.1 Hipótesis de teoría de circuitos.

La teoría de circuitos se sigue con un conjunto de hipótesis para cumplir con sus metas, estas son [5, 6]:

1. Las variables eléctricas se definen sin ambigüedad.
   Para cualquier tiempo la dirección y la magnitud de voltaje o corriente en cualquier parte del circuito son únicas y permanecen sin cambios.

2. La teoría de redes solo analiza propiedades eléctricas de la red.
   La teoría de redes no analiza procesos mecánicos, térmicos o químicos del circuito físico.

3. El circuito es sólo accesible en algunos puntos.
   No es posible determinar por ejemplo la carga dentro de una resistencia, los valores de voltaje y corriente que arroja la teoría de circuitos, son en puntos discretos de ramas y nodos respectivamente. La teoría de circuitos utiliza un modelo distribuido de la carga dentro de los componentes.

4. La teoría de circuitos solo estudia circuitos *lumped*.
   Un circuito *lumped* es aquel cuyas dimensiones de sus conexiones y componentes son menores que la longitud de onda de la señal que pasa a través de ellos, de tal forma que la energía se pierda en forma de calor y no en forma de ondas electromagnéticas.

5. Las leyes fundamentales en teoría de circuitos son las leyes de Kirchhoff:
   - Ley de la corriente de Kirchhoff (KCL).
   - Ley de voltajes de Kirchhoff (KVL).

3.1.2 Leyes de Kirchhoff.

Las leyes de Kirchhoff son una simplificación de las leyes de Maxwell, ellas están definidas como [6]:

KCL: Para todo circuito *lumped* en cualquier tiempo t, la suma algebraica de todas las corrientes a través de una superficie gaussiana es igual a cero.

KVL: Para todo circuito *lumped* en cualquier tiempo t, para cualquier nodo de referencia y cualquier par de nodos (i,j) se cumple.

$$v_{k\_j}(t) = v_k(t) - v_j(t)^1$$

3.1.3 Definición moderna de circuito eléctrico.

Estudiosos de teoría de circuitos y de teoría de matroides dan una definición moderna de circuito eléctrico, la cual establece que: Un circuito eléctrico está constituido por dos conjuntos:

[1] En el presente trabajo se denotara v como voltajes de rama y v como voltajes nodales.

donde:

- *C* involucra las relaciones de rama y son funciones de los dispositivos que constituyen el circuito.

- *P* involucra el patrón de interconexiones, éste puede ser representado a través de la topología. El patrón de interconexiones es importante porque en él se lleva implícito las leyes de Kirchhoff independientemente de los componentes que la formen.

3.2. Modelos de circuitos y bloques de construcción.

Algunos programas de computadora de análisis de circuitos reconocen y permiten solo algunas series de elementos básicos de diseño de circuitos. Por ejemplo, si un programa fue diseñado para el análisis de redes solo contiene resistores y baterías. Esto implica que entre más grande sea la serie de elementos que acepte un programa de análisis más versátil puede llegar a ser, pero también entre más grande sea la serie de elementos más complicado es el desarrollo de este, además de que se requiere mucho más grandes equipos de computo tanto en la capacidad de almacenamiento como en la velocidad de computo.

En este punto cabe hacer la pregunta ¿cómo utilizar un programa de computo para el análisis de circuitos que contiene elementos no permitidos o que no pertenezca a la serie de elementos permitidos? Pues la respuesta más simple y si se puede, es remplazar los elementos no permitidos por un circuito equivalente hecho solo de elementos que sean permitidos, que pertenezcan a la serie de elementos que acepte el programa de análisis. Desafortunadamente esto no es posible a menudo, porque *por definición*, un circuito es equivalente a un elemento dado, si y solo si, ellos son indistinguibles cuando medimos en sus terminales externas y presentan el mismo comportamiento. Este fuerte requerimiento casi nunca esta satisfecho por dispositivos tales como diodos, transistores, etc. Sin embargo, en la mayoría de los casos prácticos, se ha encontrado la posibilidad de remplazar cada elemento no permitido por una aproximación de *circuito equivalente llamado circuito modelo*, y así poder tener una respuesta de computo, que represente una buena aproximación de aquellos que son realmente medidos. En otras palabras, la utilidad de un programa de computo es reforzado grandemente por el uso de modelos de circuitos reales.

Para sintetizar los modelos de circuitos reales para la mayoría de dispositivos y componentes de interés, la serie básica debe contener al menos cinco clases de elementos de circuitos lumped invariantes en el tiempo listados en la tabla 3.2.1.1. y descritos a continuación.

Un ***resistor*** es un elemento de dos terminales caracterizado por la curva en el plano de voltaje (v) contra corriente (i). Si la curva v-i consiste simplemente de una línea recta a través del origen, se dice que el resistor es lineal, en cualquier otro caso es no lineal. Si la curva v-i puede ser expresada como la corriente en función del voltaje entonces el voltaje controla la corriente. Un estricto crecimiento monotonico en la

curva *v-i* esto es que tanto la corriente como el voltaje son controlados [6] como puede verse en la figura 3.2.1.1.

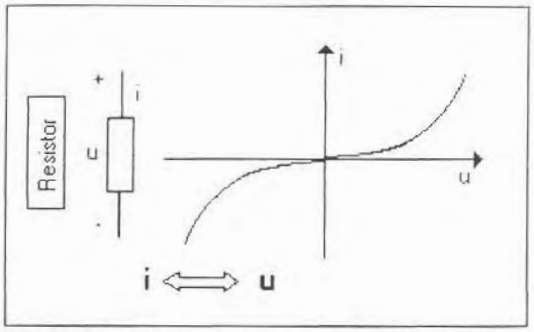


Figura 3.2.1.1. Relación de voltaje y corriente en un resitor.

Un ***capacitor*** **es** un elemento de dos terminales caracterizado por la curva en el plano carga (q) contra voltaje (v). Si la curva v-q consiste simplemente de una línea recta a través del origen, se dice que el capacitor es lineal y en cualquier otro caso se dice que es no lineal [6] como puede verse en la figura 3.2.1.2.

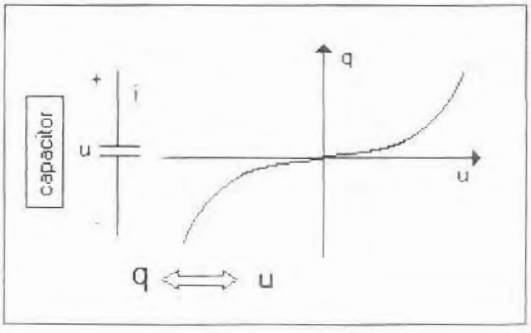


Figura 3.2.1.2. Relación de voltaje y carga en un capacitor

Un ***inductor*** es un elemento de dos terminales caracterizado por la curva en el plano de carga (q) contra corriente (i). Si la curva i-q consiste simplemente de una línea recta a través del origen, se dice que el inductor es lineal y en cualquier otro caso se dice que es no lineal [6] como puede verse en la figura 3.2.1.3.

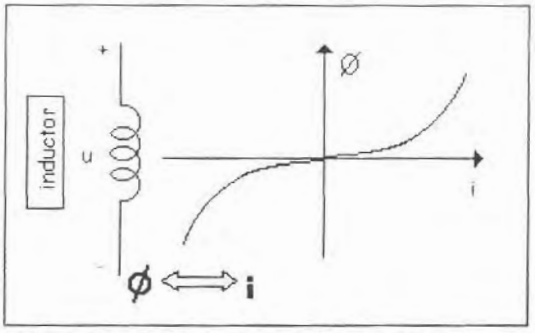


Figura 3.2.1.3. Relación de corriente y flujo de carga en un inductor.

Una ***fuente de voltaje independiente*** es un elemento de dos terminales cuya terminal de voltaje $v_s(t)$ en algún instante de tiempo se prescribe como a priori por consiguiente es independiente de la corriente en cualquier instante de tiempo [6] como puede verse en la figura 3.2.1.4.

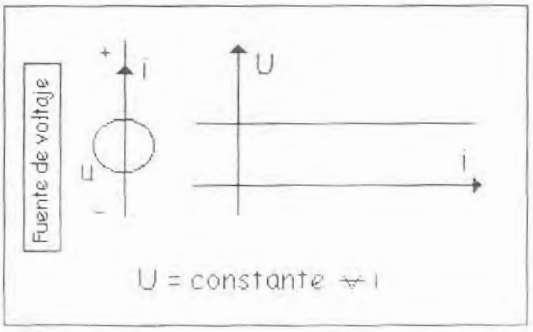


Figura 3.2.1.4. Fuente de Voltaje independiente.

Una ***fuente de corriente independiente*** es un elemento de dos terminales cuya terminal de corriente $i_s(t)$ en algún instante de tiempo se prescribe como a priori por consiguiente es independiente del voltaje en cualquier instante de tiempo [6] como puede verse en la figura 3.2.1.5.

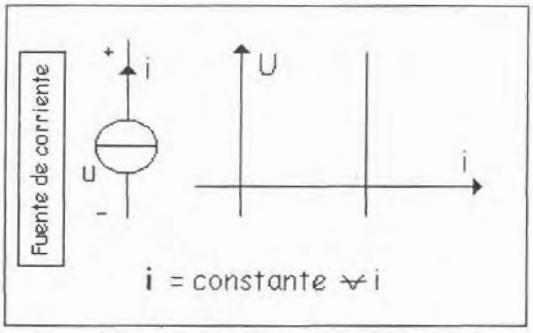


Figura 3.2.1.5. Fuente de corriente independiente.

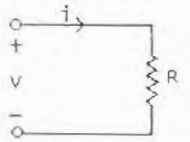
Una fuente lineal controlada (dependiente) es un elemento de dos terminales cuya terminal de voltaje o corriente en algún instante de tiempo es proporcional al voltaje v_x o la corriente i_x de alguna parte del circuito. Si el voltaje controlado v_x o la corriente controlada i_x pertenece al voltaje o corriente de un elemento x, entonces este elemento es llamado el elemento controlador.

La mayoría de los dispositivos y componentes *lumped* e invariantes en tiempo, pueden ser modelados usando solo cinco tipos de elementos de circuito listados en la tabla 3.2.1.1.

TABLA 3.2.1.1 Serie básica mínima de Elementos Lumped invariantes en el tiempo

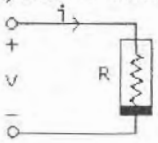
1. Resistor

a) Resistor lineal



$v = Ri$
$i = Gv$
R = Resistencia
G = Transresistencia

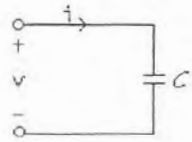
b) Resistor no lineal



Resistor controlada por corriente: $v = v(i)$.
Resistor controlada por voltaje: $i = i(v)$.
Un resistor estrictamente monotonica puede ser caracterizada por ambas.

2. Capacitor

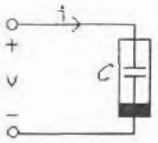
a) Capacitor lineal



$$i = C\frac{dv(t)}{dt}$$

$$v = \frac{1}{C}\int i(\tau)\,d\tau$$

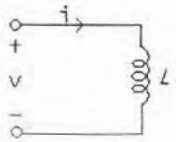
b) Capacitor no lineal



Capacitor controlada por carga: $v = v(q)$, $i(t) = dq/dt$.
Capacitor controlada por voltaje: $q = q(v)$, $i(t) = dq(t)/dt$ o $i(t) = C(v)\,dv(t)/dt$ donde $C(v) = dq(v)/dv$ es llamada la capacitancia incremental. Un capacitor estrictamente monotonica puede ser caracterizada por ambas.

3. Inductor

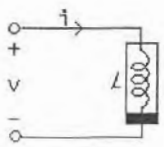
a) Inductor lineal



$$v = L\frac{di(t)}{dt}$$

$$i = \frac{1}{L}\int v(\tau)\,d\tau$$

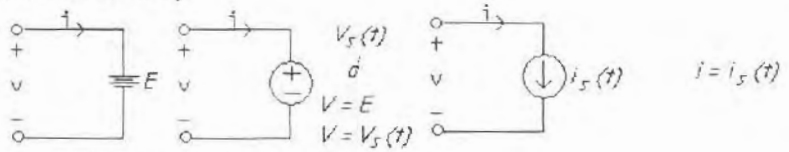
b) Inductor no lineal



Inductor controlada por flujo: $i = i(\Phi)$, $v(t) = d\Phi(t)/dt$.
Inductor controlada por corriente: $\Phi = \Phi(i)$ $v(t) = d\Phi(t)/dt$ o $v(t) = L(i)\,di(t)/dt$, donde $L(i) = d\Phi(i)/di$ es llamada la inductancia incremental. Un inductor estrictamente monotonica puede ser caracterizada por ambas.

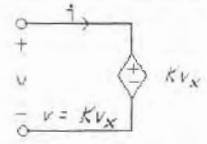
4. Fuentes independientes

a) Fuente de voltaje

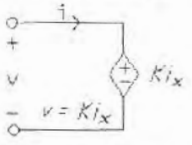


5. Fuentes lineales controladas (dependientes)

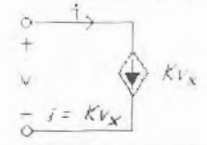
a) Fuente de voltaje controlada por voltaje



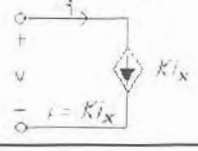
b) fuente de voltaje controlada por corriente



c) Fuente de corriente controlada por voltaje



d) Fuente de corriente controlada por corriente



3.3. Jerarquía y tipos de modelos de Circuito

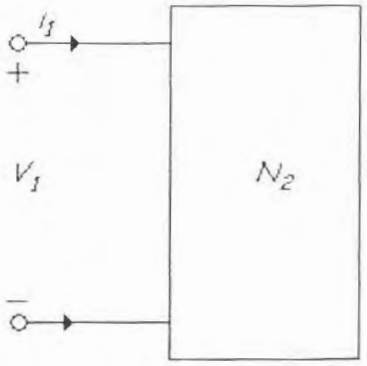
Un dispositivo de dos, tres y (n+1) terminales son mostrados en la figura 3.3.1.(a), (b) y (c), respectivamente. El principal objetivo es sintetizar una red con el mismo número de terminales externas solo usando los elementos básicos tal y como se muestra en la figura 3.3.1(d), (e) y (f) con exactitud aceptable de las características de los dispositivos correspondientes a cada uno de ellos. Donde cada cuadro de contorno negro nos representa el modelo del circuito de cada dispositivo físico.

El modelo de un elemento de circuito puede darnos características diferentes ya sea en la velocidad de subida o caída al que pudiera darnos si usamos otro modelo, entonces, es importante saber que modelo se adecua a la aplicación que se le pretende dar a este modelo de elemento de circuito.

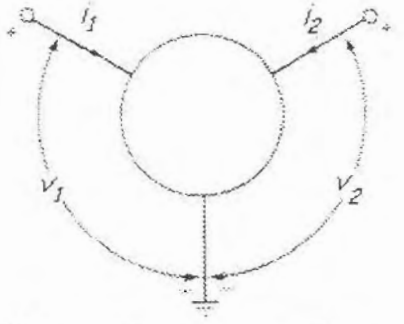
Un modelo puede ser mucho mejor que otro modelo en un circuito especifico. La habilidad de seleccionar el modelo más apropiado depende de un entender firme de la jerarquía y de los diferentes tipos de modelos de elementos de circuito.

Quizá la distinción más significante de la cantidad de varios tipos de modelos es basado en la clasificación de las señales, por lo tanto un modelo es diseñado con respecto a la propiedad de manejo de esas señales. Desde el punto de vista del modelado, las dos cualidades más importantes de una señal están dados por el rango de amplitud y el ancho de banda. El rango de amplitud correspondientes a los voltajes y corrientes máximos y mínimos instantáneos balanceados con los que el dispositivo se sujetará. El ancho de banda corresponde a la componente más baja y más alta de frecuencia de la señal.

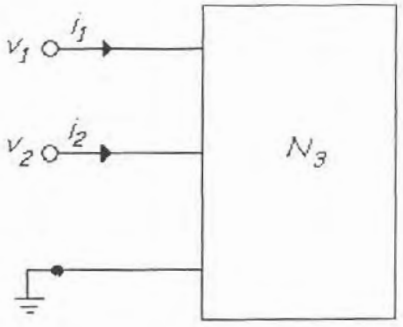
(a) Dispositivo de dos puertas



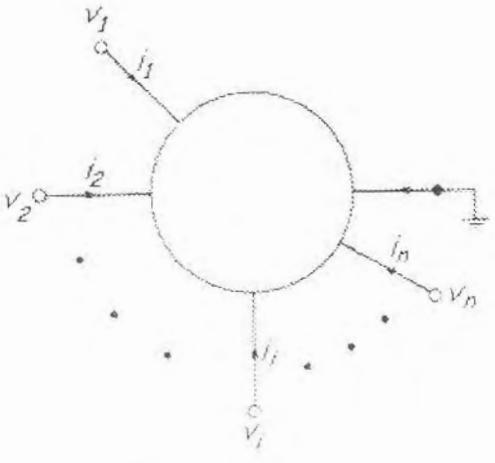
(d) Red de dos puertas



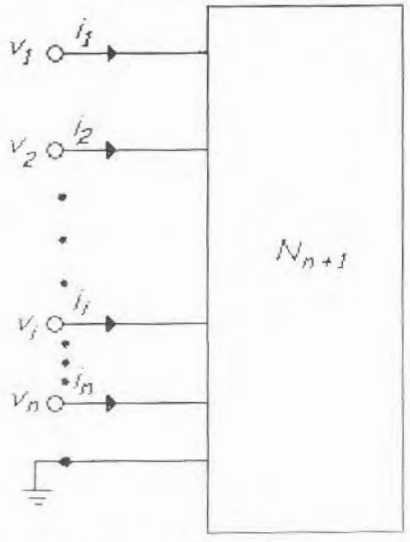
(b) Dispositivo de tres puertas



(e) Red de tres puertas



(c) Dispositivo de n puertas



(f) Red de n puertas

Figura 3.3.1. Símbolos para dispositivos y redes de dos, tres y (n + 1) terminales

3.3.1. Clasificación de modelos en términos del rango de amplitud de la señal.

Dependiendo del rango de la amplitud de operación de las señales, un modelo puede ser clasificado como modelo global, local o linealmente incremental.

- Un *modelo Global*: es diseñado para simular un dispositivo dado sobre todos los rangos medibles de voltajes y corrientes.

- Un *modelo Local*: se diseña para simular con precisión sobre algunas regiones de los rangos de operación de los dispositivos.

- Un *modelo incremental lineal* : es un modelo local hecho sobre algunos elementos lineales de la serie básica de elementos permitidos.

Un modelo global de un dispositivo físico es invariablemente no lineal; esto es que contiene un resistor, capacitor o inductor no lineal.

Un modelo local puede o no puede ser no lineal, dependiendo del rango de operación que se este tomando o en cual se este midiendo, ya que se puede medir en un rango lineal o en un rango no lineal.

Un modelo incremental lineal simula las características del dispositivo haciendo un barrido pequeño del punto de operación del dispositivo. Esto puede ser expresado geométricamente como una traslación del origen hacia el punto de operación del dispositivo tal que las características alrededor de este punto de operación se aproximen mucho a una linealidad.

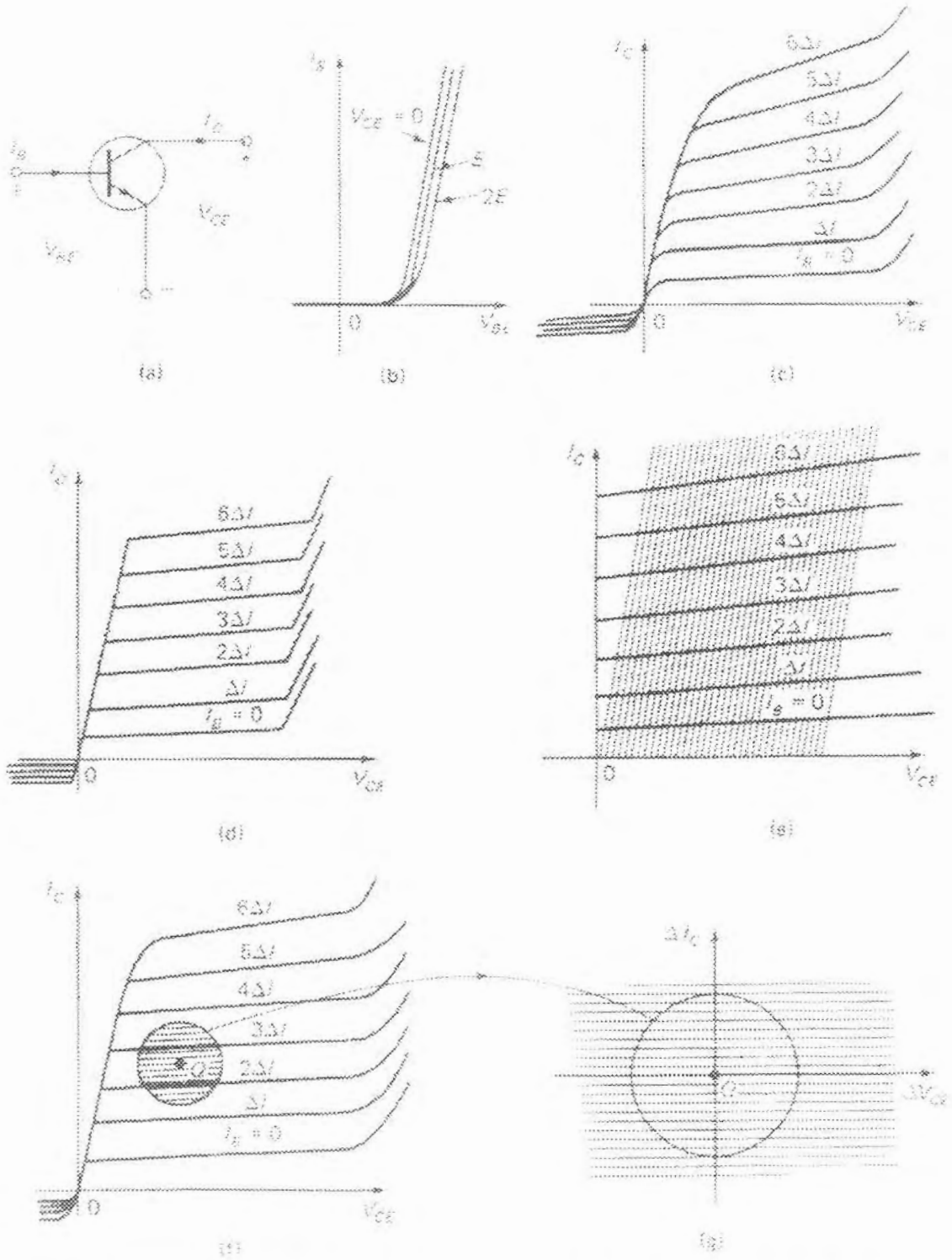


Figura 3.3.1.1. Dominio de la definición de varios tipos de modelos de circuitos, para un transistor bjt. (a) Símbolo transistor npn, (b) Curva característica, (c) curva característica diferente modelo que (b), (d) Modelo global, (e) modelo local, (f) modelo incremental lineal, (g) muestra de la linealidad de (f).

Una ilustración de los diferentes tipos de modelos para un transistor npn considerando las características de entrada - salida en la configuración de emisor común, se muestran en la figura 3.3.1.1. (a), (b) y (c).

Las curvas características de rendimiento por un modelo global típico figura 3.3.1.1. (d)., denota que aunque estas curvas no son una réplica exacta de aquellas mostradas en la fig. 3.3.1.1.(c), representan una aproximación justa encima de todas las regiones de las curvas mostradas en las dos anteriores figuras. Por otro lado, el rendimiento característico de la curva de un modelo local típico podría ser similar al mostrado en la fig. 3.3.1.1. (e)., puesto que la aproximación en este caso sólo es buena en la parte superior de la región de cruce principal (el cuadrante positivo del plano), y resaltando que el dispositivo sólo opera en esta región, donde las soluciones obtenidas por la computadora usando un modelo local, son tan buenas como aquellos obtenidas usando un modelo global que corresponden a las características mostradas en la figura 3.3.1.1. (d).

3.4. Método Nodal Modificado (MNA).

El Método Nodal Modificado (MNA) [1,4,5] es una generalización del Nodal aplicable al análisis de circuitos con todo tipo de elementos de dos terminales (R, L, C, E, J); de cuatro terminales, como lo son las fuentes controladas VCT, CCT, VVT, CVT, y el acoplamiento magnético; amplificadores operacionales ideales y con modelos que incorporan el producto finito ganancia - ancho de banda, además de bipuertos definidos por sus parámetros homogéneos, híbridos o de transmisión.

En el MNA se caracteriza por dividir los elementos en dos categorías; los MNA - compatibles (elementos en función de la corriente) y los MNA - no compatibles (elementos en función del voltaje), los cuales implican que las tensiones de los nodos sean variables del sistema de ecuaciones. En consecuencia, el orden de la matriz del sistema es mayor que la obtenida en el método Nodal.

Este método es muy conveniente en el análisis de redes orientado a la implementación en algoritmos computacionales, con el fin de facilitar la elaboración de programas para la determinación de funciones de red o la respuesta temporal del circuito, esto último por medio de técnicas de integración numérica, tales como *Forward Euler* o *Backward Euler*. El método MNA proporciona un sistema de ecuaciones de la forma:

$$A_{MNA}X = W$$

3.4.1. Backward Euler.

El método Backward Euler hacia atrás [1,5,6] se basa en la aproximación de diferencia hacia atrás y se escribe así:

$$y_{n+1} = y_n + hf(y_{n+1}, t_{n+1}) \qquad (3.4.1.1)$$

El orden de exactitud de este método es el mismo que el de Forward Euler, además, sí f es una función no lineal de y, es preciso emplear un método iterativo (sustituciones sucesivas) en cada paso. Por otro lado, las ventajas de este método son:

a) Es incondicionalmente estable.

b) Se garantiza la posibilidad de la solución cuando ésta debe ser positiva.

c) La formulación es absolutamente estable en todo el semiplano izquierdo.

Para el análisis de circuitos eléctricos, en el dominio del tiempo, las ecuaciones pueden ser formuladas elemento por elemento, usando el modelo *companion*.

Consideremos la expresión para Backward Euler para el enésimo paso:

$$x_{n+1} = x_n + hx'_{n+1} \qquad (3.4.1.2)$$

y un capacitor típico de red. El voltaje a través del capacitor y la corriente que fluye a través de él, están relacionadas por i = Cdv/dt. Si identificamos al voltaje v como x, y x' como dv/dt = i/C, e insertamos estos elementos en la expresión anterior obtenemos los siguiente:

$$v_{n+1} = v_n + i_{n+1} \cdot h/C \qquad (3.4.1.3)$$

Esto puede ser reescrito como sigue:

$$i_{n+1} = \frac{C}{h} v_{n+1} - \frac{C}{h} v_n \qquad (3.4.1.4)$$

El valor $C/h = G_{eq}$ representa una conductancia y la ecuación puede ser vista en términos del modelo companion mostrado en la Fig. I.2.1. Aquí v_{n+1} e i_{n+1} son las variables en el paso n+1, Ya que $(C/h)v_n$ representa una fuente de corriente en paralelo con G_{eq}.

Un desarrollo similar se hace para el inductor, para el cual v = L di/dt. Identificando la corriente i con x, di/dt= v/L con x' insertando adecuadamente en la expresión (3.4.1.1).

$$i_{n+1} = i_n + v_{n+1} \cdot h/L \qquad (3.4.1.5)$$

o

$$v_{n+1} = \frac{L}{h} i_{n+1} - \frac{L}{h} i_n \qquad (3.4.1.6)$$

lo cual puede ser interpretado como un elemento resistivo con una fuente de voltaje en serie como se muestra en la Fig. 3.4.1.1. En resumen los pasos que se siguen para el método Backward Euler:

1. La red se convierte en una red resistiva con un resistor L/h que reemplaza a cada inductor y con una conductancia C/h que reemplaza a cada capacitor.

2. Fuentes que dependen de la solución en el paso anterior son conectadas en paralelo con C/h y en serie con L/h.

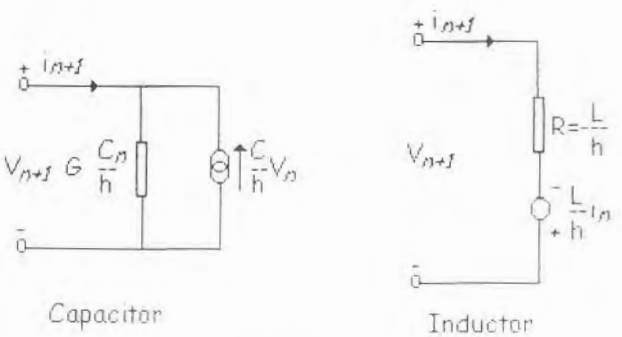


figura 3.4.1.1. Comparación de los modelos del capacitor e inductor por la formula backward Euler.

Aunque en el programa se sustituyo la inductancia como una conductancia de valor h/L en paralelo con una fuente de corriente i_n. Esto es para hacerlo un elemento MNA compatible, de acuerdo con la ecuación 3.4.1.5.

4. Desarrollo del Sistema CAD para análisis básico de circuitos eléctricos.

En este capítulo se presenta la metodología de como ha sido desarrollado El sistema CAD para el análisis básico de circuitos eléctricos, el sistema es implementado con la ayuda del lenguaje de programación Maple bajo el sistema operativo Windows, la herramienta que se desarrolló se encuentra en el archivo resolver.txt y ejecutado en ejemplos que se anexan a este trabajo, las cuales no necesitan ser instaladas solo ejecutados con el ruteo apropiado.

El capítulo comienza con la descripción rápida de las herramientas mas usadas en el análisis matemático por computadora. Continuando por que utilizar Maple como lenguaje de programación, posteriormente se proporciona información relacionada con Maple y para finalizar el capítulo se presenta una breve descripción de cada una de las funciones desarrolladas.

4.1. Programación simbólica.

Para diseñar o realizar pruebas a un circuito con la ayuda de herramientas CAD encontramos, que por un lado existe software desarrollado para el diseño y análisis de circuitos elementales, los cuales pueden ser de gran utilidad para una rápida y fácil interpretación. Estas herramientas por lo general son de bajo alcance, lo cual hace que el usuario requiera de herramientas adicionales no necesariamente de computo.

Por otro lado se tienen los ambientes CAD muy sofisticados. Estos sistemas no son capaces de soportar el diseño avanzado de circuitos, aún cuando estas herramientas resuelven estos problemas, la cantidad de información que se obtiene en los resultados es demasiada, motivo por el cual la interpretación y manipulación de estos es muy complicada de hacer.

Estas herramientas no son adecuadas para realizar un prediseño o en aplicaciones donde se requiere tener flexibilidad en la solución.

Para evitar la diferencia entre herramientas elementales y muy sofisticadas, existe una nueva generación de software para matemáticos e ingenieros, esta generación de software es conocida como Sistemas de Computación Simbólica (SCS), Los SCS son programas sofisticados que manejan expresiones matemáticas simbólicas y soluciones algebraicas como si se realizara en una hoja de papel. Los SCS son sistemas que pueden satisfacer el diseño del circuito desde la idea principal hasta la convergencia de la solución del circuito.

La utilización de SCS tiene un gran impacto en el manejo de parámetros del circuito teniendo más ventajas que el diseño de circuitos convencionales, simulación y otras herramientas que son basadas en el paradigma de la computación numérica.

4.2. MAPLE.

Maple inicialmente fue desarrollado por miembros del Symbolic Computation Group de la Universidad de Waterloo, en Ontario, Canadá; el cual ha sido comercializado por la compañía Waterloo Maple Software. Recientes investigaciones y agregados han sido realizados en colaboración con el Centro de experimentos y

construcciones matemáticas de Burnaby, Canada; así como con el Instituto Für Wissenchaftliches Rechnen, ETH - Zentrum de Zurcí, Suiza.

Maple es uno de los sistemas más utilizados por sus facilidades de manipulación simbólica, además de que puede integrar en un solo paquete las tareas de graficado, análisis numérico, presentación de resultados, aunado a esto también tiene la posibilidad de generar reportes en forma de un formato documental de salida de la o las corridas.

Maple incluye las definiciones de objetos básicos tales como: listas, conjuntos, expresiones y cadenas (*strings*), mientras que las funciones están organizadas en bibliotecas y paquetes. Además de lo anterior se tiene que presenta procesamiento numérico de precisión, grafica en dos dimensiones y facilidades de programación.

Maple fue seleccionado para implementar este sistema por lo antes mencionado y por que Maple posee un kernel pequeño y un gigantesco número de funciones.

El corazón de Maple lo constituyen sus rutinas simbólicas, que proporcionan una gran flexibilidad para su manejo. Maple provee una solución más exacta y con más precisión que los métodos de aproximación numérica. Sin embargo, cuando los métodos simbólicos no existen o son muy lentos en su ejecución, Maple resuelve este problema por aproximaciones numéricas, proporcionando un método alternativo en el análisis. Si se quiere un valor de punto flotante, este se puede calcular al final del calculo simbólico, de esta forma se eliminan los errores de redondeo que normalmente son acarreados a través de todos los cálculos en aproximaciones numéricas.

4.3. Estructura interna de Maple.

Internamente, Maple tiene una estructura jerárquica que lo conforma: ***el kernel, la librería y la interface***. Ver figura 4.3.1.

Figura 4.3.1. Estructura jerárquica de Maple.

El Kernel: es la "maquina matemática" para los cálculos de Maple. Es compacto, y está formado por un conjunto de rutinas altamente optimizadas, escritas y compiladas en lenguaje C y desarrollan la mayoría de los cálculos básicos del sistema.

La librería: La mayoría de los comandos de Maple residen en las librerías y son escritas en su propio lenguaje de programación. El código escrito en Maple no es compilado, es interpretado. Es decir, que el código se va ejecutando a medida que se va leyendo, esto permite crear un ambiente interactivo dentro de una sesión Maple.

La interface (IRIS): es la ventana de Maple que interactúa con el exterior, en ella se definen los procedimientos y da una forma interactiva de manejar los comandos. Dependiendo de la versión, la apariencia estará fluctuando entre una terminal de comandos en línea tipo *shell* o una interfase sofisticada que soporta documentos estructurados, combinando la entrada - salida de texto y gráficos. Ver figura 4.3.2.

La versión que se utilizo para la implementación del sistema CAD fue MAPLE V release 5, con una interfase gráfica corriendo en un sistema operativo Windows.

NOTA: Este software es compatible con el sistema operativo Linux.

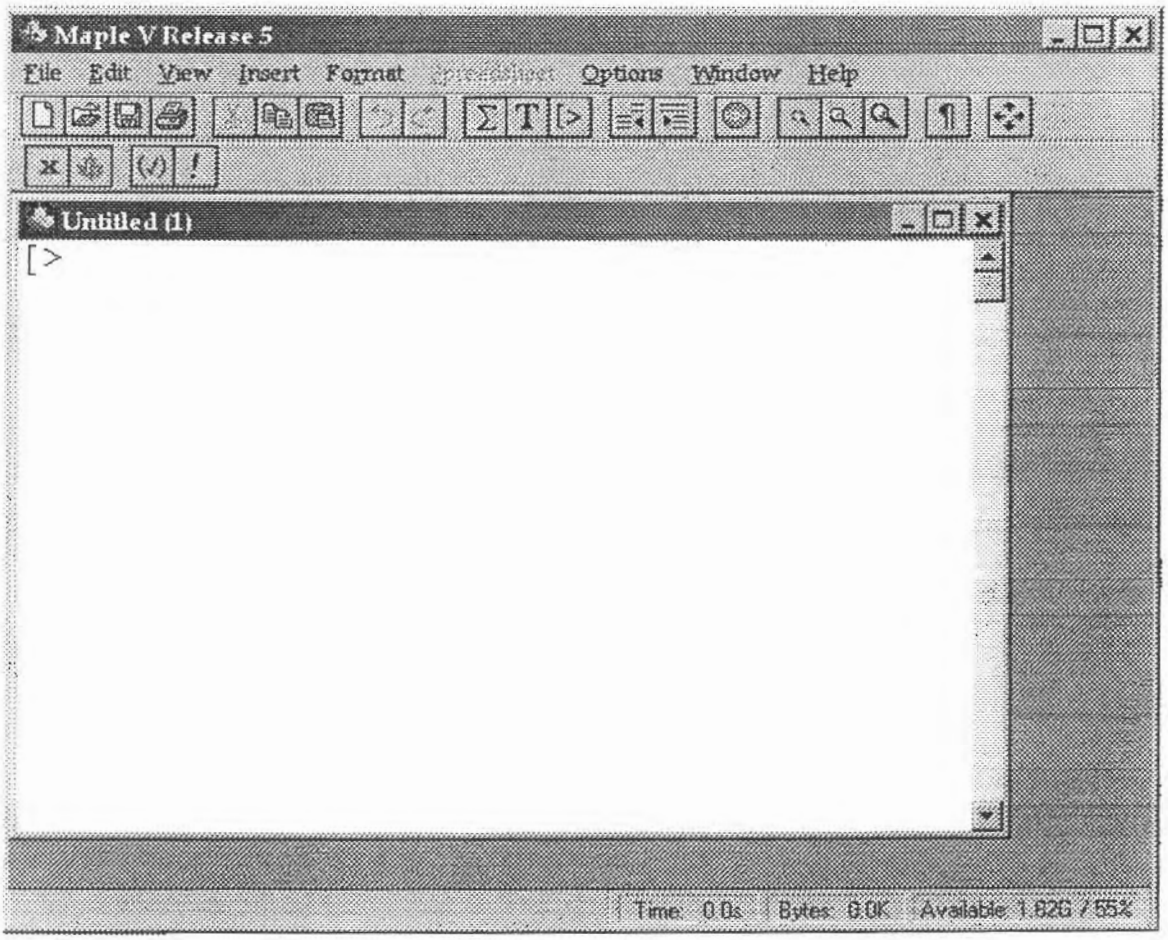


Figura 4.3.2. Interface de Maple V release 5.

4.4. Desarrollo teórico del programa.

El diseño asistido por computadora es un área que se ha desarrollado rápidamente en los últimos años. Uno de sus pasos fundamentales es la formulación de las ecuaciones del circuito, el cual se fundamenta en dos requisitos primordiales, el primero de ellos, es que la formulación de las ecuaciones de análisis esté basada en una mínima cantidad de información, relacionada a la configuración y a los valores de los elementos, el segundo es que las ecuaciones deben ser formuladas de tal forma que permitan su solución eficiente a través de una computadora digital.

4.4.1. Generación de las Matrices topológicas A, B, C y D por Computadora.

La generación de la matriz A (desde) A, es relativamente simple, se asignan enteros consecutivos llamados números de ramas de una grafo de red. Similarmente, los nodos de una grafo de red se le asignan números a los nodos, los cuales usualmente son enteros consecutivos que hincan de 1 o 0.

Para generar las matrices B y D, el programa debe seleccionar primero un árbol T. Frecuentemente T debe ser seleccionado por alguna preferencia, tal como el orden de los diferentes tipos de elementos de red que se incluyen en el árbol. Por ejemplo, fuentes de voltaje independientes, fuentes de voltaje controladas, capacitancias, resistencias e inductancias. En otras palabras el árbol T debe ser seleccionado incluyendo todas las fuentes de voltaje independientes como las primeras ramas del árbol, seguidas por las fuentes de voltaje controladas, capacitancias, resistencias e inductancias, en este orden hasta obtener un árbol T.

4.4.2. Como encontrar un árbol.

Considerando el método que hace uso de la matriz de incidencia reducida A.
Dado que las columnas de A son ordenadas de izquierda a derecha en el orden correspondiente a la preferencia de tipos de elementos que se desee. Ver figura 4.4.2.1.

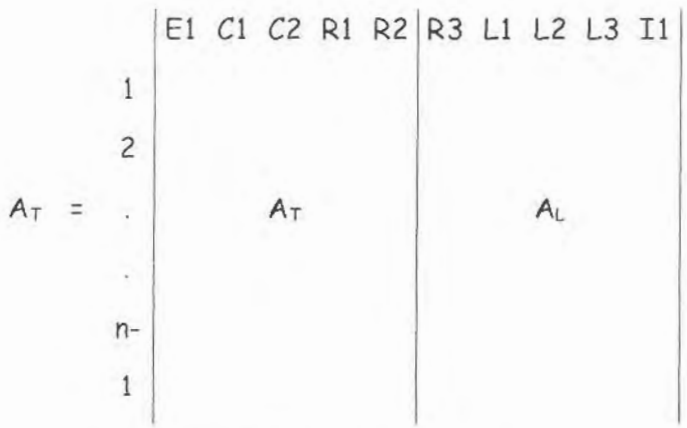


Figura 4.4.2.1. Matriz de incidencia A.

De acuerdo con el siguiente teorema, alguna n-1 columna linealmente independiente de A o A corresponde a un árbol del grafo de la red (se asume un grafo conectado con n nodos). Por lo tanto se elige un conjunto independiente de n-1 columnas, comenzando desde la izquierda y moviéndose sucesivamente a la derecha.

Teorema 1:

Siendo la matriz de incidencia de un grafo conectado G con nodos. Entonces n-1 columnas de A son linealmente independientes si y solo si las ramas correspondientes a estas columnas forman un árbol en G.

El reconocimiento de un conjunto de columnas linealmente independientes es realizado mediante la reducción de A a la forma echelón, a través de una serie de operaciones elementales de filas, tales como:

1. Intercambio de 2 filas.
2. Multiplicación de alguna fila por una constante escalar distinto de cero.
3. Reemplazo de la *j*th fila por la suma de la *j*th fila y $\propto$ veces de la *k*th fila, donde *k*= *j* y $\propto$ es alguna constante escalar.

4.4.3. Generación de C y D.

Asumiendo que un árbol T ha sido generado y que las matrices A, C y D son particionadas como:

$$A = [A_T | A_L] \qquad (4.4.3.1)$$

$$C = [C_T | 1_\mu] \qquad (4.4.3.2)$$

$$D = [1_\rho | D_L] \qquad (4.4.3.3)$$

El problema es generar B_T y D_L desde la matriz de incidencia A. Y de acuerdo al corolario.

$$AC^t = A_T C^t + A_l = 0 \qquad (4.4.3.4)$$

De donde tenemos que:

$$C^t{}_l = -A^{-1}{}_T A_l \qquad (4.4.3.5)$$

$$D_L = -C^{-1}{}_T \qquad (4.4.3.6)$$

Resolviendo simultáneamente las ecuaciones (4.4.3.5) y (4.4.3.6).

$$D = [1\rho | D_L] = A^{-1}{}_T [A_T | A_L] = A^{-1}{}_T A \qquad (4.4.3.7)$$

La ecuación (4.4.3.7) es la clave para el método de la generación de las matrices D y B. Esto quiere decir que si primero encontramos $A^{-1}{}_T$ y entonces premultiplicamos de A por $A^{-1}{}_T$ el resultado es D. Una vez que se encuentra D, obtenemos $C_T = -D^{-1}{}_L$. Aunque este método es correcto, esto es ineficiente, por que cada elemento de $A^{-1}{}_T$ ha sido encontrado explícitamente.

Las siguiente teoremas básicos de álgebra de matrices, nos inducen a encontrar un método mas eficiente para la obtención de las matrices $A^{-1}{}_T$.

Teorema 2: *Cada matriz elemental tiene su inverso, el cual es también elemental.*

Teorema 3: *Para poder llevar a cabo una operación elemental en una matriz Q, calcule el producto εQ, donde ε es la matriz elemental obtenida por el desarrollo de la operación fila en la matriz identidad.*

Siguiendo el algoritmo de Echelon, para reducir una matriz rectangular, Si Q es una matriz cuadrada nosingular, podremos siempre encontrar una secuencia de m operaciones elementales de fila que reducen Q a una matriz identidad.

De acuerdo al teorema 3 escribimos que:

$$\varepsilon_m \cdots \varepsilon_3 \varepsilon_2 \varepsilon_1 Q = 1 \qquad (4.4.3.8)$$

Donde ε_k puede ser de otro tipo 1,2 o 3., de la ecuación se puede deducir que:

$$Q^{-1} = \varepsilon_m \ldots\ldots \varepsilon_3 \varepsilon_2 \varepsilon_1$$

$$Q = \varepsilon^{-1}{}_m \ldots\ldots \varepsilon^{-1}{}_3 \varepsilon^{-1}{}_2 \varepsilon^{-1}{}_1 \qquad (4.4.3.9)$$

La ecuación (4.4.3.9) muestra que cualquier matriz nosingular se puede expresar como el producto de algunas matrices elementales, (estas matrices podrán ser diferentes, pero sus productos son todos iguales a Q).

Ahora bien de la ecuación (4.4.3.7), se puede rescribir similarmente a la ecuación (4.4.3.9).

$$A^{-1}{}_T = \varepsilon_m \ldots\ldots \varepsilon_3 \varepsilon_2 \varepsilon_1$$

$$D = A^{-1}{}_T A = (\varepsilon_m \ldots\ldots \varepsilon_3 \varepsilon_2 \varepsilon_1) A \qquad (4.4.3.10)$$

Se dice entonces que D (4.4.3.10) podrá ser obtenida de A desarrollando una secuencia de operaciones elementales las cuales cuando sean desarrolladas en A_T resultará una matriz identidad. Tomando estos dos factores simultáneamente, obtenemos el siguiente método alternativo de generar D y C desde A.

4.4.4. Algoritmo para reducir una matriz rectangular a una matriz Echelón.

Definición: La matriz rectangular A de tamaño m x n, se dice que es una matriz Echelón (matriz fila), o simplemente una matriz Echelón, si A cumple con los siguientes puntos:

1. La primera fila K, donde ---- no son cero y todas las filas permanecen, si alguna es cero.
2. En la ith fila (i=1, 2,..., k), el primer elemento distinto de cero es igual a la unidad, la columna en la cual ocurre siendo C_i.
3. $C1 < C2 < C3 < \ldots < C_k$.

Por ejemplo, la siguiente matriz A es un renglón de la matriz Echelón.

$$A = \begin{vmatrix} 0 & 1 & a_{13} & a_{14} & a_{15} & a_{16} & a_{17} & a_{18} \\ 0 & 0 & 0 & 0 & 1 & a_{26} & a_{27} & a_{28} \\ 0 & 0 & 0 & 0 & 0 & 1 & a_{37} & a_{38} \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & a_{48} \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}$$

Cualquier matriz rectangular A de orden mxn puede ser reducida a la forma Echelón por una serie de elementales operaciones en las filas. Ahora bien, descubriremos un algoritmo para llevar a cabo la reducción.

4.4.5. Algoritmo RE (Reducción a matriz Echelón)

Iniciando con una matriz A de orden mxn, este algoritmo lleva a cabo operaciones elementales sobre submatrices de A. Cuando el algoritmo termina, A es reducida a la forma Echelón.

1. Dado que A es una submatriz de A. Inicialmente $A_s = A$.
2. Inspeccionar A_s, columna por columna desde la izquierda hasta que una columna distinta a cero, columna C, es encontrada. Si no existe una columna tal, entonces $A_s = 0$, y rl algoritmo termina.
3. Inspeccionar la columna C, elemento por elemento de arriba hasta encontrar un elemento en la fila *r*.
4. Intercambiar la fila 1 y la fila *r* de A_s. (Note que A_s ha sido cambiada de la original. Todas las operaciones se han llevado a cabo en la ultima A_s).
5. Multiplicar la fila 1 de A_s por el reciproco del (1,C) elemento de A_s.
6. Reducir todos los elementos de bajo de la columna 1 y en la columna C de A_s por tres operaciones fundamentales de fila.

7. Dada que la nueva A_s consiste de los elemtos de bajo de la fila 1 y a la derecha de la columna C de última A_s. Si esta nueva A_s no existe el algoritmo termina. De otro modo ir al paso 2.

4.5. Descripción del programa principal.

En el programa principal refiriéndonos concretamente a la hoja de trabajo. Primeramente es leído el Netlist desplegando en pantalla los datos leídos y utilizados en la resolución del método MNA en el dominio del tiempo, posteriormente son leídos todos los procedimientos.

El primer procedimiento llamado es ejecutar( ) en el cual como parámetros principales son obtenidos sublistas de la lista Elements como Lista_Nodos, Valor_Comp, compo y control, además de ser ejecutados dentro de este procedimiento los procedimientos compatibles(compo), tiempos( ), Valores( ) y Lista( ).

Sobre la hoja de trabajo solo se despliega una lista de elementos con la cantidad de cada elemento existente sobre el circuito en análisis. En ese momento es ejecutado el procedimiento MatrizMNA( ) el cual obtiene principalmente la matriz MNAT y el vector de estímulos WT, los cuales son evaluados.

Como siguiente punto se ejecuta el procedimiento ciclo1(MNAT,WT) en donde resuelta la matriz MNA de acuerdo al número de pasos obtenidos previamente . En un gran arreglo son guardados los valores de todas las soluciones. Esto no es visto por el usuario, el tiempo en responder con un menú, para elegir la opción deseada depende de las características del ordenador en el cual se este ejecutando este software.

Posteriormente ejecuta el procedimiento opción( ) que presenta un menú en el cual da la posibilidad de elegir para graficar ya sea un voltaje nodal o una corriente de un elemento no MNA compatible.

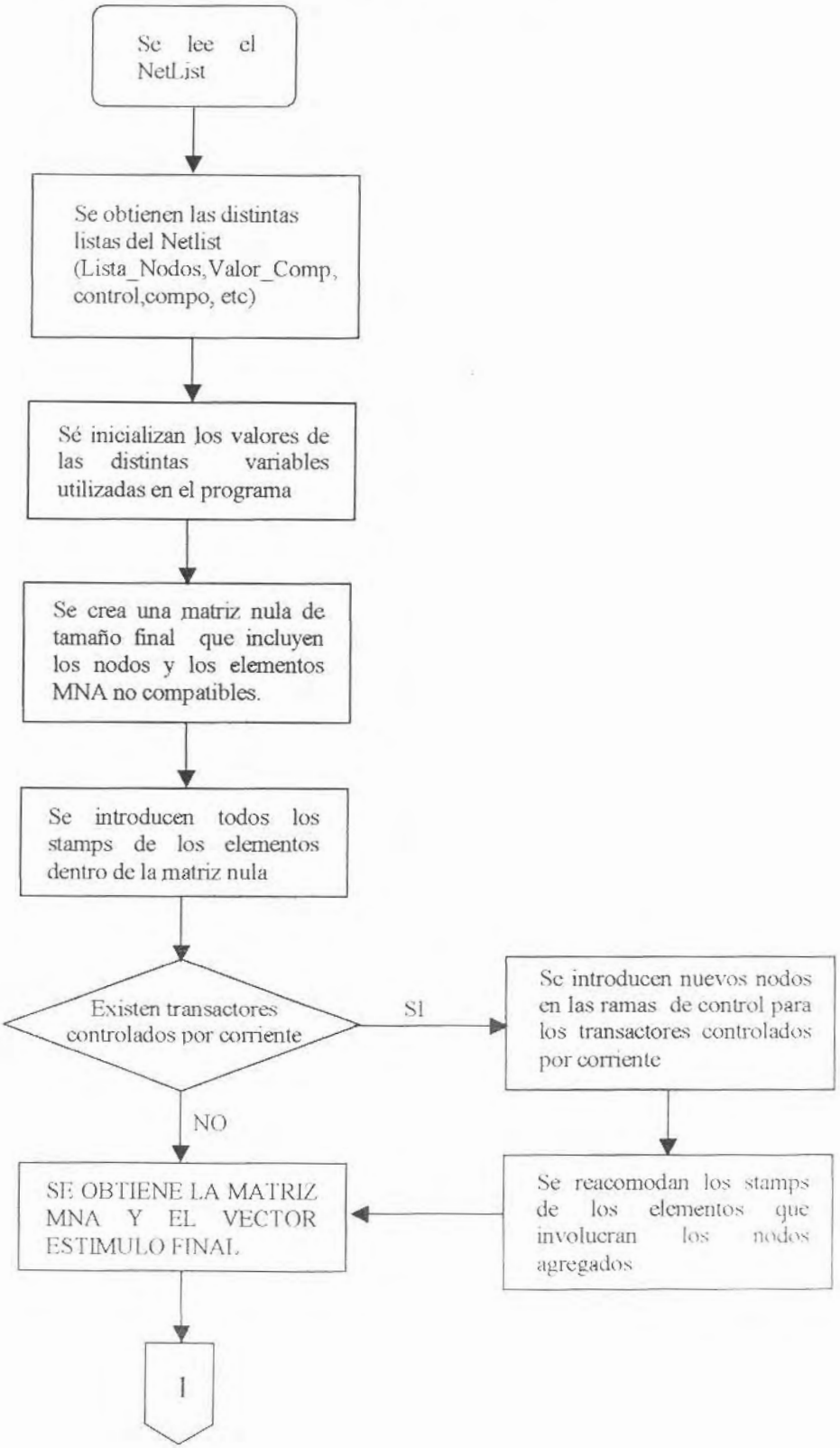
El usuario selecciona la opción y esta es capturada a partir del procedimiento ejecutado automáticamente captura( ), el valor de elección debe de ser introducido en línea de código para ser capturado.

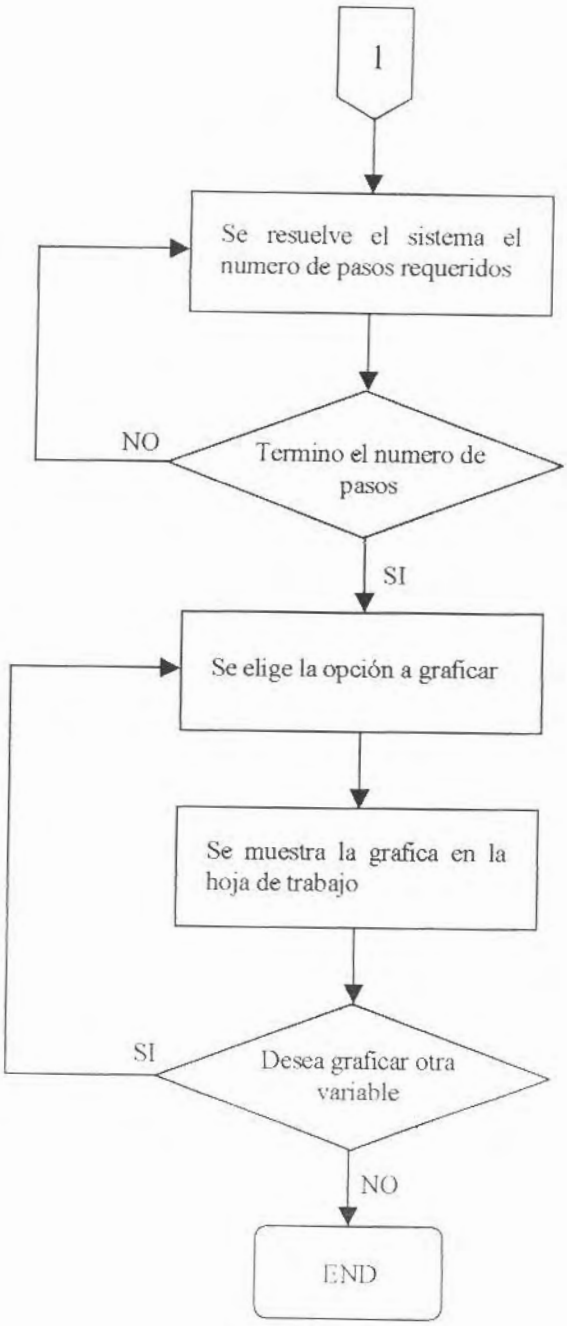
Como siguiente paso se llama el procedimiento opcion1( ) el cual de acuerdo a la opción deseada es la tarea que realizara permitiendo elegir concretamente ya sea voltaje de nodo o corriente de elemento no compatible a graficar.

Por ultimo es llamado el procedimiento graficar( ) que propiamente como su nombre lo dice, grafica el voltaje de nodo o corriente de elemento no compatible.

Si se desea obtener otra gráfica basta con regresar a la línea de llamada del procedimiento opcion1() para elegir la opción deseada, y repetir los pasos siguientes a esta función.

4.6. Diagrama de flujo del programa general.




1
Se resuelve el sistema el numero de pasos requeridos
NO
Termino el numero de pasos
SI
Se elige la opción a graficar
Se muestra la grafica en la hoja de trabajo
SI
Desea graficar otra variable
NO
END

4.7. Descripción de los procedimientos.

4.7.1. Lista de variables de los procedimientos.

Lista de los diversas variables globales utilizadas en los procedimientos y su descripción de cada una de ellas.

Lista_Nodos:	Lista de pares de nodos entre los cuales se encuentra ubicado un elemento.
Valor_Comp:	Lista de los valores de los elementos.
Control:	Lista de relación entre elementos controlados y elementos controladores.
Compo:	Es una lista donde están contenidos todos los elementos del circuito.
Compa y nocompa:	Representan el numero total de elementos compatibles y no compatibles
Un_nodos:	Número de nodos del circuito, sin incluir el nodo de referencia.
n_ext:	Nodos extras debido a elementos controlados por corriente (Método Berckeley).

nr:	Número de renglones y columnas extras debido a elementos MNA no compatibles.
MNAT:	Matriz MNA total del circuito, que incluye el numero de nodos (sin incluir el nodo de referencia), el numero de nodos añadidos debido a las fuentes controladas por corriente y los renglones y columnas extras debido a los elementos no MNA compatibles.
WT:	Representa el vector total de estímulos del circuito
i:	Esta variable es utilizada para identificar a cada elemento contenido en la lista ***compo*** dependiendo de su posición en dicha lista.
n_comp:	El número total de elementos de lista ***compo***.
Lista_Next:	Lista de los elementos de control de las fuentes dependientes controladas por corriente (***Lista_Next***).
Lista:	Lista contiene todos los nodos del circuito.
control:	La lista de las relaciones de las fuentes controladas.
i_ren:	Es el contador de los renglones y columnas de los nodos utilizados para los cortos circuitos de las fuentes controladas por corriente.
j:	Es el contador para los elementos no MNA compatibles. Identifica el renglón y columna donde se construirá el stamp.

Bandera:	Esta variable nos indica si la variable de control ya ha sido utilizada por otro transactor, si ya fue utilizada no es necesario crear otro nodo extra ya que fue creado uno con anterioridad para ese mismo elemento.
pos_n-ext:	Es una lista, esta variable nos indica el numero asignado al nodo extra utilizado por cada transactor, cada elemento de la lista tiene el valor del nodo utilizado dependiendo de la posición del elemento en el Netlist.
nuevo_nodo:	Esta variable nos indica el numero de nodo extra utilizado por los transactores y se utiliza para corregir los stamps de los elementos utilizados como variables de control.
Bandera1:	Esta variable nos indica si el elemento de la variable de control ya fue corregido su stamp debido al nodo extra añadido, si ya fue corregido lo deja igual si no corrige su stamp en la matriz MNA final.
cl y cc:	Son listas que contienen las condiciones iniciales de los inductores y capacitores estas listas de condiciones iniciales son extraídas del netlist.
xc y yc:	Son listas que contienen los nodos de inicio (**xc**) y de termino (**yc**) de los capacitores.

xl y yl:	Son listas que contienen los nodos de inicio (**xl**) y de termino (**yl**) de los inductores.
kl y kc:	Representa él número total de inductores y capacitores.
B:	Es una lista donde se encuentran los valores de los inductores del circuito, se utiliza en la solución de la matriz MNA.
tiempo:	Es la lista donde se encuentra el tiempo inicial y el tiempo final a simular, además contiene también el tamaño del paso (h).
h:	Es el ancho del paso (step), se obtiene del NetList.
pasos:	Es el numero total de iteraciones a ejecutar.
t:	Es el tiempo real en cada iteración.
jl:	Es la corriente del inductor en el tiempo actual.
Vc:	Es el voltaje del capacitor al tiempo actual.
in_cap:	Nos proporciona el índice del capacitor que sé esta ingresando en el stamp actual.
in_ind:	Nos proporciona el índice del inductor que sé esta ingresando en el stamp actual.
M3:	Es el arreglo con todos los vectores de solución para cada iteración

res:	En esta variable se guarda la primera opción es decir, si se desea graficar un voltaje nodal o la corriente en un elemento MNA no compatible.
resp:	En esta variable se guarda la segunda opción, es decir el numero de nodo que se va graficar o el elemento del que se desea la corriente.
renglon:	En esta variable se guarda el renglón y la columna en la que se encuentra ubicado la corriente del elemento MNA no compatible.
P:	Es el renglón del arreglo de soluciones donde se encuentran todos los puntos a graficar del nodo o elemento seleccionado.
P1:	Es un vector donde se encuentra todos los puntos del tiempo donde se va a graficar.
P3:	Es el vector que contiene las parejas de los puntos en tiempo y los valores de las soluciones para cada uno de estos tiempos. Este es el vector que sé gráfica.

4.7.2. Procedimientos generales de lectura del NetList.

Argumento(compo, control)

Dadas las listas **compo** y **control**, este procedimiento obtiene el elemento que involucra la variable controladora.

compatibles(compo)

Este procedimiento haciendo uso de la lista **compo** obtiene el numero de cada elemento y despliega una lista además de obtener dos vectores con los nodos iniciales de los capacitores e inductores (**xc y xl**) y otros dos vectores con los nodos finales (**yc y yl**) de los mismos elementos. También se obtiene el numero de ramas adicionales debidas a los componentes no MNA compatibles y crea una lista de los elementos de control de las fuentes dependientes controladas por corriente (**Lista_Next**).

Nodes(Lista_Nodos)

A partir de la lista **Lista_Nodos** este procedimientos crea una lista llamada Lista con todos los nodos del circuito (incluyendo el nodo de referencia) y además obtiene el numero de nodos sin incluir el nodo de referencia (**nu_nodos**).

4.7.3 Procedimientos para crear los stamps de los elementos.

MatrizK(Lista_Nodos,nu_nodos,n_ext,nr)

Dada la lista **Lista_Nodos** y las variables **nu_nodos**, **n_ext** y **nr**, el procedimiento **MatrizK** obtiene la matriz **MNAK** de dimensiones n x n, dicha matriz representa el stamp para una conductancia. El valor de la conductancia es tomado de la lista **Valor_Comp**.

MatrizR(Lista_Nodos,nu_nodos,n_ext,nr,j)

Dada la lista **Lista_Nodos** y las variables **nu_nodos**, **n_ext** y **nr**, el procedimiento **MatrizR** obtiene la matriz **MNAR** de dimensiones n x n, dicha matriz representa el stamp para una resistencia. Además se incluye un parámetro global "**j**" que identifica el renglón y columna donde se construirá el stamp para la resistencia, este será incrementado al salir del procedimiento. El valor de la resistencia es tomado de la lista **Valor_Comp**.

MatrizFVI(Lista_Nodos,nu_nodos,n_ext,nr,j)

Dada la lista **Lista_Nodos** y las variables **nu_nodos**, **n_ext** y **nr**, el procedimiento **MatrizFVI** obtiene la matriz **MNAFVI** de dimensiones n x n, y el vector **WFVI** de dimensiones n x 1 que representan el stamp para una fuente de voltaje independiente Además se incluye un parámetro global "**j**" que identifica el renglón y columna donde se construirá el stamp para la fuente de voltaje independiente, este será incrementado al salir del procedimiento.

MatrizFCI(Lista_Nodos,nu_nodos,n_ext,nr)

Dada la lista **Lista_Nodos** y las variables **nu_nodos**, **n_ext** y **nr**, el procedimiento **MatrizFCI** retorna el vector estímulo **WFCI** de dimensiones n x 1 que representa el stamp de una fuente de corriente independiente.

MatrizC(Lista_Nodos,nu_nodos,n_ext,nr)

Dado los parámetros **Lista_Nodos**, **nu_nodos**, **n_ext**, **nr**, el procedimiento **MatrizC** obtiene matriz **MNAC** y el vector estímulo **WTC** de dimensiones n x n y n x 1 respectivamente. Este procedimiento ocupa las variables globales "**h**" que es el tamaño del step usado para la integración y la variable **Vc** como voltaje de capacitor al tiempo n + 1, la variable **in_cap** nos proporciona el índice del capacitor que se esta ingresando en el stamp actual, y será incrementado al salir del procedimiento.

MatrizL(Lista_Nodos,nu_nodos,n_ext,nr)

Dado los parámetros **Lista_Nodos**, **nu_nodos**, **n_ext**, **nr** se obtiene la matriz **MNAL** y el vector estímulo **WTL** de dimensiones n x n y n x 1 respectivamente. Este procedimiento ocupa las variables globales "**h**" que es el tamaño del step usado para la integración y la variable **Jl** como corriente de inductor al tiempo n + 1, la variable **in_ind** nos proporciona el índice del inductor que se esta ingresando en el stamp actual, y será incrementado al salir del procedimiento.

MatrizOP(nu_nodos,n_ext,nr)

Dada las variables **nu_nodos**, **n_ext**, **nr**, este procedimiento obtiene la matriz **MNAOP**, cuyas dimensiones son de n x n. Este procedimiento modela el stamp de un amplificador operacional ideal, tomando sólo en cuenta las terminales inversora, no inversora y de salida. Se ocupa la variable global "**j**" que identifica el renglón y columna donde se construirá el stamp para el OPAMP, este será incrementado al salir del procedimiento.

MatrizVCCS(nu_nodos,n_ext,nr)

Dadas las variables **nu_nodos**, **n_ext**, **nr**, el procedimiento **MatrizVCCS** obtiene la matriz **MNAVCCS** de dimensiones n x n que representa el stamp de una fuente de corriente controlada por voltaje. Este procedimiento hace uso del procedimiento **Argumento** que obtiene el elemento que posee el voltaje de control. La relación numérica entre el voltaje de control y la corriente controlada se obtiene de **Valor_Comp**.

MatrizCCCS(nu_nodos,n_ext,nr)

Dadas las variables **nu_nodos**, **n_ext**, **nr**, el procedimiento **MatrizCCCS** obtiene la matriz **MNACCCS** de dimensiones n x n que representa el stamp de una fuente de corriente controlada por corriente. Este procedimiento hace uso del procedimiento **Argumento** que obtiene el elemento por donde circula la corriente de control. La relación numérica entre la corriente de control y la corriente controlada se obtiene de **Valor_Comp**.

MatrizCCVS(nu_nodos,n_ext,nr)

Dadas las variables **nu_nodos**, **n_ext**, **nr**, el procedimiento **MatrizCCVS** obtiene la matriz **MNACCVS** de dimensiones n x n que representa el stamp de una fuente de voltaje controlada por corriente. Este procedimiento hace uso del procedimiento **Argumento** que obtiene el elemento por donde circula la corriente de control. La relación numérica entre la corriente de control y el voltaje controlado se obtiene de **Valor_Comp**.

MatrizVCVS(nu_nodos,n_ext,nr)

Dadas las variables **nu_nodos**, **n_ext**, **nr**, el procedimiento **MatrizCCVS** obtiene la matriz **MNACCVS** de dimensiones n x n que representa el stamp de una fuente de voltaje controlada por voltaje. Este procedimiento hace uso del procedimiento **Argumento** que obtiene el elemento que posee el voltaje de control. La relación numérica entre la corriente de control y el voltaje controlado se obtiene de **Valor_Comp**.

MatrizcorrK(Lista_Nodos,nu_nodos,n_ext,nr)

Dadas la lista **Lista_Nodos** y las variables **nu_nodos**, **n_ext**, **nr**, el procedimiento **MatrizcorrK** obtiene la matriz **MNACK** de dimensiones n x n que representa al stamp de una conductancia por donde circula una corriente de control de un transactor controlado por corriente y donde se ha introducido un nuevo nodo

con el fin de poder sensar la corriente de control a través de un corto circuito entre uno de los nodos de la conductancia y el nuevo nodo.

MatrizcorrR(Lista_Nodos,nu_nodos,n_ext,nr)

Dadas la lista "**Lista_Nodos**" y las variables **nu_nodos**, **n_ext**, **nr**, el procedimiento **MatrizcorrR** obtiene la matriz **MNACR** de dimensiones n x n que representa al stamp de una resistencia por donde circula una corriente de control de un transactor controlado por corriente y donde se ha introducido un nuevo nodo con el fin de poder sensar la corriente de control a través de un corto circuito entre uno de los nodos de la resistencia y el nuevo nodo.

MatrizcorrL(Lista_Nodos,nu_nodos,n_ext,nr)

Dadas la lista "**Lista_Nodos**" y las variables **nu_nodos**, **n_ext**, **nr**, el procedimiento **MatrizcorrL** obtiene la matriz **MNACL** de dimensiones n x n y el vector **WTCL** de dimensiones n x 1 que representa al stamp de un inductor por donde circula una corriente de control de un transactor controlado por corriente y donde se ha introducido un nuevo nodo con el fin de poder sensar la corriente de control a través de un corto circuito entre uno de los nodos del inductor y el nuevo nodo.

MatrizcorrC(Lista_Nodos,nu_nodos,n_ext,nr)

Dadas la lista "**Lista_Nodos**" y las variables **nu_nodos**, **n_ext**, **nr**, el procedimiento **MatrizcorrC** obtiene la matriz **MNACC** de dimensiones n x n y el vector **WTCC** de dimensiones n x 1 que representa al stamp de un capacitor por donde circula una corriente de control de un transactor controlado por corriente y donde se ha introducido un nuevo nodo con el fin de poder sensar la corriente de control a través de un corto circuito entre uno de los nodos del capacitor y el nuevo nodo.

4.7.4 Procedimientos generales.

Aquí se describen los procedimientos necesarios para el creado de la matriz MNA así como los procedimientos generales para el análisis temporal del circuito dado por el Netlist de entrada.

Valores( )

En este procedimiento se inicializan todos las variables utilizadas a lo largo de programa, con los valores adecuados.

Llenar( )

Este procedimiento va insertando los stamps de los componentes del circuito en la matriz **MNAT** y en el vector estimulo **WT**. Este procedimiento hace uso de todos los procedimientos que crean un stamp para un elemento dado. Los componentes pueden ser MNA compatibles o no NM A compatibles. El llenado de la matriz **MNAT** y

el vector **WT** se hace desde el primer componente de la lista **compo**, hasta el ultimo a través de variable i.

Llenar1( )

Este procedimiento hace la corrección de los stamps de los componentes del circuito por donde circula una corriente de control, en la matriz **MNAT** y en el vector estimulo **WT**. Este procedimiento hace uso solo de los siguientes procedimientos **MatrizcorrK**, **MatrizcorrR**, **MatrizcorrL** y **MatrizcorrC**. La corrección de la matriz **MNAT** y el vector **WT** se hace solo para las fuentes dependientes controladas por corriente.

tiempos ( )

En este procedimiento se obtiene el numero total de pasos o iteraciones del que constara el ciclo solución. Este se obtiene restando el valor final menos el valor inicial y dividiendo entre el tamaño del paso.

ciclo1 ( )

En este procedimiento se resuelve el sistema de la matriz **MNAT** y el vector estimulo **WT** recursivamente mediante un ciclo for desde el valor inicial hasta el valor final, con un tamaño de paso igual a h. Primeramente se cargan las condiciones iniciales de los capacitores e inductores si es que existen. Estos valores servirán para los primeros valores que se introducen en la matriz según el método de backward Euler,

en cada paso del ciclo se obtiene un nuevo valor de voltajes en los capacitores y un nuevo valor de la corriente en los inductores.

Cada paso del ciclo se obtiene un nuevo vector de soluciones que se van anexando a un gran arreglo que contiene todas las soluciones para cada paso.

4.7.5 Procedimientos que se utilizan para seleccionar la opción a graficar y para graficar.

opción( )

En este procedimiento se muestra un pequeño menú que indica las opciones para graficar.

captura( )

En este procedimiento se captura la opción seleccionada ya sea graficar un voltaje nodal o graficar la corriente de un elemento MNA no compatible.

opcion1( )

Este procedimiento indica las instrucciones a seguir, ya sea teclear el nodo deseado a graficar o teclear el elemento del que se desea la corriente.

capturas ( )

Este procedimiento captura la segunda opción, es decir el numero de nodo o el elemento MNA no compatible

mataux ( )

Este procedimiento crea un vector auxiliar para graficacion, este contiene el valor de todos los puntos del tiempo que se van a graficar.

grafi ( ) y grafiaux ( )

Estos procedimientos crean el vector final a graficar, este vector contiene los pares coordenados de tiempo del valor y el valor del voltaje o la corriente en este tiempo.

graficar ( )

Este procedimiento se encarga de mandar a graficar la opción seleccionada ya sea el voltaje de un nodo o la corriente de un elemento MNA no compatible.

4.7 Tutorial para el diseñador.

Estructura del Netlist como la descripción simbólica del circuito. Por lo tanto cada elemento del circuito queda definido por el tipo de elemento, los nodos entre los que se encuentra conectado, el valor numérico de cada elemento y su posible relación de control en el caso de una fuente dependiente.

Circuito ejemplo:

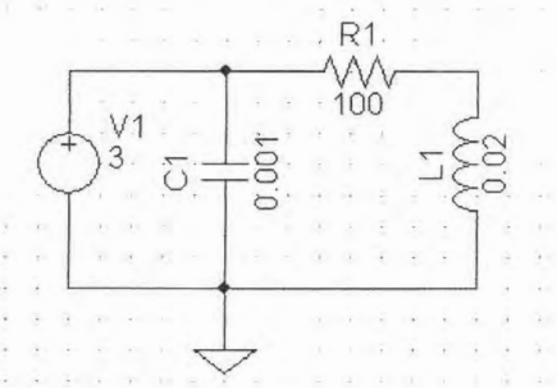


Figura 4.7.1. Circuito RLC.

Para realizar la simulación de un circuito primero se construye su Netlist. Esta se hace de la siguiente manera:

- En un archivo de texto se declara la variable Elements, la cual tiene la siguiente sintaxis.

Elements:=[[Lista de componentes],[Lista de nodos inicial y final de cada componente],[Lista de valores de cada componente],[Lista de las variables de control],[Valores iniciales de voltaje de capacitores],[Valores iniciales de corriente de inductores],[Tiempos de la simulación]];

Representación de circuito de la Figura 4.7.1.

```
Elements:=[
        [V1, R1, C1, L1],              Elementos.
        [[1, 0], [1, 2],[1, 0],[2, 0]], Nodos correspondientes a los elementos.
        [3, 100, .001, .02 ],          Valores correspondientes a los elementos.
        [[0], [0], [0]],               variables de control.
        [0],                           valores iniciales de voltaje de capacitores.
        [0],                           valores iniciales de corriente de inductor.
        [0,10,0.5]                     Tiempo de simulación.
];
```

Todos los elementos y subelementos deberán estar encerrados entre corchetes y separados por comas para diferenciar cada elemento.

Como se puede observar la variable Elements es una lista conformada por 7 elementos que en el lenguaje Maple se conocen como Listas.

En la primera lista se definen los componentes del circuito, para nuestro simulador se tiene la siguiente nomenclatura para cada uno de los componentes.

Kx	Conductancia
Rx	Resistencia
Cx	Capacitor
Lx	Inductor
OPx	Amplificador Operacional
Vx	Fuente Independiente de Voltaje
Jx	Fuente Independiente de Corriente

Gx	Fuente de Corriente Controlada por Voltaje
Hx	Fuente de Voltaje Controlada por Corriente
Fx	Fuente de Corriente Controlada por Corriente
Ex	Fuente de Voltaje Controlada por Voltaje

El orden de los elementos debe ser ascendente, ejemplo:

[R1, C1, K1, R2, R3,.....]

A esta lista el procedimiento **todos** asigna el nombre de **compo**, de la siguiente forma:

compo:=Elements[1];

La segunda lista debe contener el par ordenado de nodos entre los cuales se encuentra ubicado el componente y por la estructura del programa el orden de los pares ordenados de nodos de esta lista, debe corresponder con el orden de los componentes en la lista de componentes. Es decir el primer par ordenado de nodos corresponde con el primer componente de la lista de componentes.

Para el caso del amplificador operacional se tiene tres nodos diferentes siendo su estructura la siguiente:

[entrada inversora, entrada no inversora , salida]

A esta lista el procedimiento **todos** asigna el nombre de **Lista_Nodos**, de la siguiente forma:

Lista_Nodos:=Elements[2];

Los nodos que contenga el circuito total deberán de ser ascendentes esto es que los números deberán ser continuos y no deberán ser salteados como se ve en los ejemplos anexos.

La tercera lista debe contener los valores numéricos de los componentes asociados a las funciones de rama de los mismos, que pueden representa corriente, voltaje, resistencia, capacitancia, etc. Esto dependerá del elemento relacionado con este valor. En esta lista como en la anterior el orden en que se introducen los valores, debe ser el mismo orden que el de los componentes.

El procedimiento **resolver** asigna a esta lista el nombre de **Valor_Comp** , de la siguiente forma:

Valor_Comp:=Elements[3];

NOTA: Cabe aclarar que para el caso del amplificador operacional, en la lista Valor_Comp deberá colocarse un cero en el lugar correspondiente.

En la cuarta lista se colocan las relaciones de control de las fuentes controladas. El numero de elementos de esta lista es igual al numero de fuentes controladas. La nomenclatura para cada elemento de esta lista es la siguiente:

[Gx, V(Compx)]	Fuente de corriente controlada por voltaje. Donde Compx es el componente que posee el voltaje de control y puede ser una resistencia, una conductancia, un inductor, o un capacitor.
[Hx, J(Compx)]	Fuente de voltaje controlada por corriente. Donde Compx es el componente por donde circula la corriente de control y puede ser una resistencia, una conductancia, un

	inductor, o un capacitor.
[Fx, J(Compx)]	Fuente de corriente controlada por corriente. Donde Compx es el componente por donde circula la corriente de control y puede ser una resistencia, una conductancia, un inductor, o un capacitor.
[Ex, V(Compx)]	Fuente de voltaje controlada por voltaje. Donde Compx es el componente que posee el voltaje de control y puede ser una resistencia, una conductancia, un inductor, o un capacitor.

Cuando no existen fuentes controladas esta lista debe contener un cero.

La quinta lista debe contener los valores iniciales de voltaje de todos los capacitores del circuito, el orden de los elementos de esta lista debe corresponder con el orden en el que aparecen en la lista de componentes. En caso de que no existan capacitores en el circuito, esta lista debe contener un cero.

El procedimiento **compatibles** asigna a esta lista el nombre de **cc**, de la siguiente forma:

```
cc:=Elements[5];
```

La sexta lista deberá contener los valores iniciales de la corriente de todos los inductores del circuito, el orden de los elementos de esta lista debe corresponder con el orden con que aparecen en la lista de componentes.

El procedimiento **compatibles** asigna a esta lista el nombre de **cl** de la siguiente forma:

```
cl:=Elements[6];
```

La última lista contiene los tiempos inicial, final y el valor de paso de integración. La estructura de esta lista es la siguiente.

[Tiempo inicial, tiempo final, paso]

El procedimiento **compatibles** asigna el nombre de tiempo a esta lista de la siguiente forma:

```
tiempo:=Elements[7];
```

Para hacer la simulación de un circuito se abre una hoja de trabajo, se reinicia *(restart)* la hoja de trabajo, se cargan las librerías de álgebra lineal por medio del comando *with(linalg)* y a continuación se cargan todos los archivos donde se encuentran los procedimientos usados por el simulador.

Después de haber realizado todo lo anterior se ejecuta el procedimiento **ejecutar( )**, que muestra una tabla de todos los elementos que reconoce el simulador y el número de cada uno de ellos en el circuito, a partir de ese momento realiza el proceso de solución del circuito esto pondrá en un lapso de tiempo que dependerá de la velocidad del procesador en donde se ejecute, mostrándonos al final las opciones de graficar un voltaje nodal o una corriente de un elemento no compatible.

Automáticamente se ejecutara el procedimiento para la solución del sistema de la matriz MNA **ciclo1( )**.

Para poder graficar:

A partir de la línea de **opción1 ( )** uno puede escoger el voltaje o corriente a graficar, este deberá ser introducido como línea de código, en caso de ser un voltaje nodal deberá de ser el numero de nodo a graficar y en caso de ser una corriente de un elemento MNA no compatible deberá de ponerse el nombre con mayúsculas, y posteriormente ejecutar la función **graficar()** y con eso bastara para visualizar la grafica.

5. Resultados, Conclusiones y Perspectivas.

Resultados

Una de las partes importantes en cualquier trabajo o proyecto son los resultados obtenidos. Esta sección muestra los resultados obtenidos, la forma de mostrarlos es mediante la resolución de circuitos ejemplo analizándolos en DC ya que es el único análisis que realiza este software.

Ejemplo 1.

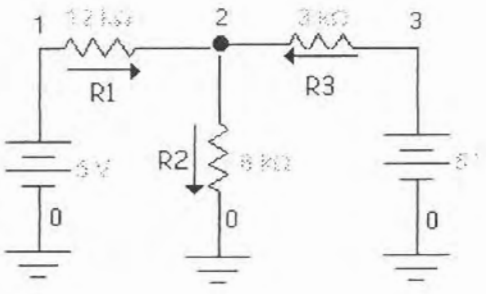


figura 5.1.1. circuito ejemplo 1.

En este ejemplo se analizara en DC o sea se analiza para obtener el punto de operación del circuito y el comportamiento en DC de los elementos en corrientes y de los nodos producto de la corriente de los nodos los cuales son presentados en la tabla 5.1.1.

Resolviendo el circuito en análisis CD obtenemos los siguientes resultados.

Nodo	Voltaje
1	5v
2	3.5714v
3	5v

Elemento	Corriente en
R1	119mA
R2	595mA
R3	-476.1mA

Tabla 5.1.1. Resultados obtenidos del circuito de la figura 5.1.1.

Los datos obtenidos en la tabla 5.1.1., fueron obtenidos resolviendo a mano el circuito de la figura 5.1.1., y para comprobar que los resultados obtenidos mediante el software desarrollado en este trabajo se presentan las siguientes graficas.

Las graficas que a continuación se muestran son los resultados obtenidos por el software desarrollado en Maple V. Release 5.

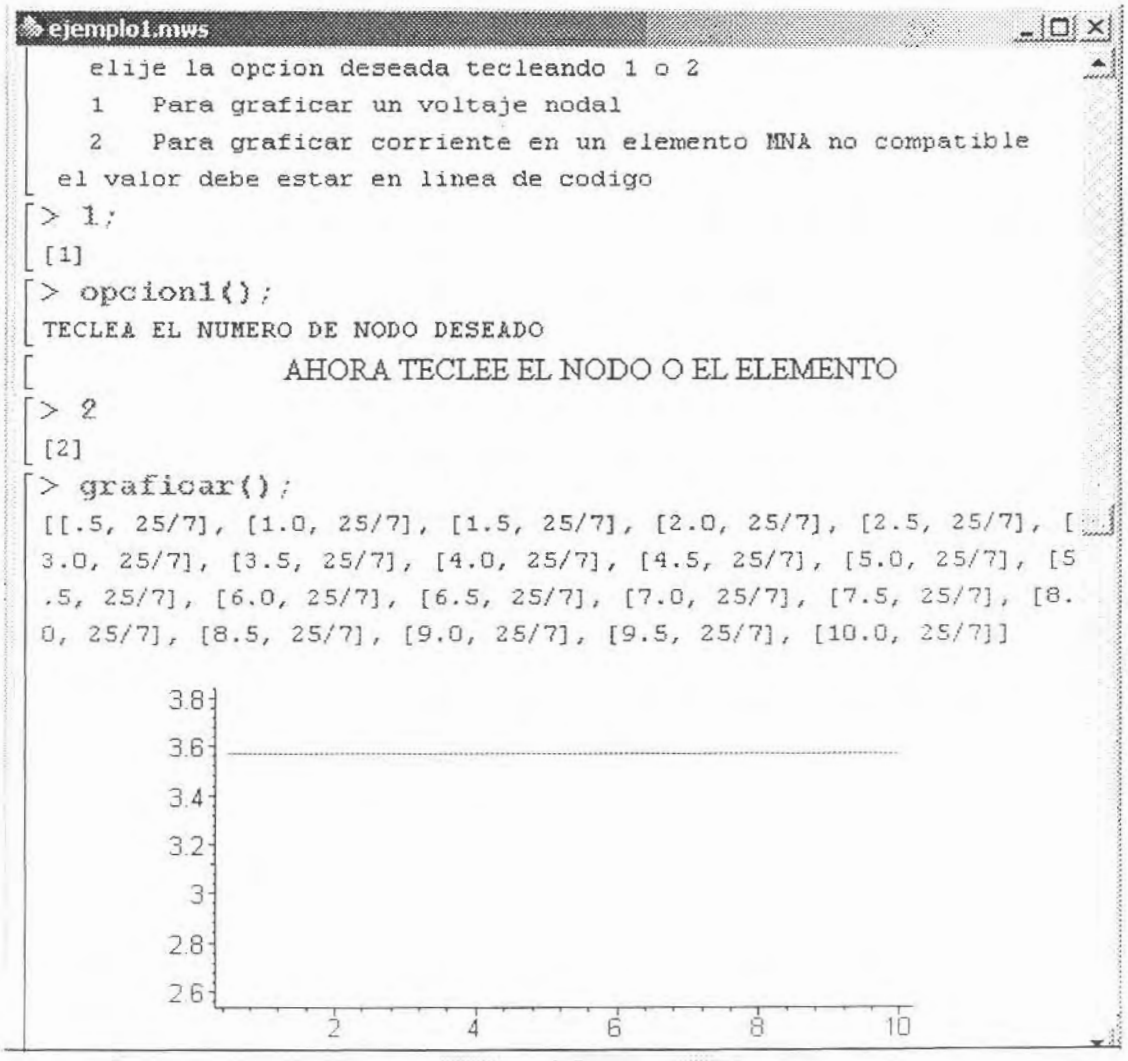


Figura 5.1.2. Voltaje del nodo 2 del circuito ejemplo1.

Los voltajes del nodo 1 y 3 como puede verse en la figura 5.1.1. son nodos que se encuentran conectados a las fuentes de alimentación por lo tanto tienen el voltaje de la fuentes correspondientes.

La corriente que circula en cada elemento R1, R2 Y R3 se presentan en la figura 5.1.3, 5.1.4 y 5.1.5 respectivamente.

ejemplo1.mws

```
   elije la opcion deseada tecleando 1 o 2
   1   Para graficar un voltaje nodal
   2   Para graficar corriente en un elemento MNA no compatible
 el valor debe estar en linea de codigo
> 2;
[2]
> opcion1();
TECLEA EL ELEMENTO MNA NO COMPATIBLE DEL QUE DESEAS LA CORRIENTE
                AHORA TECLEE EL NODO O EL ELEMENTO
> R1
["R1"]
> graficar();
[[.5, 5/42], [1.0, 5/42], [1.5, 5/42], [2.0, 5/42], [2.5, 5/42],
[3.0, 5/42], [3.5, 5/42], [4.0, 5/42], [4.5, 5/42], [5.0, 5/42],
[5.5, 5/42], [6.0, 5/42], [6.5, 5/42], [7.0, 5/42], [7.5, 5/42],
[8.0, 5/42], [8.5, 5/42], [9.0, 5/42], [9.5, 5/42], [10.0, 5/42]]
p1
p3
```

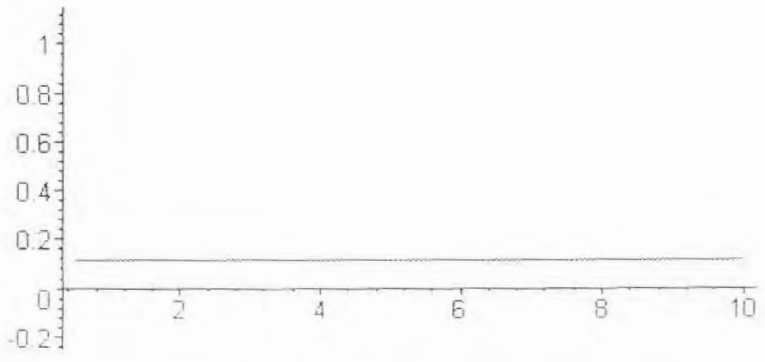


figura 5.1.3 Grafica de la corriente que circula por R1 del circuito ejemplo1.

ejemplo1.mws

```
   elije la opcion deseada tecleando 1 o 2
   1   Para graficar un voltaje nodal
   2   Para graficar corriente en un elemento MNA no compatible
 el valor debe estar en linea de codigo
> 2;
[2]
> opcion1();
TECLEA EL ELEMENTO MNA NO COMPATIBLE DEL QUE DESEAS LA CORRIENTE
                AHORA TECLEE EL NODO O EL ELEMENTO
> R2
["R2"]
> graficar();
[[.5, 25/42], [1.0, 25/42], [1.5, 25/42], [2.0, 25/42], [2.5, 25/
42], [3.0, 25/42], [3.5, 25/42], [4.0, 25/42], [4.5, 25/42], [5.0
, 25/42], [5.5, 25/42], [6.0, 25/42], [6.5, 25/42], [7.0, 25/42],
[7.5, 25/42], [8.0, 25/42], [8.5, 25/42], [9.0, 25/42], [9.5, 25/
42], [10.0, 25/42]]
```

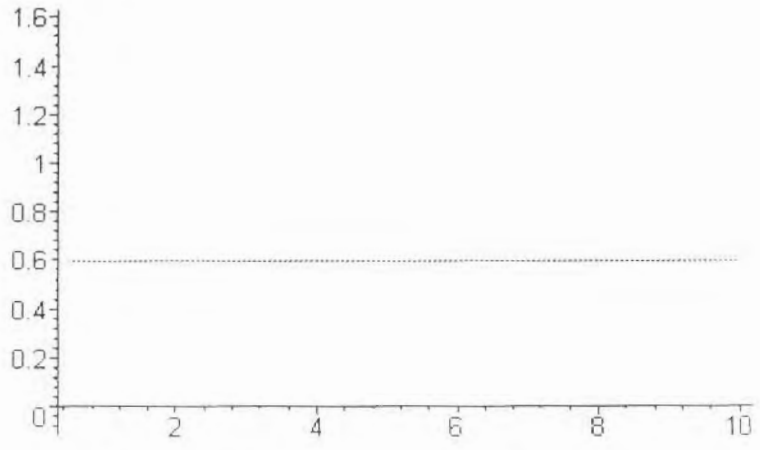


figura 5.1.4 Grafica de la corriente que circula por R2 del circuito ejemplo1.

ejemplo1.mws

```
   elije la opcion deseada tecleando 1 o 2
   1   Para graficar un voltaje nodal
   2   Para graficar corriente en un elemento MNA no compatible
 el valor debe estar en linea de codigo
> 2;
[2]
> opcion1();
TECLEA EL ELEMENTO MNA NO COMPATIBLE DEL QUE DESEAS LA CORRIENTE
                AHORA TECLEE EL NODO O EL ELEMENTO
> R3
["R3"]
> graficar();
[[.5, -10/21], [1.0, -10/21], [1.5, -10/21], [2.0, -10/21], [2.5,
-10/21], [3.0, -10/21], [3.5, -10/21], [4.0, -10/21], [4.5, -10/2
1], [5.0, -10/21], [5.5, -10/21], [6.0, -10/21], [6.5, -10/21], [
7.0, -10/21], [7.5, -10/21], [8.0, -10/21], [8.5, -10/21], [9.0,
-10/21], [9.5, -10/21], [10.0, -10/21]]
```

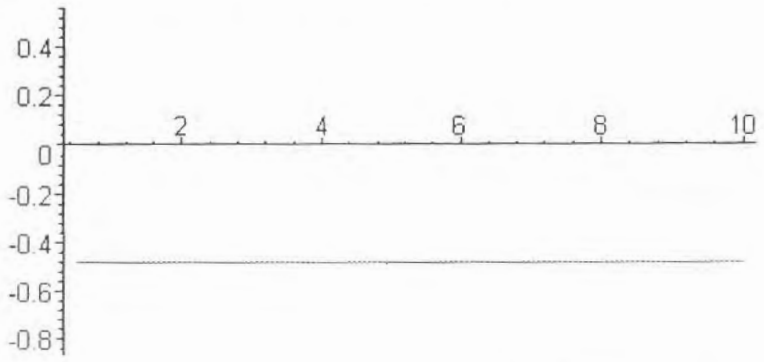
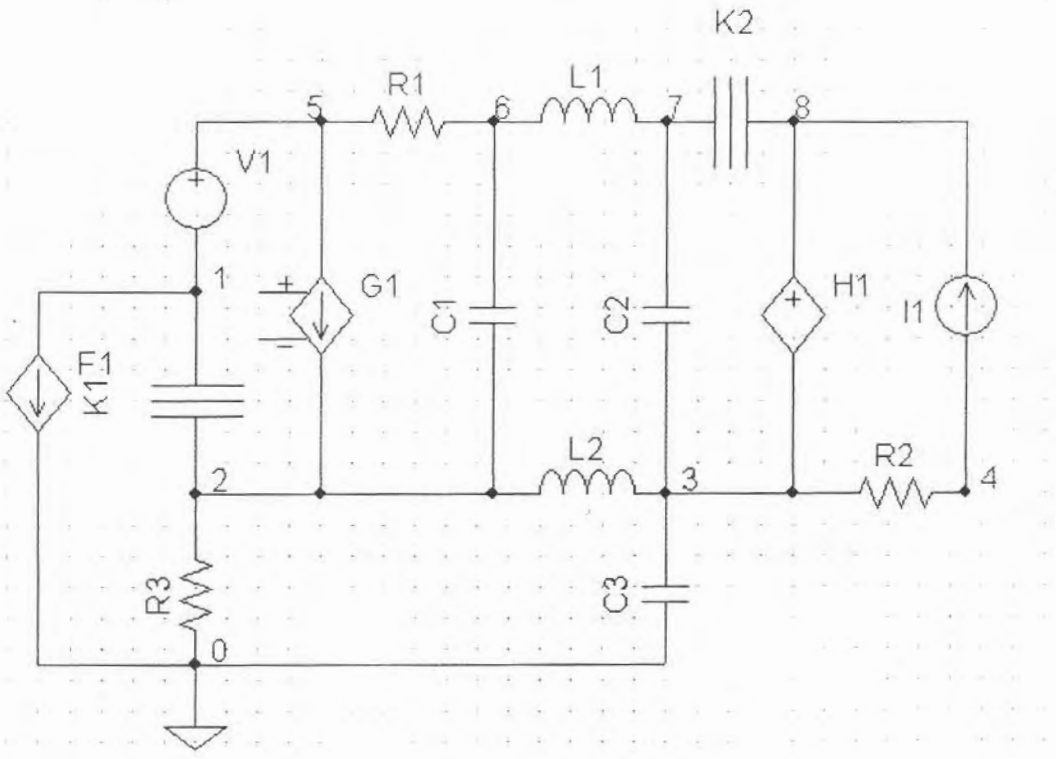


Figura 5.1.5. Grafica de la corriente que circula por R3 del circuito ejemplo1.

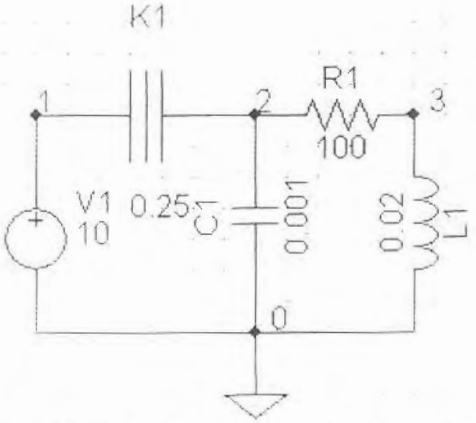
El mismo circuito analizado en el simulador SPICE nos da los siguientes resultados que son iguales a los obtenidos.

```
*******03-JUN-2001*****SPICEw32 5.82b Win32 *******13:59:13********
 circuito r
 ****     INPUT LISTING                    TEMPERATURE =   27.000 DEG C
 ************************************************Ser#582TN9070426 ***

 .op

 v1  1 0 5v
 v2 3 0 5v

 r1 1 2 12k
 r2 2 0 6k
 r3 3 2 3k


 .end
*******03-JUN-2001*****TopSPICEw32 5.82b Win32 *******13:59:13********
 circuito r
****     SMALL SIGNAL BIAS SOLUTION       TEMPERATURE =   27.000 DEG C

 ************************************************Ser#582TN9070426 ***

                NODE       VOLTAGE

+               V(1)       5.00000000E+00
+               V(3)       5.00000000E+00
+               V(2)       3.57142857E+00

*******03-JUN-2001*****TopSPICEw32 5.82b Win32 *******13:59:13********

 circuito r

 ****     SOURCE VOLTAGES AND CURRENTS     TEMPERATURE =   27.000 DEG C

 ************************************************Ser#582TN9070426 ***

    NAME       CURRENT/VOLTAGE

    V1       -1.19048E-04 AMPS
    V2       -4.76190E-04 AMPS

    TOTAL POWER DISSIPATION   2.97619E-03 WATTS
0
        JOB CONCLUDED

        Simulation Time          0.02 secs

        TOTAL JOB TIME           0.03 secs
```

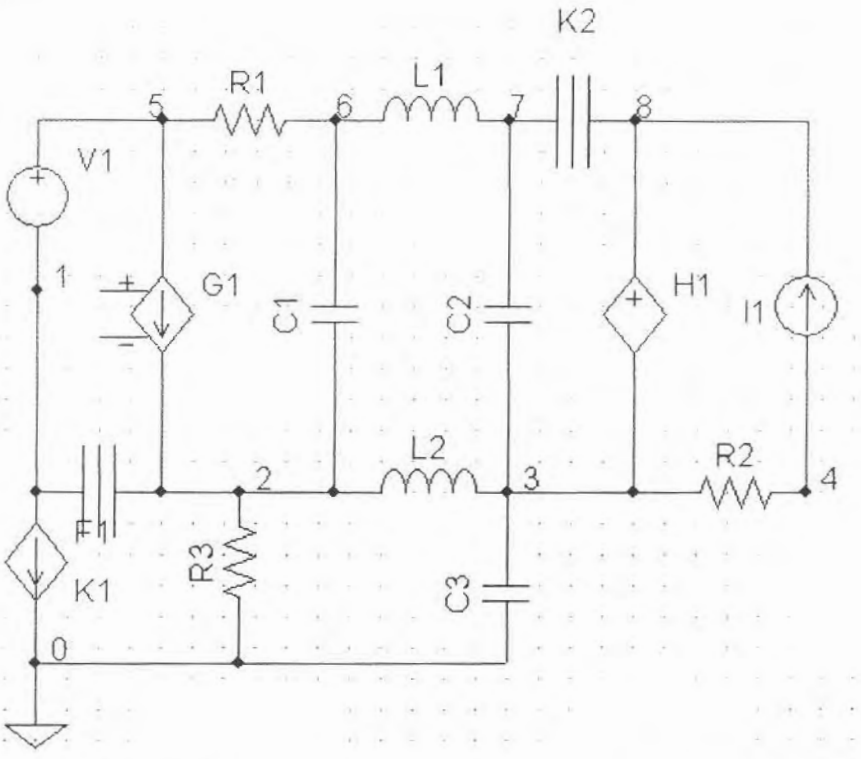
NOTA: los ejemplos que a continuación se muestran, primeramente se muestra el esquemático y posteriormente se muestran los resultados obtenidos.



Circuito ejemplo 2 en su forma Esquemática.



Circuito ejemplo 3 en su forma Esquemática.



Circuito ejemplo 4 en su forma Esquemática.

Nota: En esta sección se muestran los esquemáticos de los circuitos analizados como ejemplos para demostrar el funcionamiento del sistema, cabe mencionar en este punto que a pesar de mostrar los esquemáticos, el software no realiza análisis de este tipo.

Ejemplo2.

ejemplo2.mws

```
> restart;
> with (linalg):
Warning, new definition for norm
Warning, new definition for trace
> read( d:\\tesis\\NetList1.txt ):
```

$$Elements = [[V1, K1, R1, K2, R2, J1, L1, C1, C2, L2, R3, K3, G1, H1, F1, E1],$$
$$[[1,5],[1,2],[5,6],[7,8],[3,4],[4,8],[6,7],[6,2],[7,3],[2,3],[2,0],[3,0],[5,2],[8,3],[1,0],[4,0]],$$
$$[1,1,1,1,1,1,2,3,3,2,1,1,11,12,13,14],[[G1,V(R1)],[H1,J(K3)],[F1,J(K2)],[E1,V(R1)]],[5,2],[1,2],$$
$$[1,50,.8]]$$

```
> read( d:\\tesis\\resolver.txt ):
> ejecutar();
```

Conductancias:, 3

Resistencias:, 3

Capacitancias:, 2

Inductancias:, 2

Fuentes de Voltaje Independientes:, 1

Fuentes de Corriente Independientes:, 1

FCCC:, 1

FVCV:, 1

FCCV:, 1

FVCC:, 1

Opamps:, 0

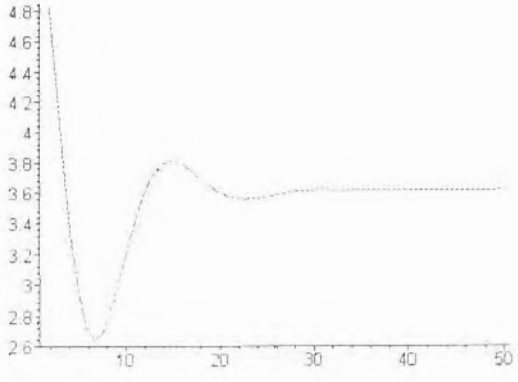
```
   elije la opcion deseada tecleando 1 o 2
    1   Para graficar un voltaje nodal
    2   Para graficar corriente en un elemento MNA no compatible
 el valor debe estar en linea de codigo
> 1;
```

[1]

```
> opcion1();
TECLEA EL NUMERO DE NODO DESEADO
                AHORA TECLEE EL NODO O EL ELEMENTO
> 6
```

[6]

```
> graficar();
```



Ejemplo3.

ejemplo3.mws

```
> restart;
> with (linalg);
Warning, new definition for norm
Warning, new definition for trace
> read('d:\\tesis\\HotList1.txt');
```

$$Elements = [[V1, K1, R1, K2, R2, J1, L1, C1, C2, L2, R3, K3, G1, H1, F1, E1], [[1, 5], [1, 2], [5, 6], [7, 8], [3, 4], [4, 8], [6, 7], [6, 2], [7, 3], [2, 3], [2, 0], [3, 0], [5, 2], [8, 3], [1, 0], [4, 0]], [1, 1, 1, 1, 1, 1, 2, 3, 3, 2, 1, 1, 11, 12, 13, 14], [[G1, V(R1)], [H1, J(R3)], [F1, J(K2)], [E1, V(R1)]], [5, 2], [1, 2], [1, 50, 8]]$$

```
> read('d:\\tesis\\resolver.txt');
> ejecutar();
```

Conductancias:, 3

Resistencias:, 3

Capacitancias:, 2

Inductancias:, 2

Fuentes de Voltaje Independientes:, 1

Fuentes de Corriente independientes:, 1

FCCC:, 1

FVCV:, 1

FCCV:, 1

FVCC:, 1

Opamps:, 0

```
   elije la opcion deseada tecleando 1 o 2
   1   Para graficar un voltaje nodal
   2   Para graficar corriente en un elemento MNA no compatible
 el valor debe estar en linea de codigo
> 1;
```

[1]

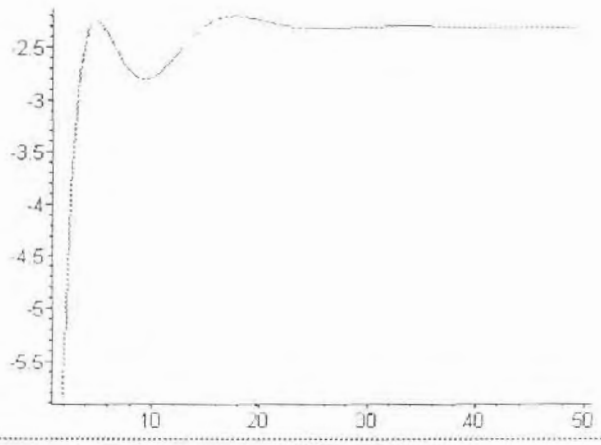
```
> opcion1();
TECLEA EL NUMERO DE NODO DESEADO
```

AHORA TECLEE EL NODO O EL ELEMENTO

```
> 4
```

[4]

```
> graficar();
```



Ejemplo 4.

ejemplo1.mws

```
> restart;
> with (linalg);
Warning, new definition for norm
Warning, new definition for trace
> read( d:\\tesis\\NetList.txt );
```

$$Elements = [[V1, K1, R1, C1, L1], [[1, 0], [1, 2], [2, 3], [2, 0], [3, 0]], [10, .25, 100, .001, .02], [[0], [0], [0], [0]], [0], [0], [0, 10, .5]]$$

```
> read( d:\\tesis\\resolver.txt );
> ejecutar();
```

$$Conductancias:, 1$$

$$Resistencias:, 1$$

$$Capacitancias:, 1$$

$$Inductancias:, 1$$

$$Fuentes\ de\ Voltaje\ Independientes:, 1$$

$$Fuentes\ de\ Corriente\ Independientes:, 0$$

$$FCCC:, 0$$

$$FVCV:, 0$$

$$FCCV:, 0$$

$$FVCC:, 0$$

$$Opamps:, 0$$

```
   elije la opcion deseada tecleando 1 o 2
   1   Para graficar un voltaje nodal
   2   Para graficar corriente en un elemento MNA no compatible
 el valor debe estar en linea de codigo
> 2;
```

$$[2]$$

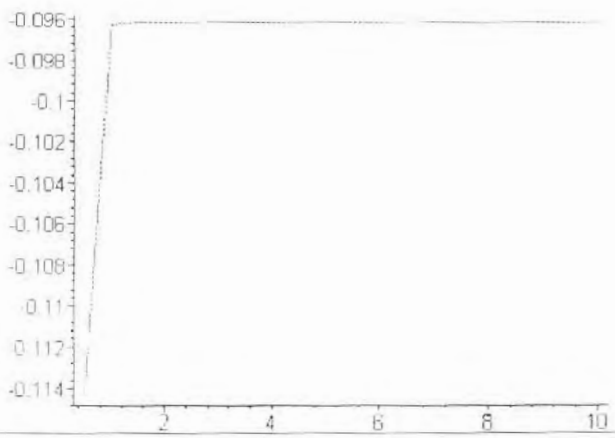
```
> opcion1();
TECLEA EL ELEMENTO MNA NO COMPATIBLE DEL QUE DESEAS LA CORRIENTE
```

AHORA TECLEE EL NODO O EL ELEMENTO

```
> V1
```

$$["V1"]$$

```
> graficar();
```



Conclusiones

En este trabajo se ha desarrollado una herramienta simple que integra una Herramienta CAD la cual realiza un solo tipo de análisis (análisis en CD), cabe mencionar que a pesar de ser un solo tipo de análisis, es el más importante de todos ellos, por que proporciona el punto de operación del circuito a analizar, en el que se basan los demás análisis, puesto que una vez linealizado o encontrado el punto de operación del circuito, es posible realizar el análisis en AC.

El sistema desarrollado tiene una resolución aceptable con respecto a simuladores comerciales existentes, por contar con un buen grado de confiablidad en los resultados obtenidos por el sistema, resaltando que el análisis en DC que realiza el software desarrollado maneja modelos ideales de los elementos de circuito.

Perspectivas.

Este trabajo se basó solo en el análisis de CD, y queda abierto para trabajos futuros donde sea posible aumentar características en los modelos implementados haciendo con esto que el sistema tiende ha asemejarse más al comportamiento de un dispositivo real.

El sistema está estructurado para que se le pueda implementar el análisis en AC, a él mismo se le puede hacer el mejoramiento de los algoritmos utilizados (Newton Raphson) así como el análisis transitorio.

GLOSARIO

Árbol. Un árbol es un subgrafo de un grafo original. Para considerarse como un árbol, el subgrafo debe contener todos los nodos del grafo, formando un subgrafo conectado.

Lazo. Es un conjunto de ramas que forman una trayectoria cerrada.

Grafo. La estructura o configuración de una red esta dada por el grafo de la red. Un grafo se construye, remplazando cada elemento de la red por un segmento de línea. Hay casos, donde un segmento de línea puede remplazar una fuente de voltaje y/o corriente y/o más elementos pasivos.

Nodo. La unión de dos o más segmentos de línea se le llama nodo.

Rama. A un segmento de línea de un grafo que conecta a dos nodos, se le llama rama. Una rama podrá remplazar un solo elemento o la combinación de varios elementos de la red.

Rama de árbol. Cualquier rama de un árbol en particular bajo consideración, se dice que es una rama de árbol.

Rama de coárbol. Cualquier rama del grafo que no corresponda el árbol particular en consideración.

Un Circuito Eléctrico. Es una colección de *elementos* eléctricamente interconectados. Es una entidad que esta gobernada por leyes físicas; El flujo de carga eléctrica es un fenómeno discreto, que es el número de electrones, debido a las conexiones de los

elementos y a las características de los elementos de un circuito físico debido a esto es posible considerar que el flujo de carga es un fenómeno continuo.

Leyes de Kirchoff.- Las leyes de Kirchoff son una simplificación de las *leyes de Maxwell*, convertidas en dos leyes fundamentales en la teoría de circuitos conocidas como ley de corrientes de Kirchoff (KCL) y ley de voltajes de Kirchoff (KVL).

Ley de Voltajes de Kirchhoff (KVL). La suma algebraica de las fuentes de voltaje y las caídas de voltaje **IR** a lo largo de cualquier trayectoria cerrada es cero.

KVL. Para todo circuito *lumped* en cualquier momento t, para cualquier nodo de referencia y cualquier par de nodos i y j se cumple:

$$u_{k-j}(t) = vk(t) - vj(t)^{1}.$$

Ley de Corrientes de Kirchhoff (KCL). En cualquier punto de un circuito, la suma algebraica de las corriente que se dirigen a tal punto y de las que salen de él es cero.

KCL. Para todo circuito *lumped* en cualquier tiempo t, la suma algebraica de todas las corrientes a través de una *superficie gaussiana* es igual a cero.

Circuito Físico. Es una colección de *dispositivos* eléctricamente interconectados.

Fanaout. Es la medida que indica la capacidad que tiene un circuito para poder manejar circuitos iguales a el conectados a su salida.

Capacitancias parasitas. Son capacitancias que no se encuentran como el elemento *Capacitor* en un circuito, son capacitancias que se encuentran internas en un elemento, un ejemplo son las capacitancias formadas en el interior de un transistor MOS, las

capacitancias formadas entre los diferentes puntos que tiene un transistor MOS, entre la Compuerta y el Drenaje (C_{gd}), entre la Compuerta y la Fuente (C_{gs}), etc.

Teoría de Redes. Es una serie de definiciones, métodos y suposiciones usadas para analizar algunas de las propiedades de un circuito eléctrico, tanto en su comportamiento cuantitativo como cualitativo. Para el Cuantitativo la meta final es la de determinar los valores cualitativos de las variables eléctricas de un circuito y para el caso Cualitativo la meta es más sutil, esta relación es la determinación de las características cualitativas del circuito obtenidas de los valores cuantitativas de las variables eléctricas y/o de la estructura del circuito.

Definición Moderna de Circuito Eléctrico. Un circuito Eléctrico se encuentra constituido por dos conjuntos:

Circuito Eléctrico ⟹ C + P

Donde:

C involucra las relaciones de ramas y son funciones de los dispositivos que constituyen el circuito.

P involucra el patrón de interconexiones, éste puede ser representado a través de la topología. El patrón de interconexiones es importante porque en el se llevan implícitas las leyes de Kirchhoff independientemente de los componentes que la formen.

ANEXOS

ANEXO 1. Método de Newton - Raphson.

Este método es para encontrar las soluciones de una ecuación de la forma

$$f(x) = 0 \qquad (1)$$

las cuales se llaman raíces de la ecuación o ceros de la función *f*. Si *f* es una función polinomial de grado menor que cinco, esto es por que para obtener los ceros de estas ya existen fórmulas. En caso de que *f* sea de grado uno (función lineal) o dos (función cuadrática) existe la formula general. Para el caso de ser de grado tres, cuatro o cinco el método de resolución es complicado.

Los teoremas existentes para los ceros de una función polinomial de grado cinco o mayores, se acreditan a Niels Abel (1802-1829), este teorema expresa que no se puede tener una formula general en términos de un número finito de operaciones sobre los coeficientes, sin embargo para esto existen procesos numéricos para aproximar soluciones de ecuaciones.

Uno de estos procesos fue desarrollado por Sir Isaac Newton en el siglo XVIII el cual implica una aplicación a la derivada lo que se le conoce como el método de Newton y es el tema central de esta sección.

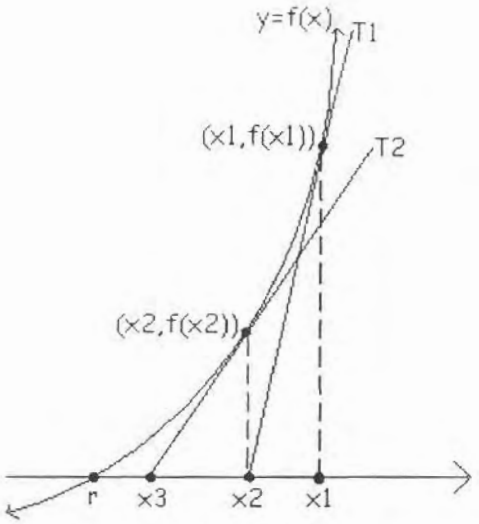


Figura A.1.1. Método de Newton

Considerando la ecuación (1), donde ***f*** es una función diferenciable, el método de Newton nos implica un procedimiento para aproximar una raíz de ***f*** esto es un numero **r** tal que $f(r) = 0$.

Dando una interpretación geométrica y tomando como base la figura (1) para explicar este método, la cual muestra la grafica de ***y=f(x)*** donde r es la intercepción x en la grafica, para poder encontrar una primera aproximación de se selecciona un número ***x1***, la elección de ***x1*** puede hacerse consultando la grafica correspondiente y como recomendación debe de ser próxima a r, como siguiente paso se considera la recta tangente a la gráfica ***f*** en el punto ***(x1, f(x1))***, la recta tangente que se representa por **T1** se muestra en la figura (1) y T1 tiene una abscisa en el origen ***x2***. El número ***x2*** sirve como una segunda aproximación de r, entonces se repite el proceso con la recta tangente T2 en el punto ***(x2, f(x2))***, la intercepción de x de T2 es ***x3*** y así se continua el proceso hasta obtener el grado de precisión requerido.

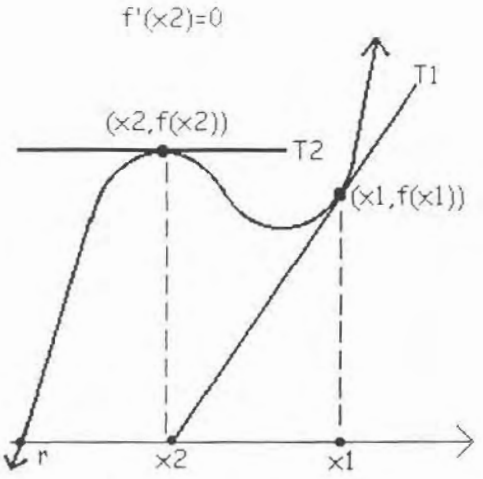


Figura A.1.2. f'(x2)=0, el método de Newton no se aplica

Para obtener las aproximaciones sucesivas **x2, x3, ,,,** a partir de la primera aproximación de x1 se usan las ecuaciones de las rectas tangentes. La recta tangente T1 en el punto **(x1, f(x1))** tiene la pendiente **f'(x),** donde la ecuación de **T1** esta dada por la ecuación (2).

$$y - f(x1) = f'(x1)(x - x1) \tag{2}$$

La intercepción de x en T1 es x2, y se determina haciendo ***x=x2*** y ***y=0*** en (2). Así obtenemos

$$0 - f(x_1) = f'(x_1)(x - x_1)$$

$$x_2 = x_1 - \frac{f(x_1)}{f(x_1)} \qquad \text{si} \qquad f'(x_1) \neq 0$$

con el valor de x2 la ecuación de T2 es

$$y - f(x_2) = f'(x_2)(x - x_2) \tag{3}$$

si en (3) hacemos $x = x3$ y $y=0$ tenemos que

$$0 - f(x_2) = f'(x_2)(x - x_2)$$

$$x_3 = x_2 - \frac{f(x_2)}{f(x_2)} \qquad \text{si} \qquad f'(x_2) \neq 0$$

si se continua de la misma forma se obtiene la fórmula general para la aproximación x_{n+1} en términos de la aproximación anterior x_n y se encuentra dada por (4).

$$x_{n+1} = x_n - \frac{f(x_n)}{f(x_n)} \quad \text{si} \qquad f'(x_n) \neq 0 \qquad (4)$$

La formula (4) se adopta fácilmente para usarse en una computadora o calculadora programable.

De la fórmula (4) se puede obtener la (n+1)-ésima aproximación, siempre que $f'(xn) \neq 0$. Cuando $f'(xn) = 0$, la reta tangente es horizontal, y en tal caso, a menos que la recta tangente sea el eje x mismo, no tiene intercepción **x**. La figura 2 muestra este suceso cuando $f'(x2) = 0$. Así, el método de Newton no se aplica si $f'(xn) = 0$, para alguna x_n. Si por ejemplo, x1 no está razonablemente cerca de r, entonces $|f'(x1)|$ puede ser pequeño de modo que la recta tangente **T1** sea casi horizontal. Entonces **x2**, abscisa en el origen de **T1**, pudiera estar todavía más lejos de *r* que ***x1***. ver la figura 3, donde se presenta esta situación.

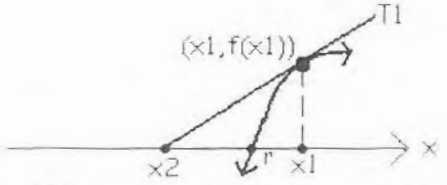


Fig A.1.3. no cenverge el Método de Newton.

ANEXO 2. Código fuente del software de la herramienta CAD desarrollada.

```
#=============================================================================
# UNIVERSIDAD TECNOLOGICA DE LA MIXTECA

# CODIGO FUENTE DE LA HERRAMIENTA DESARROLLADA
# COMO SISTEMA CAD DE ANALISIS BASICO DE CIRCUITOS
# ELECTRICOS

# AUTOR:   LANCELOT GARCIA LEYVA
# ASESOR: M. C. JOSE ANTONIO MORENO ESPINOZA
#=============================================================================


#=============================================================================
# EJECUTAR ES EL PROCEDIMIENTO PRINCIAL DEL SISTEMA
#=============================================================================

ejecutar:=proc()

 global d,L,h,t,Lista_Nodos,Valor_Comp,control,compo;


 L:='L';h:='h';t:='t';

 Elements;
 Lista_Nodos:=Elements[2];
 Valor_Comp:=Elements[3];
 control:=Elements[4];
 d:=Nodes(Lista_Nodos);
 compo:=Elements[1];
 compatibles(compo);
 tiempos();
 Valores();
 MatrizMNA();
 #evalm(MNAT);   es para ver los valores de la matriz MNA de
 #evalm(WT);
 ciclo1(MNAT,WT);
 opcion();
 captura();
end:
```

```
#=============================================================================
# procedure used for create a netlist with all circuit nodes
#=============================================================================
Nodes:= proc(Lista_Nodos)
# este procedimiento crea una lista (Lista) de todos los nodos del circuito
# examina todos los los pares de nodos de la lista Elements y si encuemtra nuevo nodo,
# lo coloca en la lista Nodes_list

local Contador, Lista;
global nu_nodos;
if nops(Lista_Nodos) > 0 then
      Lista := [Lista_Nodos[1,1],Lista_Nodos[1,2]];
  for Contador from 2 to nops(Lista_Nodos) do
    if not(member(Lista_Nodos[Contador,1],Lista)) then
      Lista := [op(Lista),Lista_Nodos[Contador,1]];
    fi;
    if not(member(Lista_Nodos[Contador,2],Lista)) then
      Lista := [op(Lista),Lista_Nodos[Contador,2]];
    fi;
  od;
else
  Lista := [];
fi;
  nu_nodos:=nops(Lista)-1;

#RETURN (Lista,nu_nodos);
end:

#=============================================================================

compatibles:=proc(compo)

global n_comp,compa,nocompa,nr,Lista_Next,n_ext,control,argu,i,
     ncond,nresis,nfv,nfc,nfvcv,nfccc,nfccv,nfvcc,no,
     xc,yc,xl,yl,kl,kc,cc,cl,tiempo,B;

kc:=0;  kl:=0;
ncond:=0; nresis:=0;
nfv:=0;   nfc:=0;
nfvcv:=0; nfccc:=0;
nfvcc:=0; nfccv:=0;
no:=0;

n_comp:=nops(compo);
compa:=0; nocompa:=0; nr:=0;
Lista_Next:=[];

for i from 1 by 1 to n_comp do
 if ( substring(compo[i],1)=K
   then  compa:=compa+1; ncond:=ncond+1;
```

```
elif ( substring(compo[i],1))=V
  then nocompa:=nocompa+1; nr:=nr+1; nfv:=nfv+1;

elif ( substring(compo[i],1))=R
  then  nocompa:=nocompa+1; nr:=nr+1; nresis:=nresis+1;


elif ( substring(compo[i],1))=C
  then  compa:=compa+1; kc:=kc+1;
  xc[kc]:=op(1,Lista_Nodos[i]); yc[kc]:=op(2,Lista_Nodos[i]);

elif ( substring(compo[i],1))=L
  then compa:=compa+1;  kl:=kl+1;
  xl[kl]:=op(1,Lista_Nodos[i]); yl[kl]:=op(2,Lista_Nodos[i]);
  B[kl]:=op(i,Elements[3]);

elif ( substring(compo[i],1))=J
  then  compa:=compa+1; nfc:=nfc+1;

elif ( substring(compo[i],1))=G
  then  compa:=compa+1; nfccv:=nfccv+1;

elif ( substring(compo[i],1))=H
  then  nocompa:=nocompa+1; nr:=nr+2; nfvcc:=nfvcc+1;

  Argumento(compo,control);
  if (member(argu,Lista_Next))
    then
      Lista_Next:=Lista_Next;
    else
      Lista_Next:=[op(Lista_Next),argu];
  fi;

elif ( substring(compo[i],1))=F
  then  nocompa:=nocompa+1; nr:=nr+1; nfccc:=nfccc+1;

  Argumento(compo,control);
  if (member(argu,Lista_Next))
    then
      Lista_Next:=Lista_Next;
    else
      Lista_Next:=[op(Lista_Next),argu];
  fi;

elif ( substring(compo[i],1))=E
  then  nocompa:=nocompa+1; nr:=nr+1; nfvcv:=nfvcv+1;

elif ( substring(compo[i],1))=O
  then nocompa:=nocompa+1; nr:=nr+1; no:=no+1;
```

```
 else  ERROR(` ESTE ELMENTO NO ES RECONOCIDO ` ,compo[i]);
 fi;
od;
 print(` Conductancias:` ,ncond);
 print(` Resistencias:` ,nresis);
 print(` Capacitancias:` ,kc);
 print(` Inductancias:` ,kl);
 print(` Fuentes de Voltaje Independientes:` ,nfv);
 print(` Fuentes de Corriente Independientes:` ,nfc);
 print(` FCCC:` ,nfccc);
 print(` FVCV:` ,nfvcv);
 print(` FCCV:` ,nfccv);
 print(` FVCC:` ,nfvcc);
 print(` Opamps:` ,no);

n_ext:=nops(Lista_Next);
cc:=Elements[5];
cl:=Elements[6];
tiempo:=Elements[7];

 #RETURN(nocompa,compa,nr,n_ext,kc,kl,xc,yc,xl,yl,cc,cl,tiempo,B);
end:

#=========================================================================
# procedimiento que inicializa valores de las variables utilizadas
#=========================================================================

Valores:=proc()

        global j,in_cap,MNAT,WT,in_ind,nu_nodos,n_ext,nr,Bandera,pos_n_ext,
         n_comp,i_ren,bandera1,renglon,corr,no_cap,seg_ren;
    local  n;

        j:=1;
     in_cap:=1;
     in_ind:=1;
      i_ren:=nu_nodos;
      with(linalg):
        n:=nu_nodos+n_ext+nr;
       MNAT:=matrix(n,n,0);
         WT:=matrix(n,1,0);
         Bandera:=vector(n_comp,0);
         pos_n_ext:=vector(n_comp,0);
    bandera1:=vector(n_comp,0);
        renglon:=vector(n_comp,0);
    seg_ren:=vector(n_comp,0);
    no_cap:=vector(n_comp,0);
    corr:=0;
end:
```

```
#=============================================================================
# Obtiene el argumento de una fuente controlada.
#=============================================================================

Argumento:=proc(compo,control)

    global argu,i;
    local  nci,k,p;

     nci:=nops(control);
     for k from 1 to nci do
       if (member(compo[i], control[k]))
         then
           argu:=op(op(2,control[k]));
       fi;
     od;
#    RETURN(argu);
end:

#=============================================================================
# Procedimiento para el stamp MNA de una Fuente de corriente controlada por
# voltaje (VCCS).
#=============================================================================

MatrizVCCS:=proc(nu_nodos,n_ext,nr)

   global MNAVCCS,MNAT,compo,i,control,Valor_Comp;
   local  nodo1,nodo2,nodo3,nodo4,n,p;

   n:=nu_nodos+n_ext+nr;
   MNAVCCS:=matrix(n,n,0);
   Argumento(compo,control);
   member(argu, compo, 'p');

   nodo1:=op(1,Lista_Nodos[p]);
   nodo2:=op(2,Lista_Nodos[p]);
   nodo3:=op(1,Lista_Nodos[i]);
   nodo4:=op(2,Lista_Nodos[i]);

if (nodo1>0 and nodo2>0 and nodo3>0 and nodo4>0) then

     MNAVCCS[nodo3,nodo1]:=op(i,Valor_Comp);
     MNAVCCS[nodo3,nodo2]:=-op(i,Valor_Comp);
     MNAVCCS[nodo4,nodo1]:=-op(i,Valor_Comp);
     MNAVCCS[nodo4,nodo2]:=op(i,Valor_Comp);

elif (nodo4=0 and nodo2=0) then
     MNAVCCS[nodo3,nodo1]:=op(i,Valor_Comp);
elif (nodo1=0 and nodo3=0) then
     MNAVCCS[nodo4,nodo2]:=op(i,Valor_Comp);
```

```
elif (nodo3=0 and nodo2=0) then
      MNAVCCS[nodo4,nodo1]:=-op(i,Valor_Comp);
elif (nodo1=0 and nodo4=0) then
      MNAVCCS[nodo3,nodo2]:=-op(i,Valor_Comp);
elif (nodo1=0) then

          MNAVCCS[nodo3,nodo2]:=-op(i,Valor_Comp);
          MNAVCCS[nodo4,nodo2]:=op(i,Valor_Comp);

elif (nodo2=0) then

          MNAVCCS[nodo3,nodo1]:=op(i,Valor_Comp);
          MNAVCCS[nodo4,nodo1]:=-op(i,Valor_Comp);

elif (nodo3=0) then

          MNAVCCS[nodo4,nodo1]:=-op(i,Valor_Comp);
          MNAVCCS[nodo4,nodo2]:=op(i,Valor_Comp);

elif (nodo4=0) then

          MNAVCCS[nodo3,nodo1]:=op(i,Valor_Comp);
          MNAVCCS[nodo3,nodo2]:=-op(i,Valor_Comp);

fi;

#RETURN(MNAVCCS);
end:

#=============================================================================
# Procedimiento para el stamp MNA de una Fuente de corriente controlada por
# corriente (CCCS).
#=============================================================================

MatrizCCCS:=proc(nu_nodos,n_ext,nr)

   global MNACCCS,MNAT,compo,i,control,Valor_Comp,i_ren,j,Bandera,pos_n_ext,renglon;
   local  nodo1,nodo2,nodo3,nodo4,n,p,pos;

   n:=nu_nodos+n_ext+nr;
   MNACCCS:=matrix(n,n,0);

   Argumento(compo,control);
   member(argu, compo, 'p');

if (Bandera[p]=0)
  then
    i_ren:=i_ren+1;
    nodo1:=i_ren;
    nodo2:=op(2,Lista_Nodos[p]);
    nodo3:=op(1,Lista_Nodos[i]);
```

```
    nodo4:=op(2,Lista_Nodos[i]);
    pos_n_ext[p]:=i_ren;
    Bandera[p]:=1;
  else
    nodo1:=pos_n_ext[p];
    nodo2:=op(2,Lista_Nodos[p]);
    nodo3:=op(1,Lista_Nodos[i]);
    nodo4:=op(2,Lista_Nodos[i]);
fi;
    pos:=nu_nodos+n_ext+j;
    renglon[i]:=pos;

if (nodo1>0 and nodo2>0 and nodo3>0 and nodo4>0) then

        MNACCCS[pos,nodo1]:=MNACCCS[pos,nodo1]+1;
    MNACCCS[pos,nodo2]:=MNACCCS[pos,nodo2]-1;
    MNACCCS[nodo1,pos]:=MNACCCS[nodo1,pos]+1;
    MNACCCS[nodo2,pos]:=MNACCCS[nodo2,pos]-1;
    MNACCCS[nodo3,pos]:=MNACCCS[nodo3,pos]+op(i,Valor_Comp);
    MNACCCS[nodo4,pos]:=MNACCCS[nodo4,pos]-op(i,Valor_Comp);

elif (nodo4=0 and nodo2=0) then

    MNACCCS[pos,nodo1]:=MNACCCS[pos,nodo1]+1;
    MNACCCS[nodo1,pos]:=MNACCCS[nodo1,pos]+1;
    MNACCCS[nodo3,pos]:=MNACCCS[nodo3,pos]+op(i,Valor_Comp);

elif (nodo3=0 and nodo2=0) then

    MNACCCS[pos,nodo1]:=MNACCCS[pos,nodo1]+1;
    MNACCCS[nodo1,pos]:=MNACCCS[nodo1,pos]+1;
    MNACCCS[nodo4,pos]:=MNACCCS[nodo4,pos]-op(i,Valor_Comp);


elif (nodo2=0) then

    MNACCCS[pos,nodo1]:=MNACCCS[pos,nodo1]+1;
    MNACCCS[nodo1,pos]:=MNACCCS[nodo1,pos]+1;
    MNACCCS[nodo3,pos]:=MNACCCS[nodo3,pos]+op(i,Valor_Comp);
    MNACCCS[nodo4,pos]:=MNACCCS[nodo4,pos]-op(i,Valor_Comp);


elif (nodo3=0) then

    MNACCCS[pos,nodo1]:=MNACCCS[pos,nodo1]+1;
    MNACCCS[pos,nodo2]:=MNACCCS[pos,nodo2]-1;
    MNACCCS[nodo1,pos]:=MNACCCS[nodo1,pos]+1;
    MNACCCS[nodo2,pos]:=MNACCCS[nodo2,pos]-1;
    MNACCCS[nodo4,pos]:=MNACCCS[nodo4,pos]-op(i,Valor_Comp);

elif (nodo4=0) then
```

```
      MNACCCS[pos,nodo1]:=MNACCCS[pos,nodo1]+1;
      MNACCCS[pos,nodo2]:=MNACCCS[pos,nodo2]-1;
      MNACCCS[nodo1,pos]:=MNACCCS[nodo1,pos]+1;
      MNACCCS[nodo2,pos]:=MNACCCS[nodo2,pos]-1;
      MNACCCS[nodo3,pos]:=MNACCCS[nodo3,pos]+op(i,Valor_Comp);

fi;
j:=j+1;

#RETURN(MNACCCS);
end:

#=============================================================================
# Procedimiento para el stamp MNA de una Fuente de Voltaje controlada por
# corriente (CCVS).
#=============================================================================

MatrizCCVS:=proc(nu_nodos,n_ext,nr)

  global MNACCVS,MNAT,compo,i,control,Valor_Comp,i_ren,j,Bandera,pos_n_ext,renglon,
       seg_ren;
  local  nodo1,nodo2,nodo3,nodo4,n,p,pos1,pos2;

  n:=nu_nodos+n_ext+nr;
  MNACCVS:=matrix(n,n,0);

  Argumento(compo,control);
  member(argu, compo, 'p');

if (Bandera[p]=0)
  then
    i_ren:=i_ren+1;
    nodo1:=i_ren;
    nodo2:=op(2,Lista_Nodos[p]);
    nodo3:=op(1,Lista_Nodos[i]);
    nodo4:=op(2,Lista_Nodos[i]);
    pos_n_ext[p]:=i_ren;
    Bandera[p]:=1;
  else
    nodo1:=pos_n_ext[p];
    nodo2:=op(2,Lista_Nodos[p]);
    nodo3:=op(1,Lista_Nodos[i]);
    nodo4:=op(2,Lista_Nodos[i]);
fi;
    pos1:=nu_nodos+n_ext+j;
    pos2:=nu_nodos+n_ext+j+1;
    renglon[i]:=pos1;
    seg_ren[i]:=pos2;

if (nodo1>0 and nodo2>0 and nodo3>0 and nodo4>0) then
```

```
        MNACCVS[pos1,nodo1]:= 1;
        MNACCVS[pos1,nodo2]:=-1;
        MNACCVS[pos2,nodo3]:= 1;
        MNACCVS[pos2,nodo4]:=-1;
        MNACCVS[nodo1,pos1]:= 1;
        MNACCVS[nodo2,pos1]:=-1;
        MNACCVS[nodo3,pos2]:= 1;
        MNACCVS[nodo4,pos2]:=-1;
        MNACCVS[pos2,pos1]:=-op(i,Valor_Comp);

elif (nodo4=0 and nodo2=0) then
        MNACCVS[pos1,nodo1]:= 1;
        MNACCVS[pos2,nodo3]:= 1;
        MNACCVS[nodo1,pos1]:= 1;
        MNACCVS[nodo3,pos2]:= 1;
        MNACCVS[pos2,pos1]:=-op(i,Valor_Comp);

elif (nodo3=0 and nodo2=0) then
        MNACCVS[pos1,nodo1]:= 1;
        MNACCVS[pos2,nodo4]:=-1;
        MNACCVS[nodo1,pos1]:= 1;
        MNACCVS[nodo4,pos2]:=-1;
        MNACCVS[pos2,pos1]:=-op(i,Valor_Comp);


elif (nodo2=0) then
        MNACCVS[pos1,nodo1]:= 1;
        MNACCVS[pos2,nodo3]:= 1;
        MNACCVS[pos2,nodo4]:=-1;
        MNACCVS[nodo1,pos1]:= 1;
        MNACCVS[nodo3,pos2]:= 1;
        MNACCVS[nodo4,pos2]:=-1;
        MNACCVS[pos2,pos1]:=-op(i,Valor_Comp);

elif (nodo3=0) then
        MNACCVS[pos1,nodo1]:= 1;
        MNACCVS[pos1,nodo2]:=-1;
        MNACCVS[pos2,nodo4]:=-1;
        MNACCVS[nodo1,pos1]:= 1;
        MNACCVS[nodo2,pos1]:=-1;
        MNACCVS[nodo4,pos2]:=-1;
        MNACCVS[pos2,pos1]:=-op(i,Valor_Comp);



elif (nodo4=0) then
        MNACCVS[pos1,nodo1]:= 1;
        MNACCVS[pos1,nodo2]:=-1;
        MNACCVS[pos2,nodo3]:= 1;
        MNACCVS[nodo1,pos1]:= 1;
        MNACCVS[nodo2,pos1]:=-1;
```

```
    MNACCVS[nodo3,pos2]:= 1;
    MNACCVS[pos2,pos1]:=-op(i,Valor_Comp);
fi;
j:=j+2;

#RETURN(MNACCVS);
end:



#=========================================================================
# Procedimiento para el stamp MNA de una Fuente de Voltaje Controlada por
# Voltaje (VCVS).
#=========================================================================


MatrizVCVS:=proc(Lista_Nodos,nu_nodos,n_ext,nr)

  global MNAVCVS,MNAT,compo,j,control,Valor_Comp,j,renglon;
  local  nodo1,nodo2,nodo3,nodo4,n,p,pos;

  n:=nu_nodos+n_ext+nr;
  MNAVCVS:=matrix(n,n,0);

  Argumento(compo,control);

  member(argu, compo, 'p');
  pos:=nu_nodos+n_ext+j;
  renglon[i]:=pos;

  nodo1:=op(1,Lista_Nodos[p]);
  nodo2:=op(2,Lista_Nodos[p]);
  nodo3:=op(1,Lista_Nodos[i]);
  nodo4:=op(2,Lista_Nodos[i]);

  if (nodo1>0 and nodo2>0 and nodo3>0 and nodo4>0)
    then
        MNAVCVS[pos,nodo1]:=MNAVCVS[pos,nodo1]-op(i,Valor_Comp);
      MNAVCVS[pos,nodo2]:=MNAVCVS[pos,nodo2]+op(i,Valor_Comp);
      MNAVCVS[pos,nodo3]:=MNAVCVS[pos,nodo3]+1;
      MNAVCVS[pos,nodo4]:=MNAVCVS[pos,nodo4]-1;
      MNAVCVS[nodo3,pos]:=MNAVCVS[nodo3,pos]+1;
      MNAVCVS[nodo4,pos]:=MNAVCVS[nodo4,pos]-1;

    elif (nodo4=0 and nodo2=0)
      then
        MNAVCVS[pos,nodo1]:=MNAVCVS[pos,nodo1]-op(i,Valor_Comp);
          MNAVCVS[pos,nodo3]:=MNAVCVS[pos,nodo3]+1;
        MNAVCVS[nodo3,pos]:=MNAVCVS[nodo3,pos]+1;

    elif (nodo1=0 and nodo3=0)
```

```
        then
          MNAVCVS[pos,nodo2]:=MNAVCVS[pos,nodo2]+op(i,Valor_Comp);
          MNAVCVS[pos,nodo4]:=MNAVCVS[pos,nodo4]-1;
          MNAVCVS[nodo4,pos]:=MNAVCVS[nodo4,pos]-1;

      elif (nodo3=0 and nodo2=0)
        then
            MNAVCVS[pos,nodo1]:=MNAVCVS[pos,nodo1]-op(i,Valor_Comp);
          MNAVCVS[pos,nodo4]:=MNAVCVS[pos,nodo4]-1;
          MNAVCVS[nodo4,pos]:=MNAVCVS[nodo4,pos]-1;

      elif (nodo1=0 and nodo4=0)
        then
          MNAVCVS[pos,nodo2]:=MNAVCVS[pos,nodo2]+op(i,Valor_Comp);
            MNAVCVS[pos,nodo3]:=MNAVCVS[pos,nodo3]+1;
          MNAVCVS[nodo3,pos]:=MNAVCVS[nodo3,pos]+1;

      elif (nodo1=0)
        then
          MNAVCVS[nodo3,pos]:=MNAVCVS[nodo3,pos]+op(i,Valor_Comp);
          MNAVCVS[pos,nodo3]:=MNAVCVS[pos,nodo3]+1;
          MNAVCVS[pos,nodo4]:=MNAVCVS[pos,nodo4]-1;
          MNAVCVS[nodo3,pos]:=MNAVCVS[nodo3,pos]+1;
          MNAVCVS[nodo4,pos]:=MNAVCVS[nodo4,pos]-1;

      elif (nodo2=0)
        then
            MNAVCVS[pos,nodo1]:=MNAVCVS[pos,nodo1]-op(i,Valor_Comp);
            MNAVCVS[pos,nodo3]:=MNAVCVS[pos,nodo3]+1;
          MNAVCVS[pos,nodo4]:=MNAVCVS[pos,nodo4]-1;
          MNAVCVS[nodo3,pos]:=MNAVCVS[nodo3,pos]+1;
          MNAVCVS[nodo4,pos]:=MNAVCVS[nodo4,pos]-1;

      elif (nodo3=0)
        then
            MNAVCVS[pos,nodo1]:=MNAVCVS[pos,nodo1]-op(i,Valor_Comp);
          MNAVCVS[pos,nodo2]:=MNAVCVS[pos,nodo2]+op(i,Valor_Comp);
          MNAVCVS[pos,nodo4]:=MNAVCVS[pos,nodo4]-1;
          MNAVCVS[nodo4,pos]:=MNAVCVS[nodo4,pos]-1;

      elif (nodo4=0)
        then
          MNAVCVS[pos,nodo1]:=MNAVCVS[pos,nodo1]-op(i,Valor_Comp);
          MNAVCVS[pos,nodo2]:=MNAVCVS[pos,nodo2]+op(i,Valor_Comp);
            MNAVCVS[pos,nodo3]:=MNAVCVS[pos,nodo3]+1;
          MNAVCVS[nodo3,pos]:=MNAVCVS[nodo3,pos]+1;
fi;
j:=j+1;
#RETURN(MNAVCVS);
end:
```

```
#=================================================================
# Procedimiento que llena la matriz MNAT
#=================================================================

 Llenar:=proc()

        global i,n_comp,Lista_Nodos,nu_nodos,n_ext,nr,j,MNAT,WT,Valor_Comp;


for i from 1 by 1 to n_comp do
if ( substring(compo[i],1))=K
  then
     MatrizK(Lista_Nodos,nu_nodos,n_ext,nr);
     MNAT:=matadd(MNAT,MNAK);
fi;

if ( substring(compo[i],1))=V
  then
     MatrizFVI(Lista_Nodos,nu_nodos,n_ext,nr);
     MNAT:=matadd(MNAT,MNAFVI);
     WT:=matadd(WT,WFVI);

fi;

if ( substring(compo[i],1))=R
  then
     MatrizR(Lista_Nodos,nu_nodos,n_ext,nr);
     MNAT:=matadd(MNAT,MNAR);
fi;

if ( substring(compo[i],1))=C
  then
     MatrizC(Lista_Nodos,nu_nodos,n_ext,nr);
     MNAT:=matadd(MNAT,MNAC);
     WT:=matadd(WT,WTC);


fi;

if ( substring(compo[i],1))=L
  then
        MatrizL(Lista_Nodos,nu_nodos,n_ext,nr);
     MNAT:=matadd(MNAT,MNAL);
     WT:=matadd(WT,WTL);

fi;

if ( substring(compo[i],1))=J
  then
     MatrizFCI(Lista_Nodos,nu_nodos,n_ext,nr);
     WT:=matadd(WT,WFCI);
```

```
fi;

if ( substring(compo[i],1))=G
  then
    MatrizVCCS(nu_nodos,n_ext,nr);
    MNAT:=matadd(MNAT,MNAVCCS);
fi;

if ( substring(compo[i],1))=H
  then
    MatrizCCVS(nu_nodos,n_ext,nr);
    MNAT:=matadd(MNAT,MNACCVS);
fi;

if ( substring(compo[i],1))=F
  then
    MatrizCCCS(nu_nodos,n_ext,nr);
    MNAT:=matadd(MNAT,MNACCCS);
fi;

if ( substring(compo[i],1))=E
  then
    MatrizVCVS(Lista_Nodos,nu_nodos,n_ext,nr);
    MNAT:=matadd(MNAT,MNAVCVS);
fi;

if ( substring(compo[i],1))=O
  then
    MatrizOP(nu_nodos,n_ext,nr);
    MNAT:=matadd(MNAT,MNAOP);
fi;

od;
#RETURN(MNAT,WT);
end:


#=====================================================================
# Procedimiento que genera la martiz MNA final
#=====================================================================


MatrizMNA:=proc()

Llenar();
Llenar1();

#RETURN(MNAT,WT);

end:
```

```
#=============================================================================
# Procedimiento para  el stamp MNA de un Capacitor.
#=============================================================================

 MatrizC:=proc(Lista_Nodos,nu_nodos,n_ext,nr)

 global MNAC,WTC,i,Valor_Comp,h,Vc,in_cap,no_cap;
 local  nodo1,nodo2,n;

 n:=nu_nodos+n_ext+nr;
 MNAC:=matrix(n,n,0);
 WTC:=matrix(n,1,0);

 Vc:='Vc';
#  h:='h';

 if corr=0  then
  no_cap[i]:=in_cap;
 else
  in_cap:=no_cap[i];
 fi;

 nodo1:=op(1,Lista_Nodos[i]);
 nodo2:=op(2,Lista_Nodos[i]);

 if (nodo1>0 and nodo2>0)
   then
      MNAC[nodo1,nodo1]:=op(i,Valor_Comp)/h;
      MNAC[nodo2,nodo2]:=op(i,Valor_Comp)/h;
      MNAC[nodo1,nodo2]:=-op(i,Valor_Comp)/h;
      MNAC[nodo2,nodo1]:=-op(i,Valor_Comp)/h;

      WTC[nodo1,1]:= (op(i,Valor_Comp)/h)*Vc[in_cap];
      WTC[nodo2,1]:=-(op(i,Valor_Comp)/h)*Vc[in_cap];


   else if (nodo2=0)
       then
          MNAC[nodo1,nodo1]:=op(i,Valor_Comp)/h;
          WTC[nodo1,1]:= (op(i,Valor_Comp)/h)*Vc[in_cap];
       else
          MNAC[nodo2,nodo2]:=op(i,Valor_Comp)/h;
          WTC[nodo2,1]:=-(op(i,Valor_Comp)/h)*Vc[in_cap];
       fi;
 fi;
 if corr=0 then
  in_cap:=in_cap+1;
 fi;
#  RETURN(MNAC,WTC);
 end:
```

```
#=============================================================================
# Procedimiento para  el stamp MNA de un Inductor.
#=============================================================================

 MatrizL:=proc(Lista_Nodos,nu_nodos,n_ext,nr)

 global MNAL,WTL,i,Valor_Comp,h,Jl,in_ind,no_ind;
 local  nodo1,nodo2,n;

 n:=nu_nodos+n_ext+nr;
 MNAL:=matrix(n,n,0);
 WTL:=matrix(n,1,0);

 Jl:='Jl';
#  h:='h';

if corr=0  then
  no_ind[i]:=in_ind;
 else
  in_ind:=no_ind[i];
fi;


 nodo1:=op(1,Lista_Nodos[i]);
 nodo2:=op(2,Lista_Nodos[i]);

 if (nodo1>0 and nodo2>0)
   then
      MNAL[nodo1,nodo1]:= h/op(i,Valor_Comp);
      MNAL[nodo2,nodo2]:= h/op(i,Valor_Comp);
      MNAL[nodo1,nodo2]:=-h/op(i,Valor_Comp);
      MNAL[nodo2,nodo1]:=-h/op(i,Valor_Comp);

      WTL[nodo1,1]:=Jl[in_ind];
      WTL[nodo2,1]:=-Jl[in_ind];


   else if (nodo2=0)
       then
          MNAL[nodo1,nodo1]:=h/op(i,Valor_Comp);
          WTL[nodo1,1]:= Jl[in_ind];
       else
          MNAL[nodo2,nodo2]:=h/op(i,Valor_Comp);
          WTL[nodo2,1]:=-Jl[in_ind];
       fi;
 fi;

 if corr=0 then
 in_ind:=in_ind+1;
 fi;
```

```
# RETURN(MNAL,WTL);
 end:
#=============================================================================
# Procedimiento para el stamp MNA de un Amplificador Operacional Ideal.
#=============================================================================

MatrizOP:=proc(nu_nodos,n_ext,nr)

 global MNAOP,i,Lista_Nodos,j,renglon;
 local  n,nodo1,nodo2,nodo3,pos;

 n:=nu_nodos+n_ext+nr;
 MNAOP:=matrix(n,n,0);

 nodo1:=op(1,Lista_Nodos[i]);
 nodo2:=op(2,Lista_Nodos[i]);
 nodo3:=op(3,Lista_Nodos[i]);

 pos:=nu_nodos+n_ext+j;
 renglon[i]:=pos;

 if (nodo1>0 and nodo2>0 and nodo3>0)
   then
        MNAOP[pos,nodo1]:= 1;
     MNAOP[pos,nodo2]:=-1;
     MNAOP[nodo3,pos]:= 1;

   elif (nodo1=0)
      then
        MNAOP[pos,nodo2]:=-1;
        MNAOP[nodo3,pos]:= 1;

   elif (nodo2=0)
      then
        MNAOP[pos,nodo1]:= 1;
        MNAOP[nodo3,pos]:= 1;
   fi;
 j:=j+1;
# RETURN(MNAOP);
 end:

#=============================================================================
# Procedimiento para  el stamp MNA de una conductancia
#=============================================================================

 MatrizK:=proc(Lista_Nodos,nu_nodos,n_ext,nr)

 global MNAK,i,Valor_Comp;
 local  nodo1,nodo2,n;
```

```
n:=nu_nodos+n_ext+nr;
MNAK:=matrix(n,n,0);

nodo1:=op(1,Lista_Nodos[i]);
nodo2:=op(2,Lista_Nodos[i]);

if (nodo1>0 and nodo2>0)
  then
     MNAK[nodo1,nodo1]:=op(i,Valor_Comp);
     MNAK[nodo2,nodo2]:=op(i,Valor_Comp);
     MNAK[nodo1,nodo2]:=-op(i,Valor_Comp);
     MNAK[nodo2,nodo1]:=-op(i,Valor_Comp);
  else if (nodo2=0)
       then
          MNAK[nodo1,nodo1]:=op(i,Valor_Comp);
       else
          MNAK[nodo2,nodo2]:=op(i,Valor_Comp);
       fi;
fi;
RETURN(MNAK);
end:

#=============================================================================
# Procedimiento para el stamp MNA de una resistencia
#=============================================================================

MatrizR:=proc(Lista_Nodos,nu_nodos,n_ext,nr)

global MNAR,i,Valor_Comp,j,renglon,corr;
local  nodo1,nodo2,n,pos;

n:=nu_nodos+n_ext+nr;
MNAR:=matrix(n,n,0);

nodo1:=op(1,Lista_Nodos[i]);
nodo2:=op(2,Lista_Nodos[i]);

if corr = 0 then
  pos:=nu_nodos+n_ext+j;
  renglon[i]:=pos;
    else
     pos:=renglon[i];
fi;

if (nodo1>0 and nodo2>0)
  then
     MNAR[nodo1,pos]:= 1;
     MNAR[nodo2,pos]:=-1;
     MNAR[pos,nodo1]:= 1;
     MNAR[pos,nodo2]:=-1;
     MNAR[pos,pos]:=-op(i,Valor_Comp);
```

```
   else if (nodo2=0)
        then
          MNAR[nodo1,pos]:=1;
          MNAR[pos,nodo1]:=1;
          MNAR[pos,pos]:=-op(i,Valor_Comp);
        else
          MNAR[nodo2,pos]:=-1;
          MNAR[pos,nodo2]:=-1;
          MNAR[pos,pos]:=-op(i,Valor_Comp);
        fi;
 fi;
 if corr=0 then
 j:=j+1;
 fi;
# RETURN(MNAR);
 end:
#=============================================================================
# Procedimiento para el stamp MNA de una Fuente de Voltaje Independiente
#=============================================================================

 MatrizFVI:=proc(Lista_Nodos,nu_nodos,n_ext,nr)

 global MNAFVI,WFVI,i,Valor_Comp,j,renglon;
 local  nodo1,nodo2,n,pos;

 n:=nu_nodos+n_ext+nr;
 MNAFVI:=matrix(n,n,0);
 WFVI:=matrix(n,1,0);

 nodo1:=op(1,Lista_Nodos[i]);
 nodo2:=op(2,Lista_Nodos[i]);
 pos:=nu_nodos+n_ext+j;
 renglon[i]:=pos;

 if (nodo1>0 and nodo2>0)
   then
     MNAFVI[nodo1,pos]:= 1;
     MNAFVI[nodo2,pos]:=-1;
     MNAFVI[pos,nodo1]:= 1;
     MNAFVI[pos,nodo2]:=-1;
     WFVI[pos,1]:=op(i,Valor_Comp);
   else if (nodo2=0)
        then
          MNAFVI[nodo1,pos]:=1;
          MNAFVI[pos,nodo1]:=1;
          WFVI[pos,1]:=op(i,Valor_Comp);
        else
          MNAFVI[nodo2,pos]:=-1;
          MNAFVI[pos,nodo2]:=-1;
          WFVI[pos,1]:=op(i,Valor_Comp);
        fi;
```

```
 fi;
 j:=j+1;
#  RETURN(MNAFVI,WFVI);
 end:
#=============================================================================
# Procedimiento para el stamp MNA de una Fuente de Corriente Independiente
#=============================================================================

 MatrizFCI:=proc(Lista_Nodos,nu_nodos,n_ext,nr)

 global WFCI,i,Valor_Comp;
 local  nodo1,nodo2,n;

 n:=nu_nodos+n_ext+nr;
 WFCI:=matrix(n,1,0);

 nodo1:=op(1,Lista_Nodos[i]);
 nodo2:=op(2,Lista_Nodos[i]);

 if (nodo1>0 and nodo2>0)
   then
      WFCI[nodo1,1]:=-op(i,Valor_Comp);
      WFCI[nodo2,1]:= op(i,Valor_Comp);
   else if (nodo2=0)
       then
          WFCI[nodo1,1]:=-op(i,Valor_Comp);
       else
          WFCI[nodo2,1]:= op(i,Valor_Comp);
       fi;
 fi;
#  RETURN(WFCI);
 end:


#=============================================================================
# Procedimiento para  el stamp MNA de una conductancia corregida la posicion
#=============================================================================

 MatrizcorrK:=proc(Lista_Nodos,nu_nodos,n_ext,nr)

 global MNACK,i,Valor_Comp,nnuevo_nodo;
 local  nodo1,nodo2,n;

 n:=nu_nodos+n_ext+nr;
 MNACK:=matrix(n,n,0);

 nodo1:=op(1,Lista_Nodos[i]);
 nodo2:=nuevo_nodo;

 if (nodo1>0 and nodo2>0)
   then
```

```
      MNACK[nodo1,nodo1]:=op(i,Valor_Comp);
      MNACK[nodo2,nodo2]:=op(i,Valor_Comp);
      MNACK[nodo1,nodo2]:=-op(i,Valor_Comp);
      MNACK[nodo2,nodo1]:=-op(i,Valor_Comp);
   else if (nodo2=0)
       then
         MNACK[nodo1,nodo1]:=op(i,Valor_Comp);
       else
         MNACK[nodo2,nodo2]:=op(i,Valor_Comp);
       fi;
 fi;
# RETURN(MNACK);
 end:

#=============================================================================
# Procedimiento para el stamp MNA de una resistencia corregida la posicion
#=============================================================================

 MatrizcorrR:=proc(Lista_Nodos,nu_nodos,n_ext,nr)

 global MNACR,i,Valor_Comp,j,nuevo_nodo,renglon;
 local  nodo1,nodo2,n,pos;

 n:=nu_nodos+n_ext+nr;
 MNACR:=matrix(n,n,0);

 nodo1:=op(1,Lista_Nodos[i]);
 nodo2:=nuevo_nodo;
 pos:=renglon[i];

 if (nodo1>0 and nodo2>0)
   then
      MNACR[nodo1,pos]:= 1;
      MNACR[nodo2,pos]:=-1;
      MNACR[pos,nodo1]:= 1;
      MNACR[pos,nodo2]:=-1;
      MNACR[pos,pos]:=-op(i,Valor_Comp);
   else if (nodo2=0)
       then
         MNACR[nodo1,pos]:=1;
         MNACR[pos,nodo1]:=1;
         MNACR[pos,pos]:=-op(i,Valor_Comp);
       else
         MNACR[nodo2,pos]:=-1;
         MNACR[pos,nodo2]:=-1;
         MNACR[pos,pos]:=-op(i,Valor_Comp);
       fi;
 fi;

# RETURN(MNACR);
 end:
```

```
#=============================================================================
# Procedimiento para  el stamp MNA de un Capacitor corregida la posicion
#=============================================================================

 MatrizcorrC:=proc(Lista_Nodos,nu_nodos,n_ext,nr)

 global MNACC,WTCC,i,Valor_Comp,h,Vc,in_cap,nuevo_nodo,no_cap;
 local  nodo1,nodo2,n;

 n:=nu_nodos+n_ext+nr;
 MNACC:=matrix(n,n,0);
 WTCC:=matrix(n,1,0);

 Vc:='Vc';
 h:='h';
 in_cap:=no_cap[i];
 nodo1:=op(1,Lista_Nodos[i]);
 nodo2:=nuevo_nodo;

 if (nodo1>0 and nodo2>0)
   then
      MNACC[nodo1,nodo1]:=op(i,Valor_Comp)/h;
      MNACC[nodo2,nodo2]:=op(i,Valor_Comp)/h;
      MNACC[nodo1,nodo2]:=-op(i,Valor_Comp)/h;
      MNACC[nodo2,nodo1]:=-op(i,Valor_Comp)/h;

      WTCC[nodo1,1]:= (op(i,Valor_Comp)/h)*Vc[in_cap];
      WTCC[nodo2,1]:=-(op(i,Valor_Comp)/h)*Vc[in_cap];


   else if (nodo2=0)
       then
          MNACC[nodo1,nodo1]:=op(i,Valor_Comp)/h;
          WTCC[nodo1,1]:= (op(i,Valor_Comp)/h)*Vc[in_cap];
       else
          MNACC[nodo2,nodo2]:=op(i,Valor_Comp)/h;
          WTCC[nodo2,1]:=-(op(i,Valor_Comp)/h)*Vc[in_cap];
       fi;
 fi;
#  RETURN(MNACC,WTCC);
 end:

#=============================================================================
# Procedimiento para  el stamp MNA de un Inductor corregida la posicion
#=============================================================================

 MatrizcorrL:=proc(Lista_Nodos,nu_nodos,n_ext,nr)

 global MNACL,WTCL,i,Valor_Comp,h,Jl,in_ind,no_ind;
 local  nodo1,nodo2,n;
```

```
 n:=nu_nodos+n_ext+nr;
 MNACL:=matrix(n,n,0);
 WTCL:=matrix(n,1,0);

 Jl:='Jl';
 h:='h';
 nodo1:=op(1,Lista_Nodos[i]);
 nodo2:=nuevo_nodo;

 in_ind:=no_ind[i];

 if (nodo1>0 and nodo2>0)
   then
     MNACL[nodo1,nodo1]:= h/op(i,Valor_Comp);
     MNACL[nodo2,nodo2]:= h/op(i,Valor_Comp);
     MNACL[nodo1,nodo2]:=-h/op(i,Valor_Comp);
     MNACL[nodo2,nodo1]:=-h/op(i,Valor_Comp);

     WTCL[nodo1,1]:=Jl[in_ind];
     WTCL[nodo2,1]:=-Jl[in_ind];

   else if (nodo2=0)
     then
       MNACL[nodo1,nodo1]:=h/op(i,Valor_Comp);
       WTCL[nodo1,1]:= Jl[in_ind];
     else
       MNACL[nodo2,nodo2]:=h/op(i,Valor_Comp);
       WTCL[nodo2,1]:=-Jl[in_ind];
     fi;
 fi;
# RETURN(MNACL,WTCL);
 end:


#=============================================================
# Procedimiento que genera la matriz MNA corregidas las posiciones de nodos
#=============================================================

 Llenar1:=proc()

    local conta;

       global i,n_comp,Lista_Nodos,nu_nodos,n_ext,nr,j,MNAT,WT,Valor_Comp,
        argu,p,bandera1,nuevo_nodo,pos_n_ext_,corr;

corr:=1;

for conta from 1 by 1 to n_comp do
 if ( substring(compo[conta],1))=H
   then
     i:=conta;
```

```
Argumento(compo,control);
member(argu, compo, 'p');

  if bandera1[p]=0 then

          bandera1[p]=1;
      if ( substring(argu,1))=R
          then
            i:=p;
            MatrizR(Lista_Nodos,nu_nodos,n_ext,nr);
            MNAT:=matadd(MNAT,-MNAR);
            nuevo_nodo:=pos_n_ext[p];
                    MatrizcorrR(Lista_Nodos,nu_nodos,n_ext,nr);
               MNAT:=matadd(MNAT,MNACR);
       fi;
      if ( substring(argu,1))=K
          then
            i:=p;
            MatrizK(Lista_Nodos,nu_nodos,n_ext,nr);
            MNAT:=matadd(MNAT,-MNAK);
            nuevo_nodo:=pos_n_ext[p];
                    MatrizcorrK(Lista_Nodos,nu_nodos,n_ext,nr);
               MNAT:=matadd(MNAT,MNACK);
      fi;

          if ( substring(argu,1))=C
          then
            i:=p;
            MatrizC(Lista_Nodos,nu_nodos,n_ext,nr);
            MNAT:=matadd(MNAT,-MNAC);
                    WT:=matadd(WT,-WTC);
            nuevo_nodo:=pos_n_ext[p];
                    MatrizcorrC(Lista_Nodos,nu_nodos,n_ext,nr);
               MNAT:=matadd(MNAT,MNACC);
                    WT:=matadd(WT,WTCC);
      fi;

      if ( substring(argu,1))=L
          then
            i:=p;
            MatrizL(Lista_Nodos,nu_nodos,n_ext,nr);
            MNAT:=matadd(MNAT,-MNAL);
                    WT:=matadd(WT,-WTL);
            nuevo_nodo:=pos_n_ext[p];
                    MatrizcorrL(Lista_Nodos,nu_nodos,n_ext,nr);
               MNAT:=matadd(MNAT,MNACL);
            WT:=matadd(WT,WTCL);
      fi;

    fi;
fi;
```

```
if ( substring(compo[conta],1))=F
  then
    i:=conta;
    Argumento(compo,control);
    member(argu, compo, 'p');

      if bandera1[p]=0 then

            bandera1[p]=1;
        if ( substring(argu,1))=R
            then
              i:=p;
              MatrizR(Lista_Nodos,nu_nodos,n_ext,nr);
              MNAT:=matadd(MNAT,-MNAR);
              nuevo_nodo:=pos_n_ext[p];
                      MatrizcorrR(Lista_Nodos,nu_nodos,n_ext,nr);
                 MNAT:=matadd(MNAT,MNACR);
         fi;
        if ( substring(argu,1))=K
            then
              i:=p;
              MatrizK(Lista_Nodos,nu_nodos,n_ext,nr);
              MNAT:=matadd(MNAT,-MNAK);
              nuevo_nodo:=pos_n_ext[p];
                      MatrizcorrK(Lista_Nodos,nu_nodos,n_ext,nr);
                 MNAT:=matadd(MNAT,MNACK);
        fi;

        if ( substring(argu,1))=C
            then
              i:=p;
              MatrizC(Lista_Nodos,nu_nodos,n_ext,nr);
              MNAT:=matadd(MNAT,-MNAC);
                      WT:=matadd(WT,-WTC);
              nuevo_nodo:=pos_n_ext[p];
                      MatrizcorrC(Lista_Nodos,nu_nodos,n_ext,nr);
                 MNAT:=matadd(MNAT,MNACC);
                      WT:=matadd(WT,WTCC);
        fi;

        if ( substring(argu,1))=L
            then
              i:=p;
              MatrizL(Lista_Nodos,nu_nodos,n_ext,nr);
              MNAT:=matadd(MNAT,-MNAL);
                      WT:=matadd(WT,-WTL);
              nuevo_nodo:=pos_n_ext[p];
                      MatrizcorrL(Lista_Nodos,nu_nodos,n_ext,nr);
                 MNAT:=matadd(MNAT,MNACL);
              WT:=matadd(WT,WTCL);
```

```
          fi;       fi;     fi;
od;
#RETURN(MNAT,WT);
end:


#=============================================================================
#Procedimiento que resuelve la matriz MNA recursivamente
#=============================================================================


ciclo1:=proc(MNAT,WT)

global M3,Jl,Vc,cl,cc,nc,nl,xl,yl,xc,yc,h,pasos,nu_nodos,n_ext,nr,B,tiempo,t;
local n,cont,cont1,cont2,M,M2,M1,jant;
nl:=kl;
nc:=kc;
h:=tiempo[3];
t:=tiempo[1];
n:=nu_nodos+n_ext+nr;
with (linalg):
for cont2 from 1 to nl by 1 do
Jl[cont2]:=cl[cont2];
od;
for cont2 from 1 to nc by 1 do
Vc[cont2]:=cc[cont2];
od;
M2:=matrix(n,1,[0]);
for cont2 from 1 to pasos by 1 do
t:=t+h;
M:=linsolve (MNAT,WT);
 if nl>0 then
  for cont from 1 to nl by 1 do
   jant[cont]:=Jl[cont];
    if xl[cont]>0 and yl[cont]>0 then
    Jl[cont]:=jant[cont]-((h/B[cont])*(M[xl[cont],1]-M[yl[cont],1]));
    else
     if xl[cont]=0 then
      Jl[cont]:=jant[cont]-((h/B[cont])*(0-M[yl[cont],1]));
      else
          Jl[cont]:=jant[cont]-((h/B[cont])*(M[xl[cont],1]-0));
      fi;
    fi;
od;
 else
  fi;
 if nc>0 then
  for cont1 from 1 to nc by 1 do
   if xc[cont1]>0 and yc[cont1]>0 then
    Vc[cont1]:=M[xc[cont1],1]-M[yc[cont1],1];
    else
```

```
      if xc[cont1]=0 then
      Vc[cont1]:=0-M[yc[cont1],1];
       else
            Vc[cont1]:=M[xc[cont1],1]-0;
      fi;
    fi;
  od;
 else
 fi;

M1:=augment(M2,M);
M2:=M1;
od;
M3:=delcols(M1,1..1);
RETURN (M3);
end:


#=====================================================================
#Procedimiento que calcula el numero de pasos de acuerdo al tiempo
#=====================================================================


tiempos:=proc()
global tiempo,h,pasos;
local resta;
h:=tiempo[3];
resta:=tiempo[2]-tiempo[1];
pasos:=resta/h;
pasos:=trunc(pasos);
#RETURN(h,pasos);
end:


#================================================
#Procedimiento de selección de la opción a graficar
#================================================


opcion:=proc()

printf ("  elije la opcion deseada tecleando 1 o 2  \n");
printf ("  1  Para graficar un voltaje nodal \n");
printf ("  2  Para graficar corriente en un elemento MNA no compatible\n");
printf (" el valor debe estar en linea de codigo  \n");

RETURN();
 end:
```

```
#=======================================================
#Procedimiento para captura de la primera opcion
#=======================================================
captura:=proc()

global res;

res:=scanf(%d);
end:

#=======================================================
#Procedimiento de segundo menu
#=======================================================

opcion1:=proc()

global res;

  if  res[1]=1
    then
      printf ("TECLEA EL NUMERO DE NODO DESEADO \n");
  elif res[1]=2
    then
      printf ("TECLEA EL ELEMENTO MNA NO COMPATIBLE DEL QUE DESEAS LA CORRIENTE \n");
  else
      printf ("Opción no valida \n");
  fi;

#RETURN();
capturas();
 end:

#=======================================================
#Procedimiento ue captura el nodo o elemento a graficar
#=======================================================
capturas:=proc()

global resp,res;

   if res[1]=1 then
    resp:=scanf(%d);
   elif res[1]=2 then
    resp:=scanf(%s);
   else
   ("Ejecuta desde resolver y elige una opción valida");
   fi;
end:
```

```
#=====================================================================
# Procedimiento para graficar
#=====================================================================


graficar:=proc()

global res,resp,M3,P,nu_nudos,renglon,p,p_car,seg_ren,
      Paux;
local ele,linea,seg_lin;

linea:=0;


if res[1]=1 then
  if resp[1]>nu_nodos then
     ("El Nodo no existe");
    else
     linea:=resp[1];
    if linea = 0 then
      (" Es el nodo de referencia, siempre es cero el voltaje")
     else
        P:=row(M3,linea);
        mataux();
        eval(P1);
        grafi();
        plot(P3);
    fi;
 fi;

elif res[1]=2 then

   ele:=convert(resp[1],symbol);
   p_car:=substring(ele,1);
    if member (ele, compo, 'p') then
    p:=p;
    else
    p:=0;
    fi;

 if p>0 then

   if p_car=H then

     linea:=renglon[p];
     P:=row(M3,linea);
     mataux();
     eval(P1);
     grafi();
```

```
      seg_lin:=seg_ren[p];
      Paux:=row(M3,seg_lin);
      mataux();
      eval(P1);
      grafiaux();
      plot([P3,P3aux]);

    else

      linea:=renglon[p];
      if linea=0 then
        ("Este elemento es MNA compatible")
       else


        P:=row(M3,linea);
        mataux();
        eval(P1);
        grafi();
        plot(P3);
       fi;
    fi;
  else ("No existe el elemento");
  fi;
else
   printf("Opción no valida: ejecuta desde opcion1()");

fi;

end:

#============================================================
#Procedimiento que obtiene el vector a graficar
#============================================================

grafiaux:=proc()
global pasos,Paux,P1,P3aux;
local cont;
P3aux:=[];
for cont from 1 to pasos by 1 do
P3aux:=[op(P3aux),[P1[cont],Paux[cont]]];
od;
#RETURN (P3aux);
end:

#============================================================
#Procedimiento para sacar una matriz auxiliar para graficar
#============================================================

mataux:=proc()
global h,P1,pasos,tiempo;
```

```
local x,cont,m;
x:=tiempo[1];
with(linalg);
m:=round(pasos);
P1:=vector(m,0);
for cont from 1 to pasos by 1 do
x:=x+h;
P1[cont]:=x;
od;
#RETURN(P1);
end:


#============================================================
#Procedimiento que obtiene el vector a graficar
#============================================================

grafi:=proc()
global pasos,P,P1,P3;
local cont;
P3:=[];
for cont from 1 to pasos by 1 do
P3:=[op(P3),[P1[cont],P[cont]]];
od;
#RETURN (P3);
end:
```

Índice de figuras.

Índice de Tablas.

Bibliografía

[1] Sarmiento Reyes L. A., "*A partion method for the determination of múltiple DC operating points*". PhD thesis, Delft University of Technology, May 1994.

[2] Leon O. Chua, P. M. Lin, "*Computational Methods in CAD*". Academic Press, 1973.

[3] Leon O. Chua, Pen-Min Lin, "*Computer-Aided Analysis of Electronic Circuits, Algorithms and Computational Techniques*", 1975 by PRENTICE-HALL, Inc.

[4] Leon O. Chua, Charles A. Deseor, Ernest S. Kuh. *"Linear and Nonlinear Circuits"*, McGraw-Hill INTERNATIONAL EDITIONS, 1987.

[5] Jiri Vlach, Kishore Singhal, "*Computer Methods for circuit analysis and design*", Van Nostrand Reinhold New York, 1983.

[6] L. A. Sarmiento Reyes, "*Network: Part 1(a preliminary version)*", 1995.

[7] César Pérez López, *"Métodos Matemáticos y Programación con MAPLE V"*, RA-MA, 1998 primera impresión.

[8] U. Rembold and R. Dillman. "*Computer - Aided Design and Manufacturing Methods and Tools*". Springer - Verlag, 1986.

[9] R. K. Brayton, G. D. Hachtel and A. Sangiovanni - Vincentelli . *"A taxonomia of CAD for VLSI"*. Proceeding of the 1981 European Conference on Circuit Theory and Design. Pag. 34-55, The hague, the Natherland, 1981. Delft University Press/North Holland Publisehing Company.

[10] Roy H. Campbell, Nayeem Islam, Ralph Johnson, Panos Kougiouris and Peter - Madanly. *"Choices, Framework and Refinement"*. University of Illinois at Urbana - Champaign, Department of Computer Science 1991.