# DESARROLLO DE UNA INTERFAZ GRÁFICA PARA LA SOLUCIÓN DE FLUJOS DE CARGAS CON ARMÓNICOS

ANDRÉS LEONARDO MORENO WANDURRAGA

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO - MECÁNICAS
ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES
BUCARAMANGA

28 de noviembre de 2007

# DESARROLLO DE UNA INTERFAZ GRÁFICA PARA LA SOLUCIÓN DE FLUJOS DE CARGAS CON ARMÓNICOS

ANDRÉS LEONARDO MORENO WANDURRAGA

Trabajo de grado para obtener el título de Ingeniero Electricista

Director

Msc CÉSAR ANTONIO DUARTE GUALDRÓN

Ingeniero Electricista

Codirector

WILLIAM CARVAJAL CARREÑO

Ingeniero Electricista

UNIVERSIDAD INDUSTRIAL DE SANTANDER

FACULTAD DE INGENIERÍAS FÍSICO - MECÁNICAS

ESCUELA DE INGENIERÍAS ELÉCTRICA, ELECTRÓNICA Y TELECOMUNICACIONES

BUCARAMANGA

28 de noviembre de 2007

# Resumen

**TÍTULO:** DESARROLLO DE UNA INTERFAZ GRÁFICA PARA LA SOLUCIÓN DE FLUJOS DE CARGAS CON ARMÓNICOS.[1]

**AUTOR:** ANDRÉS LEONARDO MORENO WANDURRAGA [2]

**PALABRAS CLAVES:** SASP, Interfaz, Simulación, Híbrido, Circuitos, Armónicos.

**DESCRIPCIÓN**
La necesidad de predecir el comportamientos de los sistemas de potencia para la planificación de maniobras o comportamientos de cargas conlleva a la búsqueda de métodos de simulación que sean más exactos o más rápidos que los existentes, estos requerimientos dan origen a una técnica denominada método híbrido de simulación, una combinación del método en el dominio del tiempo y de la frecuencia, que se aplica a sistemas lineales con cargas no-lineales. El método se basa en aprovechar la rapidez de la simulación en frecuencia para simular las cargas lineales y la simulación en tiempo para solucionar las cargas no lineales.

La herramienta SASP (Simulación de Armónicos en Sistemas de Potencia) consta de dos partes, una prueba de concepto del método de simulación híbrida en Matlab® y una interfaz gráfica. La interfaz es amigable y simple, permite dibujar, modificar y simular circuitos básicos no lineales y de conmutación en tiempo, redes lineales en frecuencia y sistemas lineales con cargas no lineales o variantes en el tiempo utilizando el método híbrido.

En este proyecto se pudo corroborar que el método híbrido presenta ventajas para ciertos circuitos con redes lineales grandes en comparación con el número de cargas.

---

[1] Proyecto de Grado

[2] Facultad de Ingenierías Físico-Mecánicas. Escuela de Ingenierías Eléctrica, Eléctronica y de Telecomunicaciones. César Antonio Duarte Gualdrón

# Abstract

**TITLE:** DEVELOPMENT OF A GRAPHICAL INTERFACE TO SOLVE POWER FLOWS WITH HARMONIC COMPONENTS[3].
**AUTHOR:** ANDRÉS LEONARDO MORENO[4]

**KEYWORDS:** SASP, Interface, Simulation, Hybrid, Circuit, Harmonics.

**DESCRIPTION**

The necessity to predict power systems behavior to plan maneuvers or load impact on the system, leads to research on simulation methods to achieve more speed or precision. This research originates the hybrid method of simulation, a combination of time simulation mixed with frequency simulation that is specially applied to linear systems with non-linear loads.

This method is based on the frequency simulation speed advantage to simulate linear circuits and the time domain simulation to solve non-linear circuits. The software, SASP (Simulación de Armónicos en Sistemas de Potencia) is composed by two parts, the first one is a proof of concept of the hybrid method implemented in Matlab and the second one is the graphical interface. The interface is friendly, simple, clean, it allows to capture, modify and simulate basic non-linear circuits and time commutated, non-linear circuits in time domain, linear systems in frequency and linear systems with non-linear loads using the hybrid method.

On this project it was corroborated that the hybrid method shows advantages on circuit simulation with big lineal system and small number of loads.

[3]Undergraduate Project

[4]Electrical and Electronics Engineering School. César Antonio Duarte Gualdrón

# Índice general

**1. Introducción** **1**

1.1. Organización del proyecto . . . . . 3

1.2. Objetivo General . . . . . 4

1.3. Objetivos Específicos . . . . . 4

1.4. Marco Teórico . . . . . 4

1.4.1. Circuitos lineales y no lineales . . . . . 4

1.4.2. Series de Fourier . . . . . 5

1.4.3. Simulación en tiempo y frecuencia . . . . . 6

1.4.4. Simulación Híbrida . . . . . 8

**2. Arquitectura interna** **9**

2.1. Estructura principal . . . . . 10

2.2. Modelos . . . . . 12

2.2.1. Subcomponentes . . . . . 15

2.2.2. Subcircuitos . . . . . 16

2.2.3. Modelos existentes . . . . . 18

2.3. Elementos . . . . . 20

2.4. Conexiones . . . . . 21

**3. Interfaz** **23**

3.1. Canvas o espacio de trabajo . . . . . 25

3.2. Elementos . . . . . 26

3.3. Botones de modelos . . . . . 27

3.4. Botones de herramientas . . . . . 27

3.5. Conexiones . . . . . 29

3.6. Exportar datos . . . . . 30

**4. Simulación** **35**

4.1. Simulación en Tiempo . . . . . 36

4.2. Simulación en Frecuencia . . . . . 37

4.3. Simulación Híbrida . . . . . 40

4.4. Pruebas y resultados obtenidos . . . . . 43

4.4.1. Dominio del tiempo . . . . . 44

4.4.2. Dominio de la frecuencia . . . . . 48

4.4.3. Método Híbrido . . . . . 48

**5. Observaciones y conclusiones** **55**

5.1. Observaciones . . . . . 55

5.2. Conclusiones . . . . . 56

5.3. Trabajo Futuro . . . . . 58

**A. Manual de usuario** **59**

A.1. Crear un nuevo circuito . . . . . 59

A.2. Construcción del circuito . . . . . 60

A.3. Simulación y visualización de resultados . . . . . 61

# Índice de figuras

1.1. Ejemplo de un circuito lineal y uno no lineal . . . . . 5
1.2. Señal de ejemplo . . . . . 6
1.3. Magnitud de los componentes de frecuencia . . . . . 7

2.1. Ejemplo de una librería de modelos . . . . . 12
2.2. Icono de la bobina con los puntos de conexión . . . . . 15
2.3. Fuente de tensión y sus componentes . . . . . 17
2.4. Modelos de la bobina y el condensador . . . . . 19
2.5. Modelos del diodo y el SCR . . . . . 20

3.1. Boceto de la interfaz . . . . . 24
3.2. Iconos de herramientas . . . . . 28
3.3. Ejemplos de conexiones . . . . . 30

4.1. Esquema de la simulación en tiempo . . . . . 38
4.2. Esquema de la simulación en tiempo . . . . . 39
4.3. Esquema de la simulación en frecuencia . . . . . 40
4.4. Esquema de la simulación híbrida . . . . . 41
4.5. Carga con amperímetros en los terminales . . . . . 41
4.6. Equivalente Norton de un sistema trifásico . . . . . 42

4.7. Equivalente de un sistema trifásico con 3 frecuencias importantes 42
4.8. Circuito de prueba 15 . . . . . 44
4.9. Resultados en Gnucap del circuito de prueba 15 . . . . . 45
4.10. Resultados en SASP del circuito de prueba 15 . . . . . 45
4.11. Acercamiento: Resultados en Gnucap del circuito de prueba 15 . 46
4.12. Acercamiento: Resultados en SASP del circuito de prueba 15 . . 47
4.13. Circuito de prueba 13 . . . . . 47
4.14. Circuito de prueba 2 . . . . . 48
4.15. Resultados circuito de prueba 2 . . . . . 49
4.16. Circuito de prueba 19 en tiempo . . . . . 51
4.17. Esquema circuito de prueba 4 . . . . . 51
4.18. Esquema circuito de prueba 6 . . . . . 52
4.19. Resultados circuito de prueba 4 . . . . . 53
4.20. Resultados circuito de prueba 6 . . . . . 53
4.21. Transformada de Fourier de la corriente consumida por el circuito de prueba 6 . . . . . 54

A.1. Botones de herramientas . . . . . 60
A.2. Diálogo nuevo . . . . . 60
A.3. Conexión entre elementos como se ve en el software . . . . . 61
A.4. Ejemplo propiedades de tiempo . . . . . 62
A.5. Diálogo exportar . . . . . 63

# Índice de tablas

2.1. Miembros de las estructura principal . . . . . 10
2.2. Estructura globals . . . . . 11
2.3. Estructura Canvas . . . . . 11
2.4. Características de un modelo . . . . . 13
2.5. Propiedades de los elementos gráficos . . . . . 21
2.6. Propiedades de las conexiones . . . . . 22

3.1. Tabla de colores para los botones de los elementos . . . . . 28

4.1. Circuito 15, comparación de tiempos entre SASP y gnucap . . . 46
4.2. Circuito 13, comparación de tiempos entre SASP y gnucap . . . 47
4.3. Circuito 13, comparación de tiempos entre tiempo e híbrido (SASP) . . . . . 50
4.4. Circuito 19, comparación de tiempos entre tiempo e híbrido (SASP) . . . . . 50

A.1. Extensión de los datos exportados . . . . . 63

# Lista de Simbolos

| | |
|---|---|
| $V$ | Tensión |
| $I$ | Corriente |
| $Y_{barra}$ | Matriz de admitancias |
| $t$ | Tiempo |
| $h$ | Paso de tiempo |
| $C$ | Capacitancia |
| $L$ | Inductancia |
| $Z$ | Impedancia |
| $w_s$ | Frecuencia de muestreo |
| $\dot{x}$ | Primera derivada de X respecto al tiempo |
| $a_k$ | Coeficiente de Fourier |

# Capítulo 1

# Introducción

A medida que ha avanzado la electrónica, los dispositivos no lineales se hacen cada vez más comunes, tanto así, que incluso los equipos con comportamiento lineal, se les ha introducido control electrónico o componentes electrónicos que hacen que la carga pierda su característica lineal. Estas cargas no lineales tienen la desventaja de inyectar armónicos a la red eléctrica. Actualmente, las cargas no lineales son tan comunes que la inyección de armónicos a la red por éstas, se convirtió en un problema de calidad de la energía eléctrica y en un reto para la operación económica y segura del sistema por parte de las empresas[1].

Debido a los diversos inconvenientes de los armónicos que circulan por el sistema, se han realizado y se están haciendo varios estudios para tratar de diseñar soluciones y establecer una mejor regulación para la inyección de armónicos a la red[2].

Este proyecto busca implementar una herramienta de análisis, que usa el método híbrido tiempo-frecuencia para que sirva de apoyo en los estudios que se están realizando, y que presente ventajas con respecto a los actuales métodos

de simulación.

Los resultados de las pruebas de simulación se comprobaron con el software gspiceui, usando los simuladores gnucap, desarrollado por Albert Davis, y ngspice basado en Spice3f5, Cider1b1 y Xspice. Todos con licencias de software libre.

La herramienta SASP (Simulación de Armónicos en Sistemas de Potencia) fué programada en la versión 7 de Matlab, un lenguaje de programación orientado hacia labores científicas. Este lenguaje se escogió sobre los lenguajes tradicionales por tener un muy buen soporte matemático, con librerías optimizadas y por la simplicidad de un lenguaje de alto nivel, como manejo de memoria y la facilidad en la construcción de interfaces gráficas. Para este proyecto se hizo un uso extensivo de las capacidades de Matlab en el manejo de matrices, sus funciones de matrices dispersas, el manejo de estructuras y la herramienta para desarrollar interfaces gráficas.

Matlab es multiplataforma, puede correr en Windows y en linux, los únicos detalles a tener en cuenta para correr el proyecto fueron los nombres sensitivos a mayúsculas de linux, y las terminaciones de línea en ambos sistemas operativos. Dependiendo del sistema operativo donde se corra el proyecto, y la configuración, los colores de la interfaz cambian, por eso se escogió un color fijo para el fondo de los elementos, de esta manera se aseguró que la paleta de colores concordara.

Para la instalación del programa simplemente se copia la carpeta SASP en el computador y se cambia el directorio de trabajo en Matlab a éste; para ejecutar el programa, se escribe “SASP” en la consola de Matlab, si se escribe en letra minúscula habrá una advertencia pero el simulador correrá de todos

modos. El proyecto no corre en versiones de Matlab anteriores a la 7, debido al cambio en las herramientas gráficas de Matlab.

El programa tiene cerca de 7000 líneas de código y 120 archivos, los archivos empiezan con un sufijo indicando a que sección del programa pertenecen, luego el objeto sobre el que actúan y finalmente la acción de la función.

Durante el desarrollo del trabajo, se hicieron pequeñas contribuciones al software gspiceui que se pueden ver en la página del proyecto en Sourceforge (http://www.sourceforge.net), aunque los parches no fueron aceptados tal como se enviaron, el autor incluyó los arreglos.

## 1.1. Organización del proyecto

El capítulo 2 trata sobre el manejo de los datos, como se relacionan, el diseño del programa, como se agregan modelos y elementos, que información y características posee cada elemento y como se integra la parte gráfica con los algoritmos de simulación.

El capítulo 3 profundiza en la parte visual del software, las opciones disponibles y las funciones y métodos usados.

El capítulo 4 explica como se programaron los métodos de simulación, su uso, y finalmente muestra los resultados obtenidos con las simulaciones.

En el capítulo 5.2 se presetan las observaciones y conclusiones del trabajo, el trabajo futuro, las limitaciones y ventajas que tiene el software.

## 1.2. Objetivo General

Diseñar y programar una herramienta de simulación para la solución de flujo de cargas con armónicos.

## 1.3. Objetivos Específicos

- Programar las rutinas para la simulación en tiempo, en frecuencia y la implementación del método híbrido.
- Evaluar la convergencia, precisión y velocidad del método de solución.
- Diseñar un entorno gráfico, que permita la introducción de los diagramas circuitales del sistema y los valores de sus parámetros.

## 1.4. Marco Teórico

### 1.4.1. Circuitos lineales y no lineales

Los circuitos lineales son los circuitos en los que sus relaciones tensión-corriente pueden ser expresadas como ecuaciones lineales, es decir, la salida del circuito, puede expresarse como una transformación lineal de la entrada, estos circuitos son fáciles de simular debido a la simplicidad de sus ecuaciones.

Los circuitos no lineales como las cargas de los sistemas de potencia, conformados por dispositivos electrónicos y de conmutación como por ejemplo rectificadores y FACTS tienen un alto componente de armónicos y deben ser simulados en el dominio del tiempo. En algunos casos y dependiendo de la

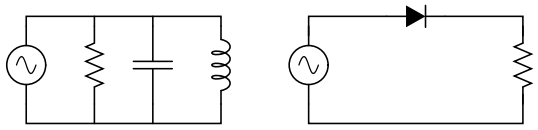


Figura 1.1: Ejemplo de un circuito lineal y uno no lineal

simulación se pueden linealizar los elementos no lineales sobre un rango pequeño de simulación [3, 4], y utilizar un método de simulación menos costoso computacionalmente. En la Figura 1.1 se muestran dos ejemplos de circuitos, uno lineal y uno no lineal.

### 1.4.2. Series de Fourier

El método de simulación híbrida hace uso extenso de las series de Fourier para el análisis del sistema con las cargas, más especificamente el paso de tiempo a frecuencia. Las series de Fourier permiten hacer una representación de una señal como una suma de señales básicas, estas señales básicas son exponenciales complejas. La ecuación 1.1 muestra la expresión matemática necesaria para hacer la representación en señales básicas teniendo la señal original, y la ecuación 1.2 la expresión para obtener la señal original a partir de las señales básicas.[5]

$$a_k = \frac{1}{T}\int_T x(t)e^{-jkw_0t}dt = a_k = \frac{1}{T}\int_T x(t)e^{-jk(2\pi/T)t}dt \tag{1.1}$$

$$x(t) = \sum_{k=-\infty}^{+\infty} a_k e^{jkw_0t} = \sum_{k=-\infty}^{+\infty} a_k e^{jk(2\pi/T)t} \tag{1.2}$$

La magnitud de los coeficientes $a_k$ es la magnitud de la componente de

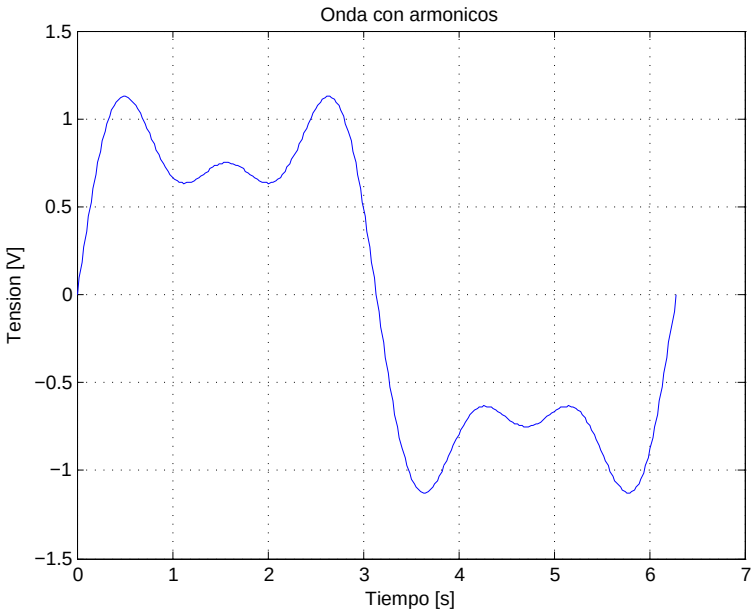


Figura 1.2: Señal de ejemplo

frecuencia $kw_0$. Usando los $a_k$ se puede hacer una gráfica del espectro de una señal, es decir, graficar la composición de la señal por frecuencias y ver su contenido armónico. En la Figura 1.2 se muestra una señal no senoidal y en la Figura 1.3 se muestra la magnitud de los componentes $a_k$.

### 1.4.3. Simulación en tiempo y frecuencia

Hay diferentes algoritmos de simulación, en el proyecto se programaron la simulación DC, la transitoria y la simulación híbrida. La simulación DC es la más simple de todas y está basada en el análisis nodal, usando la ley de Kirchoff se crea la matriz de admitancias $Y_{barra}$ y el vector $I$ de inyección de corriente del sistema, a continuación se soluciona la ecuación del sistema y se obtiene la tensión de los nodos:

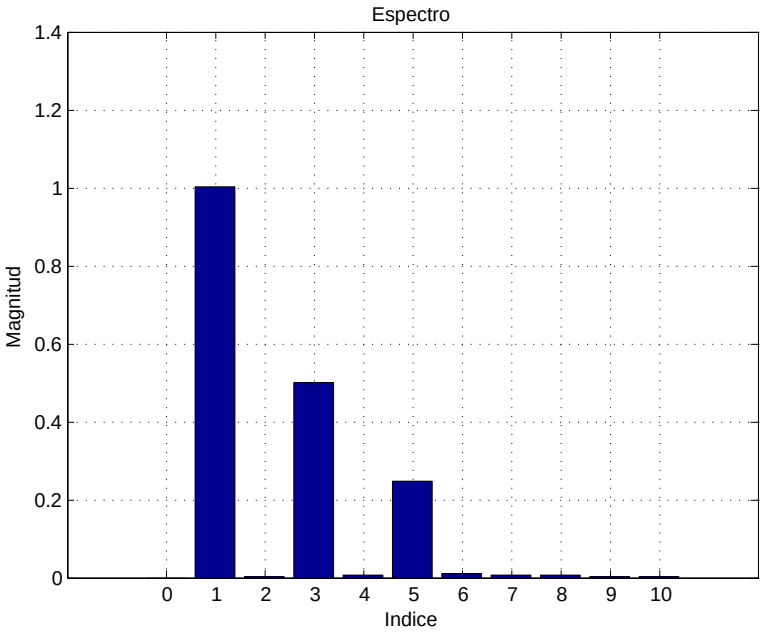


Figura 1.3: Magnitud de los componentes de frecuencia

$$V = Y_{barra}^{-1} I \tag{1.3}$$

En la simulación DC las bobinas son reemplazadas por cortos y los condensadores por circuitos abiertos [4], sin embargo en el dominio de la frecuencia a las bobinas y condensadores se les asigna una impedancia y se puede resolver el sistema usando análisis DC, normalmente estas simulaciones se resuelven usando descomposición LU y haciendo sustitución de variables [6], pero esto es útil cuando la $Y_{barra}$ del sistema no cambia, y se pueden ahorrar los pasos de la eliminación gaussiana.

La simulación transitoria es un problema de valores iniciales, primero se deben representar los elementos almacenadores de energía por sus modelos

compañía [7] tal como se explica en la sección 2.2.3, esto reduce el problema a un problema DC, finalmente se resuelve la matriz para cada paso de tiempo.

### 1.4.4. Simulación Híbrida

La simulación híbrida nace de las necesidades actuales de simulación, los sistemas de potencia son muy grandes, las cargas tienen un alto contenido de no linealidad, y los métodos de simulación descritos antes tienen sus ventajas y desventajas al enfrentarse a sistemas con estas características. El método híbrido une el dominio de la frecuencia y la simulación en tiempo, para simular por partes un sistema.

En el capítulo 4 se describe la implementación del método híbrido, que consiste en la simulación del sistema en el dominio de la frecuencia, la detección de las tensiones frontera para alimentar las cargas, la simulación de las cargas en tiempo, el análisis del contenido armónico de las cargas y la inyección de la corriente consumida por la carga en el sistema [7].

# Capítulo 2

# Arquitectura interna

La arquitectura interna del programa se diseñó de forma que concordara con las características y herramientas de Matlab, desafortunadamente hay poca documentación sobre proyectos grandes o medios usando la interfaz gráfica de Matlab. La arquitectura planteada al comienzo de la tesis hacía un uso extensivo de la estructura principal de la figura (*figure* en Matlab), esta estructura que se llama *handle* (porque fué diseñada para contener los manejadores de los elementos gráficos) se puede acceder con la función *guidata* de Matlab, sin embargo, conforme avanzó el proyecto y se necesitó más flexibilidad para las simulaciones y los datos, se optó por trasladar los datos que no se necesitaban a nivel global a espacios separados, esta separación consiste en guardar los datos en objetos de Matlab (axes, lines,...), en lugar de ser guardados en la figura principal.

Los espacios separados hacen uso de las funciones *setappdata* y *getappdata*, la ventaja de estas dos funciones es que se guardan en cada objeto, por ejemplo, un subcircuito puede tener parámetros de simulación diferentes al circuito

| *manejadores* | Manejadores a los objetos gráficos |
|---|---|
| globals | Estructura con datos globales |
| config | Estructura con los nombres de los íconos de las herramientas y modelos |
| modelos | Biblioteca de modelos |
| elements | Estructura con los elementos gráficos |
| conex | Estructura con las conexiones |

Tabla 2.1: Miembros de las estructura principal

principal.

## 2.1. Estructura principal

La estructura principal tiene los manejadores de los elementos gráficos, estos se usan para leer y cambiar sus datos, propiedades como el tamaño, el color, etc. Pero también contiene otra información, como la biblioteca de modelos, la lista de elementos gráficos, la lista de las conexiones, una estructura con datos globales y la configuración de los botones de herramientas, la Tabla 2.1 muestra los elementos de la estructura principal y sus usos.

La estructura principal es creada por Matlab al inicio de la figura, pero solo contiene los manejadores de los objetos gráficos creados usando el comando *guide* (graphic user interface design environment), así que el resto de datos se inicializan en la función *SASP_OpeningFcn* que se ejecuta cuando los elementos ya están creados pero no son visibles para evitar que Matlab los sobreescriba.

El miembro "globals" guarda datos que se necesitan en varias rutinas del programa, la cantidad de elementos, la cantidad de conexiones, y datos que se utilizan en las rutinas de dibujo, la Tabla 2.2 muestra los datos de la estructura globals y la Tabla 2.3 muestra los datos de la estructura Canvas, un miembro

| | |
|---|---|
| cElementos | Cantidad de elementos gráficos en el circuito. |
| cConex | Cantidad de conexiones en el circuito. |
| cnodos | Número de nodos en el circuito . |
| Canvas | Estructura con las dimensiones del canvas. |
| Xpix2norm | Razón horizontal entre pixeles y coordenadas normalizadas. |
| Ypix2norm | Razón vertical entre pixeles y coordenadas normalizadas. |
| Path | Ruta al archivo. |
| File | Archivo. |

Tabla 2.2: Estructura globals

| | |
|---|---|
| startX | Coordenadas en X de la esquina inferior izquierda |
| startY | Coordenadas en Y de la esquina inferior izquierda |
| Width | Ancho |
| Height | Alto |

Tabla 2.3: Estructura Canvas

por composición de la estructura “globals” que contiene los datos sobre las coordenadas del área de trabajo (canvas).

El miembro “config” de la estructura principal, guarda la información de los íconos de los botones de las herramientas, el nombre de los íconos de las herramientas se encuentra en el archivo “config.txt”, este es un archivo de texto plano con una línea por cada botón. Los miembros “elements” y “conex” son matrices con los datos que se muestran en las secciones 2.3 y 2.4 respectivamente. Todas las estructuras de los modelos se organizan en una estructura que funciona como una biblioteca de modelos como muestra la Figura 2.1, esta estructura es el miembro “modelos” de la estructura principal.

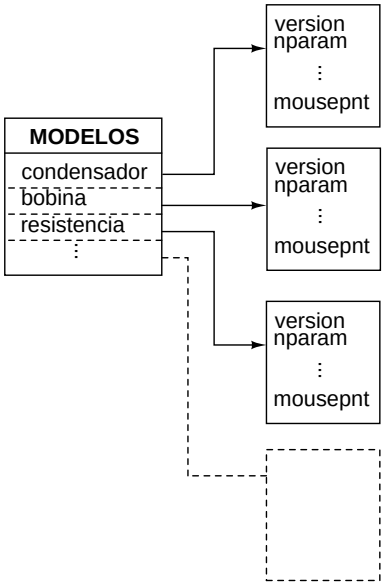


Figura 2.1: Ejemplo de una librería de modelos

## 2.2. Modelos

Los archivos de modelos describen los elementos de los circuitos, y están organizados en carpetas dependiendo del método de simulación que los necesita. Los archivos de modelos tiene una forma de escritura muy particular que explota las capacidades de Matlab con estructuras. Una plantilla llamada 'plantilla.txt' que incluye ayuda de los parámetros de los modelos se encuentra en la carpeta Models. Cada línea del modelo debe empezar por un punto, las líneas que no empiezan con un punto, son interpretadas como comentarios en el modelo. En la Tabla 2.4 aparecen las características que necesita normalmente un modelo junto con la característica que describen.

La función responsable de cargar el modelo lee línea a línea el archivo y lo anexa a una estructura con el nombre del modelo como muestra la Tabla 2.4, de esta manera es fácil agregar más parámetros compartiendo el mismo tipo

| | |
|---|---|
| .version | Version del archivo del modelo |
| .nParam | Parámetros propios que necesita el modelo |
| .exPort | Nodos que se exportan |
| .componente.Nombre | Componente |
| .icono | Nombre del icono para el canvas (.bmp) |
| .bicono | Nombre del icono del botón(.bmp) |
| .mousepnt(1,:) [0.5,1] | Puntos para conexión |

Tabla 2.4: Características de un modelo

de datos de Matlab y su sintaxis.

No todos los modelos tienen los mismos datos, dependiendo del método de simulación el modelo puede necesitar más o menos información, por ejemplo, el método híbrido es el único que necesita el dato "tipo" ya que combina dos tipos de modelos, sin embargo en tiempo y frecuencia, todos los modelos involucrados son del mismo tipo, así que esta información es redundante. El archivo del modelo de simulación en tiempo de la bobina, es un buen ejemplo para entender la construcción de modelos en SASP:

```
.version 1
.nParam {'L'}
.exPort     [1,2]
**** Modelo de trapecio ****
componente.R      {'resistencia',[1,2],{'2*L/dt'}}
componente.Isrc   {'isrc',[2,1],{'dt/(2*L)*-V(n-1)+I(n-1)'}}
****   Modelo de euler  ****
.componente.R      {'resistencia',[1,2],{'L/dt'}}
.componente.Isrc   {'isrc',[1,2],{'-I(n-1)'}}
.icono ['bobina']
.bicono ['bbobina']
```

```
.mousepnt(1,:) [0.5,1]
.mousepnt(2,:) [0.5,0]
```

La versión del modelo es 1. El único parámetro de entrada es L, es decir, cuando se agrega el modelo al circuito, este solo necesita el valor de L para poder ser simulado, L puede no ser un número, sino un valor dependiente del tiempo, del paso, o de cualquier otro parámetro interno de la simulación como por ejemplo $2 * L * dt$, utilizando esta característica se implementan los modelos de compañía de las bobinas y condensadores como se explica en la sección 2.2.3. Dentro de la lista de parámetros no pueden ir variables reservadas como: V, V(n-1), I, I(n-1), dt, etc. Como por ejemplo: .nParam {'L','I'} pues se evaluaría incorrectamente el valor de los parámetros. Los nodos que se exportan son el nodo número 1 y número 2. Las líneas que siguen, y que no empiezan con un punto, son comentarios.

Los subcomponentes del modelo se introducen llenando una celda por cada uno, como se explica en la sección 2.2.1.

La entrada icono es el nombre del archivo .bmp que será usado como la imagen que respresenta el elemento en el canvas, todos los íconos se buscan dentro de la carpeta Icons/, debido a que algunos de estos íconos son muy grandes para ser mostrados dentro de un botón, el miembro bicono sirve para indicar que icono se usa para el botón.

Los datos mousepnt indican en que lugar del ícono se encuentran los puntos de conexión para el mouse, para esto se asume que el ícono tiene coordenadas de 0 a 1 en los ejes X y Y. En este ejemplo el primer punto se encuentra en la mitad en X y la parte superior del icono, y el segundo en la mitad de X y en

la parte inferior del icono como muestra la Figura 2.2.

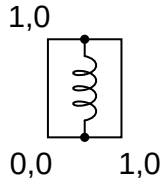


Figura 2.2: Icono de la bobina con los puntos de conexión

### 2.2.1. Subcomponentes

La introducción de componentes en los modelos se hace escribiendo una línea por cada componente de la siguiente manera:

.componente.Nombre {'modelo',[nodos],{parámetros}}

**Nombre:** Debe ser único o los elementos con el mismo nombre se sobreescribirán.

**Modelo:** Es el nombre del archivo .txt que contiene la información del modelo.

**Nodos:** Es un vector con el número de los nodos donde se conecta el componente, un número de nodo cero (0), indica conexión a tierra.

**Parámetros:** Es una celda con los valores que necesita el componente, puede ser un número, o una expresión matemática con variables internas del programa, incluyendo los parámetros del modelo, en este caso L y valores como V, V(n-1), I, I(n-1).

Hay que tener en cuenta que no se pueden introducir espacios dentro de la celda principal o de otro modo la lectura del archivo fallaría, pues el espacio es el delimitador del par, nombre-valor.

### 2.2.2. Subcircuitos

El software permite el uso de subcircuitos, siempre y cuando al final, todo se pueda reducir a primitivas, las primitivas son elementos que no se pueden expresar en subcomponentes y que el software maneja directamente. En esta versión, las resistencias, las fuentes de corriente, las tierras y los nodos son primitivas, los demás modelos están construidos sobre estas primitivas.

Después de definir un modelo inicial, se pueden definir más modelos usando este como un componente, como cada modelo tiene parámetros propios, el software hace una propagación de parámetros desde los modelos superiores hasta los subcircuitos.

La solución consistió en pasar los parámetros y sus valores en una lista de parámetros, estas listas son celdas con cadenas para ser evaluadas, por ejemplo: 'L=10', 'R=2*L'. La evaluación de la celda se hace de izquierda a derecha y se usa para pasar variables de un alcance (scope) a otro; las variables son visibles únicamentes por las funciones donde fueron definidas, un forma de solucionar esto es declarando las variables globales, pero este método tiene el problema de crear conflicto en funciones que tienen declaradas variables con el mismo nombre, la otra solución es pasar las variables como argumentos, pero el número de variables es cambiante y no se les puede asignar nombre, por eso la solución fue imitar el soporte de Matlab de la función *varargin* pero permitir asignar un nombre a las variables. Hay que tener en cuenta que cuando hay variables con nombres repetidos en la lista, sus valores se sobreescriben.

Como un ejemplo, el modelo de la fuente de tensión de la Figura 2.3 está compuesto por dos subcomponentes, la fuente de corriente y la resistencia.

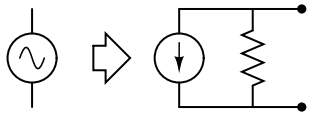

Figura 2.3: Fuente de tensión y sus componentes

La inclusión de los componentes se hace con las siguientes líneas:

```
.componente.Isrc    {'isrc',[1,2],{'V/R','Fs'}}
.componente.R       {'resistencia',[1,2],{'R'}}
```

El modelo de la fuente de corriente solo tiene un elemento:

```
.componente.Isrc       {'isrc',[1,2],{'I','Fs'}}
```

Si se supone que la fuente de tensión tiene los valores V=10, R=1e-3, Fs=60. la lista de parámetros que vería la fuente de corriente sería:

{'V=10;', 'R=1e-3;', 'Fs=60;', 'Fs=Fs;'}

El único componente de la resistencia es:

```
.componente.R     {'resistencia',[1,2],{'R'}}
```

y la lista que vería la resistencia sería:

{'V=10;', 'R=1e-3;', 'Fs=60;', 'R=V/R;'}

En el caso de la resistencia hay dos variables con el mismo nombre, la 'R' de la fuente y la 'R' de la resistencia, como la celda se evalúa de izquierda a derecha, es decir, evaluando primero los valores de los subcircuitos y luego los valores de sus componentes, no hay ningún problema con los nombres (keys) repetidos

en la lista, pues en la segunda evaluación de 'R', el valor se sobreescribirá. Hay que notar que los parámetros en la lista solo dependen de los parámetros que estén situados a la derecha de este, esto se debe a que están organizados jerárquicamente, los valores situados más a la izquierda en la lista son los valores para los parámetros de los componentes de los circuitos que estan situados más a la derecha de la lista. La sobreescritura de R no es relevante porque las evaluaciones que requerian de ese parámetro ya ocurrieron.

### 2.2.3. Modelos existentes

Los modelos usados en SASP son modelos básicos, en el caso de la bobina y el condensador se usaron los modelos de compañía; estos modelos se deducen usando las ecuaciones de tensión-corriente de estos elementos, y finalmente usando un método numérico para llegar a un circuito lineal que se pueda introducir en el circuito y que se comporte de la misma forma que el componente original [8].

En el caso de la bobina su relación tensión corriente es:

$$V = L\frac{di}{dt} \tag{2.1}$$

Si se usa Euler implícito $x_t = x_{t-1} + h\dot{x}$ para resolver la ecuación diferencial se obtiene:

$$I_n = I_{n-1} + \frac{h}{L}V \tag{2.2}$$

Aplicando el mismo procedimiento con la ecuación de tensión-corriente del condensador se puede llegar a un resultado similar y al modelo de la Figura

2.4. Normalmente los simuladores usan el modelo obtenido a través de la regla trapezoidal, pero estos modelos tienen problemas de oscilaciones y se necesita implementar un CDA (ajuste crítico de amortiguamiento) [9]. Los modelos deducidos usando Euler implícito no son tan precisos como los de la regla trapezoidal, pero son más estables.

Para los semiconductores se escogieron modelos con cambio de resistencia (lineal a pedazos), que emulan la curva de los elementos, como se ve en la Figura 2.5. El diodo y el SCR tiene dos resistencias, una para el comportamiento de conducción y otra para el estado de no-conducción, específicamente para el SRC se creó un parámetro para introducir una función de Matlab que sirve como función de disparo.

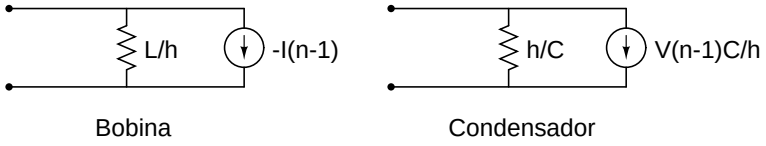


Figura 2.4: Modelos de la bobina y el condensador

Las fuentes de tensión puras no son permitidas (debido al análisis nodal), deben tener una resistencia asociada, usando el equivalente Norton se modela la fuente de tensión usando una fuente de corriente y una resistencia en paralelo con esta. El modelo de la línea es un modelo PI de línea media [6], aunque cambiando el valor de los parámetros se puede obtener el modelo de la línea corta y larga [10].

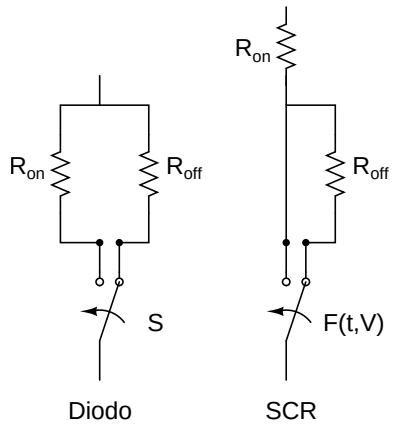


Figura 2.5: Modelos del diodo y el SCR

## 2.3. Elementos

Los elementos gráficos en el área de trabajo (canvas) son objetos *axes* de Matlab con un objeto imagen dibujado sobre estos, así que el elemento comparte todas las propiedades de los *axes* y las imágenes, y otras propiedades agregadas como valores y nodos, es como si se utilizara herencia, la Tabla 2.5 muestra los datos agregados al elemento, aunque en algunos existe redundancia.

Cuando los elementos gráficos por ejemplo los *axes* de los elementos, son agregados en tiempo de ejecución, Matlab no actualiza la estructura principal con el manejador del objeto agregado, por eso es necesario tener una referencia al manejador en el miembro hObject.

Aunque la posición, el alto y el ancho se pueden obtener al leer las propiedades del *axe*, es más cómodo y rápido tener estos valores en la estructura del elemento.

El miembro nodos guarda los nodos a los cuales está conectado el elemento y están inicializados a -1, esta información se actualiza con la función *co-*

| | |
|---|---|
| hObject | Handle al objeto axe del elemento |
| model | Modelo del elemento |
| posicion | Posicion del elemento sobre el canvas |
| elmAlto | Altura del elemento en unidades normalizadas |
| elmAncho | Ancho del elemento en unidades normalizadas |
| rot | Rotación del elemento en múltiplos de 90 grados |
| nodos | Los nodos a los cuales esta conectado el elemento |
| valores | Valores de los parámetros del elemento |
| initialized | Bandera que indica si el elemento tiene valores |

Tabla 2.5: Propiedades de los elementos gráficos

*nex_list_create*, y corresponde exactamente a la que se encuentra en la estructura de conexiones, la razón para la redundancia de datos es rapidez. Si se buscan las conexiones de un elemento en la lista de conexiones se tendría que iterar sobre toda la estructura de conexiones, sin embargo el miembro nodos da acceso a las conexiones de un elemento inmediatamente.

El miembro valores es una celda con cadenas de texto, los valores se guardan como cadenas de texto para permitir más flexibilidad, un ejemplo de esta flexibilidad es poder introducir una fuente de DC con un valor de corriente $sin(t)$, que haría que esta fuente de DC se comportara como una fuente alterna.

Las funciones que actúan directamente sobre los elementos gráficos empiezan por el prefijo “elm”.

## 2.4. Conexiones

Las conexiones están representadas por líneas en el canvas, que van desde el terminal de un elemento al terminal de otro (los nodos se consideran como elementos). Las conexiones se almacenan en una matriz de la estructura principal llamada *handles.conex* como vectores fila con los datos: [hline strIndex strNodo

| hline | Manejador de la línea |
|---|---|
| strIndex | Index del primer elemento |
| strNodo | Terminal del primer elemento |
| endIndex | Index del segundo elemento |
| endNodo | Terminal del segundo elemento |
| numNodo | Número de Nodo asignado |

Tabla 2.6: Propiedades de las conexiones

endIndex endNodo numNodo]. La Tabla 2.6 muestra los datos miembros de las conexiones y sus propósitos.

Las funciones que actúan sobre las conexiones empiezan con el prefijo "conex". La función encargada de asignar el número de las conexiones es *conex_list_create*, esta función asigna un número de nodo cuando ninguna de las terminales de la conexión tiene número asignado y propaga el número del nodo cuando alguna de las dos o las dos terminales ya tienen números asignados.

# Capítulo 3

# Interfaz

La interfaz se desarrolló usando estructuras de datos, y funciones que actúan sobre las estructuras de datos, muy similar a una programación orientada a objetos en C. Matlab tiene opciones para usar clases y objetos pero realmente no distan mucho del enfoque tomado, incluso en la sintaxis para llamar los métodos hay que pasar como argumento la instancia del objeto.

La interfaz gráfica se creó para que sea lo mas amigable posible, teniando cuidado especial en que los íconos reflejaran su función, se sometió a prueba con varios usuarios y se realizaron las modificaciones a su apariencia de acuerdo a las sugerencias recibidas. En la elaboración de los íconos se usó la paleta de colores con licencia de dominio público y las guias de diseño de "Tango project", un proyecto para crear una interfaz gráfica consistente en software libre. Todos los íconos se encuentran en la carpeta Icons, incluyendo los íconos de los botones y de los modelos. Dentro de la carpeta Icons, se encuentra la carpeta xcf que contiene los proyectos originales de GIMP (GNU Image Manipulation Program) para la creación de los íconos.

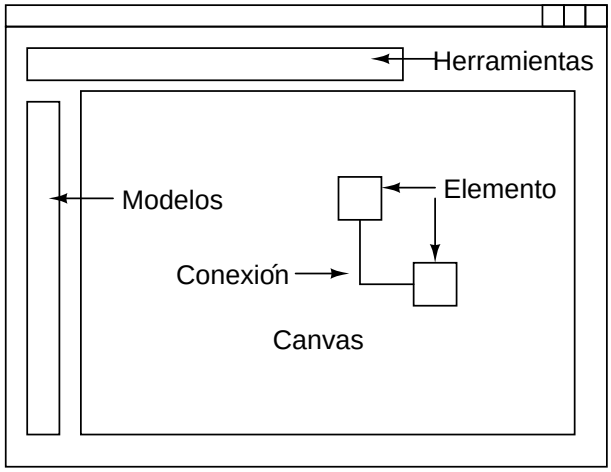


Figura 3.1: Boceto de la interfaz

El graficador (renderizador) escogido fué OPenGL. OpenGL es un conjunto de funciones para programación de gráficos en segunda y tercera dimensión que son independientes de la arquitectura del computador, así que no es necesario tener en cuenta singularidades de hardware pues cada fabricante está encargado de hacer la implementación sobre su hardware.

Una tarjeta aceleradora de video sirve para dibujar en pantalla rápidamente los circuitos, pero hay que configurar Matlab para que use el hardware en lugar de las librerías propias, dependiendo del sistema operativo esta configuración puede cambiar, pero las instrucciones están en la ayuda de Matlab si se busca por OpenGL.

La Figura 3.1 muestra un boceto simple de la interfaz.

## 3.1. Canvas o espacio de trabajo

El canvas es el espacio de trabajo donde se dibujan los diferentes componentes y se conectan, el canvas realmente es un objeto *axes* de Matlab, con las coordenadas definidas cuidadosamente para recibir las entradas del mouse y dibujar en el lugar correcto el objeto. Detrás del canvas hay un objeto *frame* para dibujar un marco alrededor del canvas.

En Matlab, las unidades de una figura pueden ser, centímetros, pulgadas, puntos, pixeles, caracteres y normalizadas. Para la figura principal, se determinó que las unidades normalizadas eran más simples porque permitían escoger el porcentaje del Canvas sobre la figura principal sin hacer cálculos extras y la figura principal puede ser escalada para que corra fácilmente en monitores con diferentes resoluciones.

Cuando hay una entrada de mouse, las coordenadas devueltas, dependen del elemento donde fué hecho el click y sus unidades, sin embargo, al dibujar nuevos elementos sobre la figura principal, las coordenadas usadas son las relativas a la figura principal y no al objeto en el cual se hizo el click, por eso fué necesario hacer corresponder las coordenadas del canvas con las coordenadas de la figura principal.

El canvas tiene una cuadrícula invisible para posicionar los elementos. La función encargada de filtrar las coordenadas se llama *grid_filter* y en el código de esta se puede cambiar el tamaño de la cuadrícula, por defecto son 60 divisiones en el eje X y 40 en el eje Y.

## 3.2. Elementos

Los objetos de los elementos son realmente objetos *axes* de Matlab con una imagen dibujada sobre ellos que capta los eventos del mouse sobre el elemento.

La interfaz permite, actuar sobre los elementos; actualmente las acciones visuales incluidas son: mover, rotar, cablear y borrar, las acciones relacionadas con la información interna son, cambiar los valores del elemento, ver la estructura de datos del elemento, editar modelo, recargar modelo (despues de editar un modelo, este debe ser actualizado en la biblioteca de modelos) y el último grupo de opciones está relacionado con la simulación, estas son: propiedades de simulación, simular, plot y exportar.

En las imágenes de los elementos, el color blanco se toma como transparencia. Debido a un error de programación de Matlab que parece ser de indexación (Fence-posting) la rotación de elementos con transparencias no se dibujan correctamente y la función encargada de dibujar la transparencia omite el primer pixel de cada columna de la gráfica, por eso para rotar los elementos es necesario eliminarlos y dibujar otros en la posición nueva. Cuando un elemento es dibujado, al igual que el canvas, el eje Y debe ser invertido (ya que Matlab lo invierte cuando dibuja la imagen) y las coordenadas internas deben arreglarse para que coincidan con las de la figura principal.

Cada elemento tiene un conjunto de datos que se encuentra en la matriz handles.Elements. Los datos de cada elemento se muestran en la Tabla 2.5. Para poder ver estos datos, se hace click derecho sobre el elemento y luego click en el menú -Estructura de datos-, los datos aparecerán en la ventana principal de Matlab.

## 3.3. Botones de modelos

Los botones de modelos están situados en una columna en la parte izquierda de la interfaz, se encuentran originalmente en blanco, es decir, no tienen ningún modelo asociado; cuando el usuario escoge un método de simulación (tiempo, frecuencia, híbrido), el programa lee un archivo de configuración llamado 'botones.txt' situado en la carpeta correspondiente al método de simulación. Para cambiar el modelo asociado a un botón, se puede hacer de dos maneras, editando el archivo de configuración, o con click derecho sobre el botón y en el menú cambiar modelo; si se edita el archivo de configuración los cambios se mantendrán durante las sesiones con SASP, sin embargo el cambio usando el menú, solo se recordará hasta que el programa se cierre. La otra opción del menú es actualizar modelo, esto es útil si el archivo del modelo fué modificado después de iniciar el SASP.

Los botones de los modelos tienen fondos de colores, dependiendo del tipo de elemento que sea, si es un elemento pasivo, tiene un fondo de color azul, si es un elemento activo, tiene un fondo rojo, si es un elemento no lineal tendrá un fondo morado, si es un elemento genérico tendrá un fondo gris y si tiene fondo verde, es tierra, la tabla 3.1 resume los colores escogidos con el tipo de elemento.

## 3.4. Botones de herramientas

Los botones de herramientas están situados en la parte superior de la ventana principal y tienen diferentes tareas, como: nuevo, abrir, guardar; las que se encuentran típicamente en cualquier software, pero además de estas tareas po-

| Color | Tipo |
|---|---|
| Azul | Pasivo |
| Rojo | Activo |
| Morado | No lineal |
| Gris | Genérico |
| Verde | Tierra |

Tabla 3.1: Tabla de colores para los botones de los elementos

Figura 3.2: Iconos de herramientas

see otras de simulación, visualización y de exportación. La Figura 3.2 muestra los botones de las herramientas.

Las herramientas de abrir y guardar trabajan con archivos .mat de Matlab por comodidad, guardar vuelca los datos del programa a disco usando la función *save* y abrir lee los datos usando *load*, para después dibujar el circuito. Como las propiedades de los elementos (nodos a plotear, graficador, etc..) tienen nombres iguales, no se puede usar *save* directamente, porque se presentaría una colisión de nombres y se sobreescribirían, para evitar esto el software renombra las propiedades de la siguiente manera: 'propiedad_' índice_del_elemento '_' nombre_de_la_propiedad. Por ejemplo, para el segundo elemento, la propiedad graficador se guardaría como: "propiedad_2_graficador"

El botón de visualización tiene un ícono con dos gráficas, una roja y una azul, los resultados se puede ver a través de dos graficadores, uno es el de Matlab, el segundo graficador es gnuplot, un graficador extremademente poderoso con numerosas opciones.

El botón de propiedades, tiene un ícono de un papel y un lápiz, al hacer click sobre este botón, aparece el diálogo propiedades, donde se pueden cambiar: los parámetros de simulación, el título de la gráfica, el graficador a ser utilizado, y los números de los nodos cuya tensión va a ser graficada.

El botón siguiente tiene un ícono que muestra unas ventanas con equis, cierra todas las ventanas de gráficas en Matlab.

El botón de simulación tiene un ícono verde con la forma del símbolo reproducir de equipos de audio y video, este botón en el encargado de correr la simulación.

El botón de exportación tiene un ícono con una hoja y una flecha azul, este botón abre el dialogo de exportación, que permite guardar los datos de la simulación en diferentes formatos.

El botón de colores, tiene un ícono con una paleta de tres colores, esta herramienta se encarga de colorear los nodos del mismo color que la gráfica de su tensión.

## 3.5. Conexiones

Las conexiones están representadas por líneas dentro del circuito, las líneas se dibujan siempre como segmentos horizontales o verticales o una mezcla, pero nunca en diagonal, para esto, se determina el vector de salida de los elementos como el vector que va del centro del elemento al punto de conexión del elemento, luego se determina si los vectores de salida son ortogonales, paralelos o antiparalelos y se dibuja la conexión basado en estos datos como se muestra en la Figura 3.3. Las conexiones van ligadas a los elementos y más especificamente

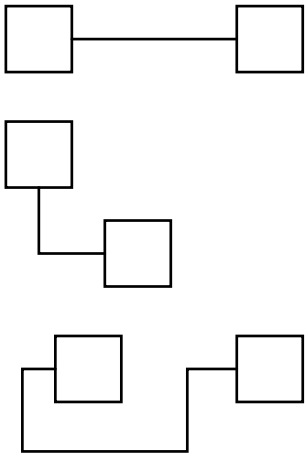

Figura 3.3: Ejemplos de conexiones

al nodo de los elementos, si un elemento se elimina, todas las conexiones a este elemento se eliminan, si el elemento se mueve o rota, las conexiones se mueven con él.

Si se accede al menú contextual con el botón derecho del mouse, aparecen las opciones borrar, plotear y ver número de nodo, el número de nodo aparece en la ventana de comandos de Matlab; inicialmente las conexiones no tienen número de nodo asignado, el número del nodo se asigna al inicio de las simulaciones.

## 3.6. Exportar datos

SASP permite exportar datos a diferentes formatos, gnuplot, matlab, excel y spice. Dependiendo del formato de los datos el archivo tendrá un nombre diferente, el archivo de matlab será 'resultados.mat', el de excel será 'resultados.xls', el de gnuplot será 'resultados.data' y si es un archivo de datos para el post-procesador de spice se llamará 'resultados.raw'. Todos los archivos se guardan dentro de la carpeta 'resultados'. La funcionalidad de exportar está es-

crita sobre exportadores genéricos, estos exportadores tienen ayuda e incluyen un ejemplo en el encabezado de cada función de manera que se pueden usar en otros proyectos; los nombres de las funciones empiezan con el nombre *export* y su ayuda se puede leer con el comando help, por ejemplo 'help export_spice'.

Cuando se exporta para gnuplot, los datos se escriben en letras, son separados por espacios y cada campo tiene una longitud de 15 caracteres con el proposito de organizarlos en columnas. Si se exportan los vectores [1 2 3 4] y [5 6 7 8] el archivo de gnuplot se veria:

```
1               5
2               6
3               7
4               8
```

Esto constituye únicamente los datos, el script para hacer la gráfica debe ser hecho a mano.

El formato de Excel es idéntico al formato de gnuplot, con la excepción que es separado con tabs en lugar de espacios; Excel y Open Office Calc pueden importar estos datos sin ningún problema.

Cuando se exporta en matfile, las tensiones se agrupan en una celda llamada vexport, el tiempo en una variable por separado, y un vector con los números de los nodos que se exportaron.

En la exportación en el formato spice se utiliza el nuevo formato ASCII como se describe en la ayuda de sconvert, un programa para convertir archivos entre los formatos de spice, la idea de estos datos es que se puedan analizar con post-procesadores de spice, como nutmeg. La parte de la ayuda concerniente al

formato ASCII es la que se muestra a continuación.

```
Ascii:

        Title: Title Card String
        Date: Date
        [ Plotname: Plot Name
          Flags: complex or real
          No. Variables: numoutputs
          No. Points: numpoints
          Command: nutmeg command
          Variables:   0 varname1 typename1
                  1 varname2 typename2
                  etc...
          Values:
            0     n    n    n    n    ...
            1     n    n    n    n    ...
            And so forth...
        ] repeated one or more times

    If one of the flags is complex, the points look like r,i where r and i
    are floating point (in %e format). Otherwise they are in %e format.
    Only one of real and complex should appear.

    The lines are guaranteed to be less than 80 columns wide (unless the
    plot title or variable names are very long), so this format is safe
```

```
to mail between systems like CMS.

Any number of Command: lines may appear between the No. Points:
and the Variables: lines, and whenever the plot is loaded into
nutmeg they will be executed.
```

# Capítulo 4

# Simulación

El programa tiene tres métodos de simulación, tiempo, frecuencia e híbrido. Cada método de simulación tiene sus modelos separados en diferentes carpetas. Cada modelo está ligado a un método de simulación y cada modelo puede ser simulado por separado usando el método que le corresponde, esto permite tener varios sistemas o componentes en el área de trabajo y poderlos simular por separado con diferentes parámetros para efectuar comparaciones como por ejemplo de sensibilidades a parámetros.

Cuando se hace una simulación, está debe tener un dueño, el dueño en el caso de simular todo el circuito, es la figura principal, pero si se simula un componente del circuito por separado usando el menú contextual, el dueño de la simulación es el componente, la diferenciación de los dueños de las simulaciones está ligado directamente a las propiedades de simulación. Si el elemento no tiene subcomponentes activos, el programa dará la advertencia y no simulará el elemento o graficará una tensión cero.

Hay que notar la diferencia entre las propiedades de simulación; si se acceden

las propiedades de simulación utilizando el icono de la barra de herramientas, estas propiedades serán las propiedades de simulación del área de trabajo, si se acceden las propiedades de simulación con el menú contextual de un elemento (click derecho), estas propiedades serán las propiedades de simulación cuando este elemento es el dueño de la simulación, es decir, cuando se simula este elemento por separado; lo mismo ocurre con el botón plotear y el menú plotear.

Los métodos de tiempo y frecuencia tienen parámetros para inyección de componentes, esto facilita la simulación de circuitos híbridos y más específicamente, la mezcla de métodos de simulación.

## 4.1. Simulación en Tiempo

La simulación en tiempo itera sobre los componentes del dueño de la simulación, revisa si hay elementos externos a adicionar , (como en el caso de una simulación híbrida) si los hay, los agrega dejando el elemento "huérfano", esto se hace pasando un valor de -1 a la función que agrega los elementos.

Las primitivas de tiempo son la fuente de corriente y la resistencia así que todos los componentes deben poderse reducir a estos modelos.

Cuando se agregan los elementos al sistema, estos se expanden utilizando sus subcomponentes, de manera que al final quede un gran circuito con solo fuentes de corriente y resistencias. Mientras se agregan los elementos, se hace un árbol de los componentes del circuito, este árbol sirve para relacionar cual componente contiene a cual en el circuito final, y para saber qué elementos se encuentran en el mismo nivel, es decir, cuales componentes están presentes en un mismo modelo. Cuando se deja un elemento huérfano no tiene ningún nexo

con el resto de elementos dentro del circuito, es decir, es independiente.

Después de solucionar las ecuaciones se comprueban los puntos de operación de los elementos en cada iteración, esto es importante para los elementos no lineales, especialmente para elementos como el diodo, debido a que en su curva de tensión-corriente hay un cambio brusco, un error en el punto de operación puede introducir errores grandes en la corriente consumida por el elemento. La comprobación de los puntos de operación se hace por el método del punto fijo y tiene un máximo de iteraciones definida en el archivo, normalmente un valor de 200 es más que suficiente y solo se produce un número grande de iteraciones sobre los cambios de pendiente de los elementos definidos por segmentos. Las Figuras 4.1 y 4.2 muestran el diagrama de flujo de la simulación en tiempo.

## 4.2. Simulación en Frecuencia

Al igual que en tiempo, las primitivas en frecuencia son la fuente de corriente y la resistencia. Antes de simular el sistema, se tiene que proveer la frecuencia a la cual se va a simular el sistema. Los pasos para la simulación en frecuencia se muestra en la figura 4.3 y corresponden a una simulación DC como se explicó en la sección 1.4.3. Se expande el circuito, se forma la Ybarra, se forma el vector de fuentes de corriente con las fuentes que están a la misma frecuencia de la simulación y se soluciona la ecuación matricial 4.1

$$V = Y_{barra}^{-1} I \tag{4.1}$$

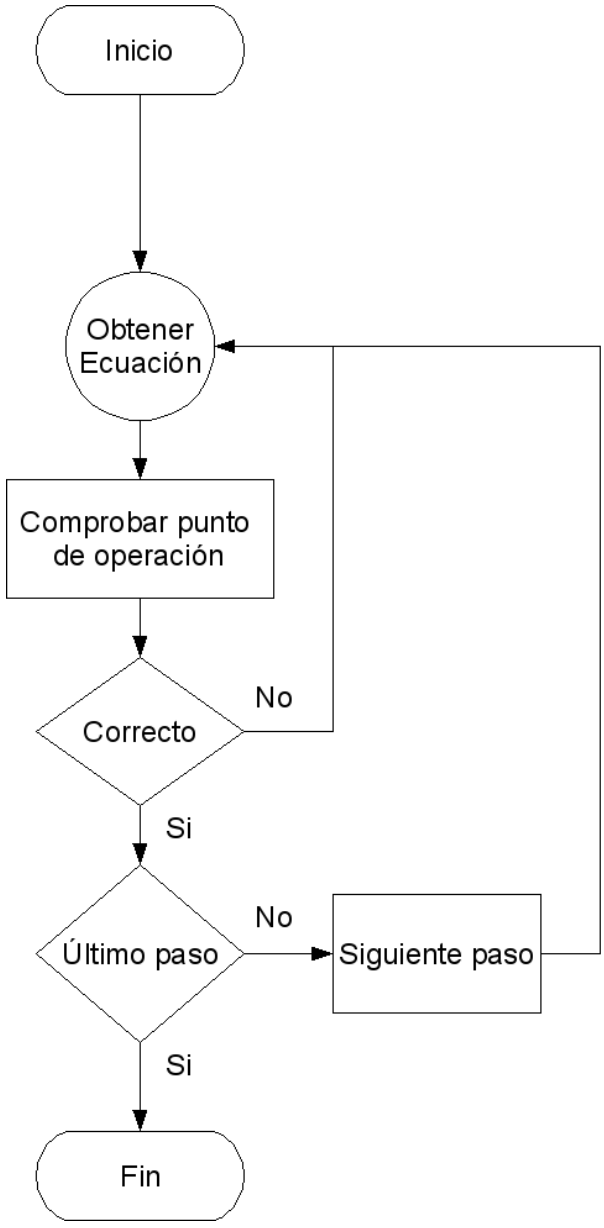


Figura 4.1: Esquema de la simulación en tiempo

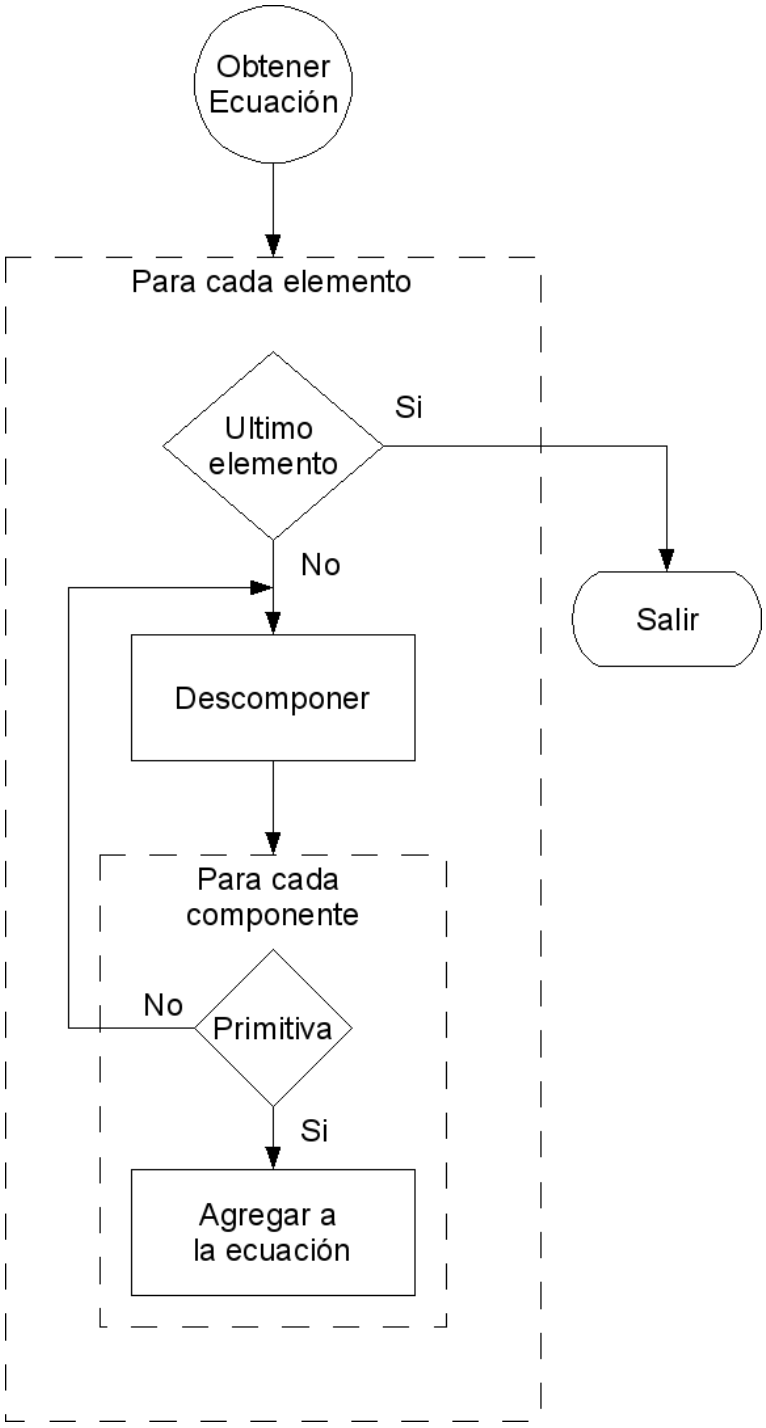


Figura 4.2: Esquema de la simulación en tiempo

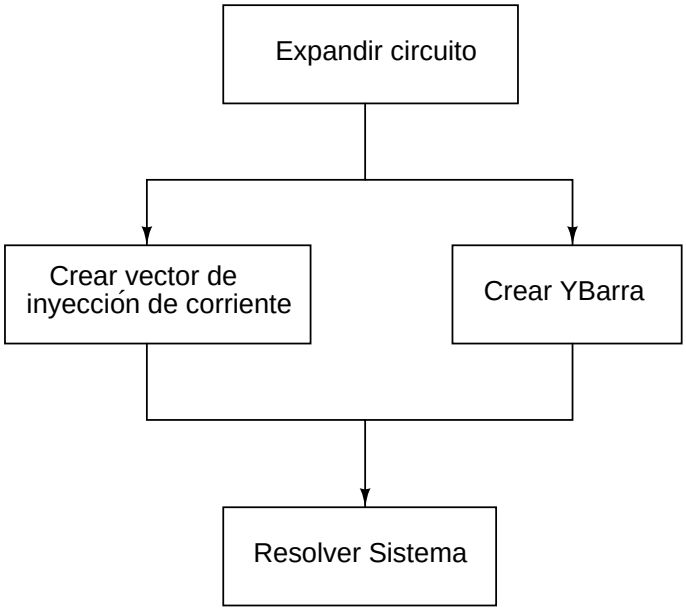


Figura 4.3: Esquema de la simulación en frecuencia

## 4.3. Simulación Híbrida

La simulación híbrida se basa en las simulaciones de tiempo y de frecuencia. El sistema lineal se simula en el dominio de la frecuencia, que es una simulación más rápida y que consume menos recursos que la simulación en el dominio del tiempo, y las cargas, se simulan en el dominio del tiempo, ya que se considera que son no lineales. La unión de los sistemas se hace por medio de inyección de corrientes. En la figura 4.4 se muestra el esquema de la simulación híbrida.

Cuando se simulan las cargas, el programa introduce amperímetros en los terminales de la carga, con el propósito de medir la corriente, como lo indica la Figura 4.5. Aunque la introducción de amperímetros que en realidad son resistencias de valor muy bajo, puede cambiar los resultados esperados de la simulación, esta solución es computacionalmente económica. Es similar a la

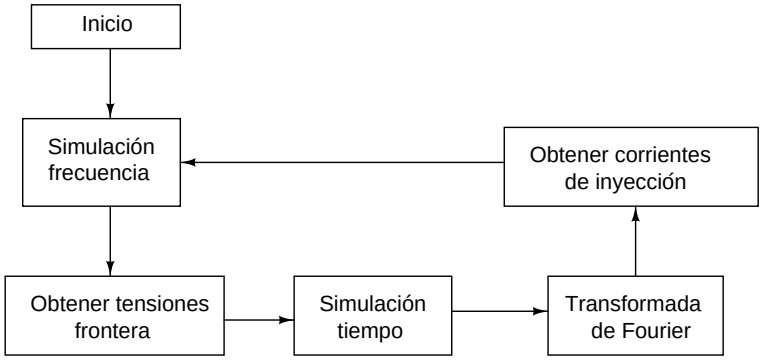


Figura 4.4: Esquema de la simulación híbrida

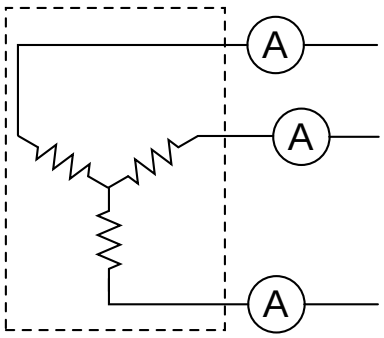


Figura 4.5: Carga con amperímetros en los terminales

aproximación de SPICE el cual no calcula corrientes directamente; es necesario introducir resistencias y medir la caída de tensión sobre ellas para encontrar la corriente.

En la inyección de tiempo a frecuencia, se analiza el espectro de corriente absorbida por la carga, se identifica la componente de mayor magnitud y se descarta cualquier componente con una magnitud menor al 1 % de la máxíma componente; de las componentes que quedan, se inyecta un número máximo al sistema que puede ser alterado en las propiedades de simulación.

Para la inyección de frecuencia a tiempo se obtienen los equivalentes Norton

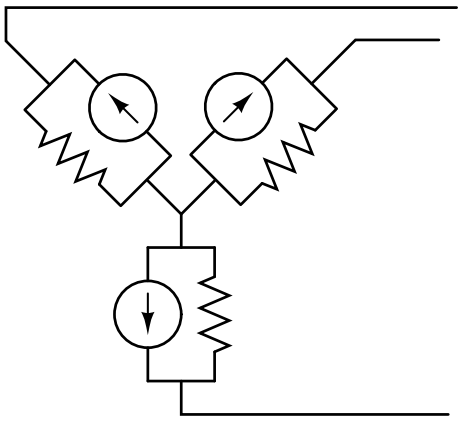

Figura 4.6: Equivalente Norton de un sistema trifásico

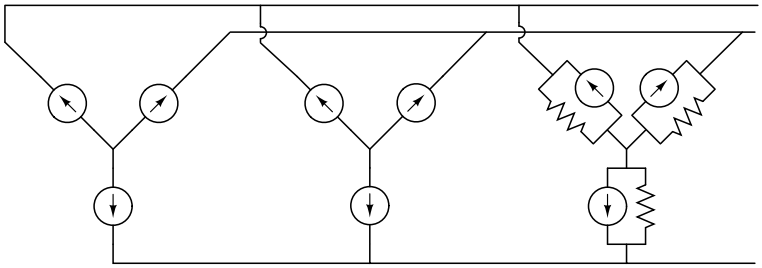

Figura 4.7: Equivalente de un sistema trifásico con 3 frecuencias importantes

del sistema a tierra a las diferentes frecuencias, en la Figura 4.6 se puede ver el equivalente Norton de un sistema trifásico. Ya que las cargas son no lineales, no se puede aplicar superposición de respuestas a diferentes frecuencias, y se hace necesario encontrar un solo equivalente del sistema a cualquier frecuencia como lo muestra la figura 4.7.

Para hallar el equivalente único del sistema se asume que la impedancia del sistema es muy baja, mucho más baja que la de la carga y se utiliza la impedancia a frecuencia fundamental como la impedancia del sistema, teniendo esta impedancia, se transforman las fuentes de corriente de los equivalentes Norton utilizando la ecuación 4.2, para mantener la tensión en los bornes de la

carga.

$$I_n = I_o \frac{Z_o}{Z_n} \tag{4.2}$$

Donde $I_n$ es la nueva fuente de corriente, $I_o$ la antigua fuente de corriente, $Z_o$ es la impedancia del equivalente Norton a esa frecuencia y $Z_n$ es la impedancia del sistema a la frecuencia fundamental.

Si $w_s$ es la frecuencia de muestreo y $w_M$ es la frecuencia máxima de las componentes de la señal, se puede reconstruir la señal si $w_s > 2w_M$, este, es el teorema de muestreo, y de él se concluye que si se está interesado en alguna componente de frecuencia, la frecuencia de muestreo debe ser mayor a dos veces la frecuencia de interés.

## 4.4. Pruebas y resultados obtenidos

Para cada método de simulación existe un banco de pruebas conformado por varios circuitos, desde simples hasta complejos; ciertos circuitos tienen propósitos específicos como comprobar la propagación de parámetros, revisar inestabilidades numéricas, etc. Los circuitos de los bancos de prueba tienen un nombre que explican a que banco de prueba pertenecen y que número de circuito es, los circuitos cuyo nombre empieza con la palabra 'test' son circuitos de prueba de frecuencia, los que empiezan con la palabra 'ttest' son circuitos de prueba de tiempo, y finalmente los circuitos que empiezan con el nombre 'testh' son circuitos de prueba híbridos. Existen cuatro circuitos de prueba en frecuencia, quince en tiempo y cuatro híbridos. Los bancos de prueba están totalmente documentados y serán entregados como información complementa-

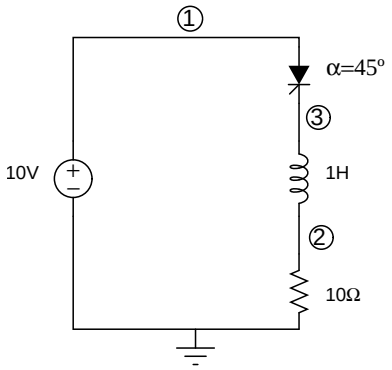


Figura 4.8: Circuito de prueba 15

ria en formato digital, los circuitos de prueba en tiempo tienen el archivo de entrada para ser simulados en gnucap, y los scripts para graficar los resultados en gnuplot; los circuitos de frecuencia e híbrido no tienen estos archivos pues gnucap no puede hacer simulaciones en frecuencia o simulaciones usando el método híbrido.

### 4.4.1. Dominio del tiempo

Un ejemplo de simulación en el dominio del tiempo se puede ver el circuito de la Figura 4.8, los números encerrados en circulos corresponden a los números de los nodos. En este circuito la corriente que pasa por la bobina depende solamente de la corriente anterior y este modelo de la bobina sumado al modelo simplificado del SRC, introduce picos de tensión en el cambio de conducción a no-conducción del SCR, como muestran las figuras 4.9, 4.10.

Los resultados obtenidos en las simulaciones muestran que en SASP los picos de tensión persisten y en la simulación de gnucap, los picos de tensión parecen desaparecen, sin embargo las figuras 4.11, 4.12 muestran un acercamiento a

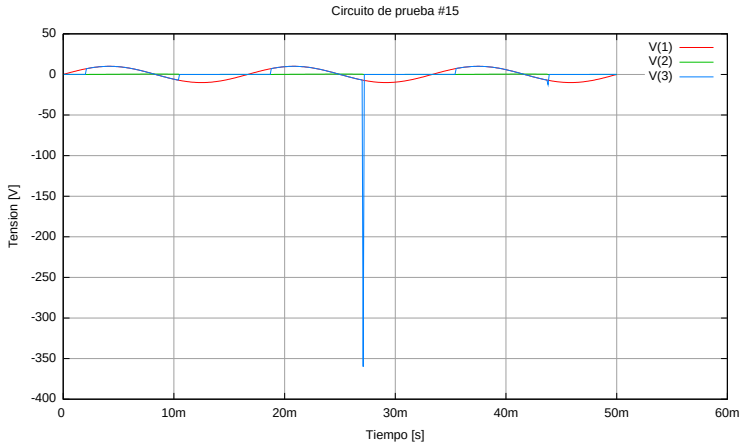


Figura 4.9: Resultados en Gnucap del circuito de prueba 15

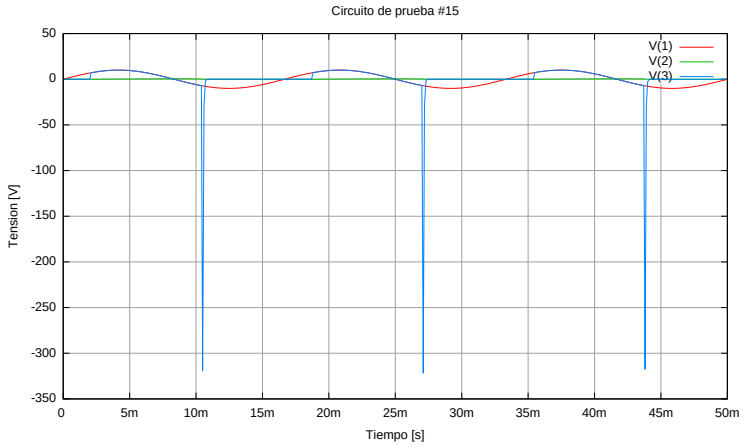


Figura 4.10: Resultados en SASP del circuito de prueba 15

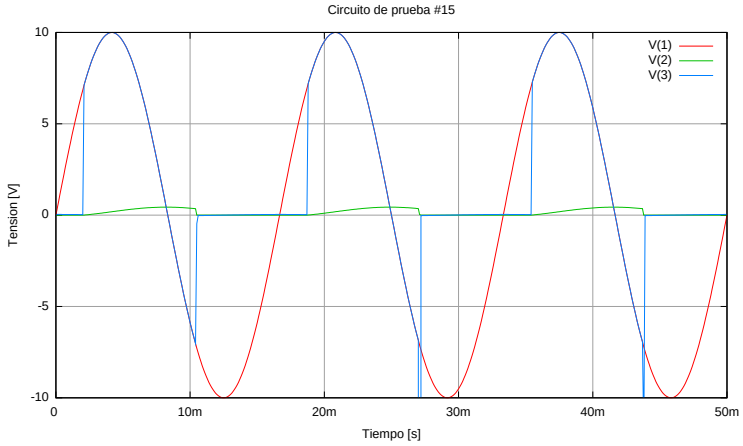


Figura 4.11: Acercamiento: Resultados en Gnucap del circuito de prueba 15

| # Simulación | 1 | 2 | 3 | 4 | 5 | **Promedio** |
|---|---|---|---|---|---|---|
| SASP | 100.859 | 101.060 | 101.197 | 101.347 | 101.39 | **101.17** |
| Gnucap | 0.064 | 0.069 | 0.062 | 0.066 | 0.064 | **0.065** |

Tabla 4.1: Circuito 15, comparación de tiempos entre SASP y gnucap

las respuestas, y se puede observar que los picos de tensión persisten en la simulación de gnucap, pero no se pueden ver en la gráfica original debido al control de paso.

Los tiempos de simulación son grandes comparados con otros simuladores, los circuitos de prueba escogidos para las comparaciones de tiempo fueron los 13 y 15 del grupo de pruebas de simulación en tiempo, que se muestran en las figuras 4.8 y 4.13. Las tablas 4.1 y 4.2 muestran los resultados obtenidos. Los tiempos de Matlab se tomaron con el par de comandos tic-toc y los tiempos para gnucap se tomaron con el comando time de linux.

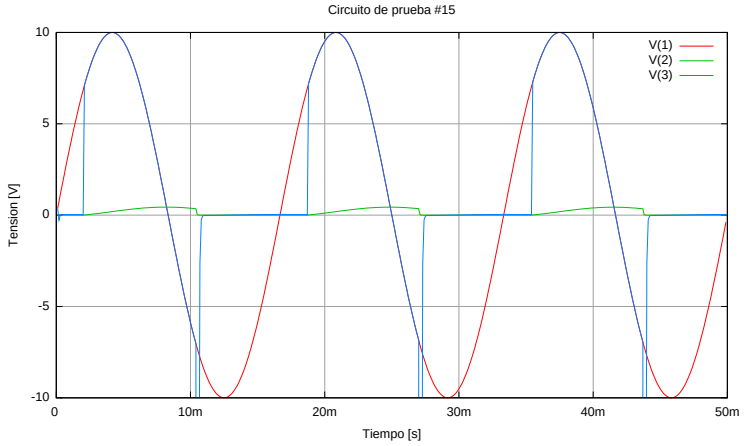


Figura 4.12: Acercamiento: Resultados en SASP del circuito de prueba 15

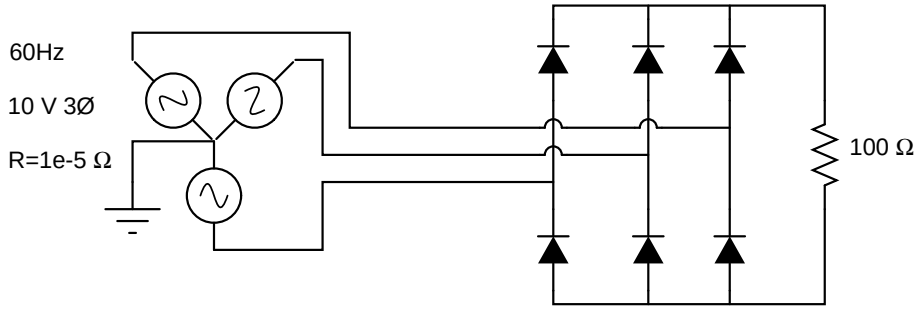


Figura 4.13: Circuito de prueba 13

| #Simulación | 1 | 2 | 3 | 4 | 5 | **Promedio** |
|---|---|---|---|---|---|---|
| SASP | 260.657 | 261.001 | 261.032 | 260.956 | 261.094 | **260.95** |
| Gnucap | 0.092 | 0.090 | 0.091 | 0.090 | 0.090 | **0.0906** |

Tabla 4.2: Circuito 13, comparación de tiempos entre SASP y gnucap

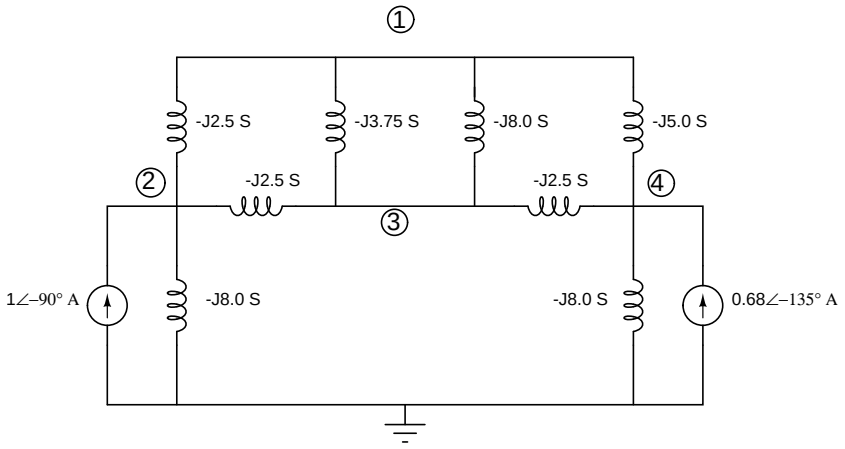


Figura 4.14: Circuito de prueba 2

### 4.4.2. Dominio de la frecuencia

Los resultados de las simulaciones en frecuencia se comprobaron haciendo los cálculos manualmente. El circuito que se muestra en la Figura 4.14, el número dos del banco de pruebas, fue extractado del capítulo 7 del libro Análisis de sistemas de potencia [6]. Los resultados se muestran en la figura 4.15

### 4.4.3. Método Híbrido

Las circuitos híbridos que se introdujeron en el grupo de prueba se diseñaron especialmente para comparar tiempos y analizar componentes armónicos de sistemas. Se simuló el circuito de la Figura 4.13 en tiempo y usando el método híbrido, los tiempos obtenidos se muestran en la Tabla 4.3, y se puede ver que los tiempos de simulación usando el método híbrido son aproximadamente 3 veces mayores a los de tiempo, esto se puede explicar debido la naturaleza de la simulación híbrida, los ahorros de tiempo se obtienen simulando el sistema en frecuencia, de manera que si el sistema (como en este ejemplo) es una parte

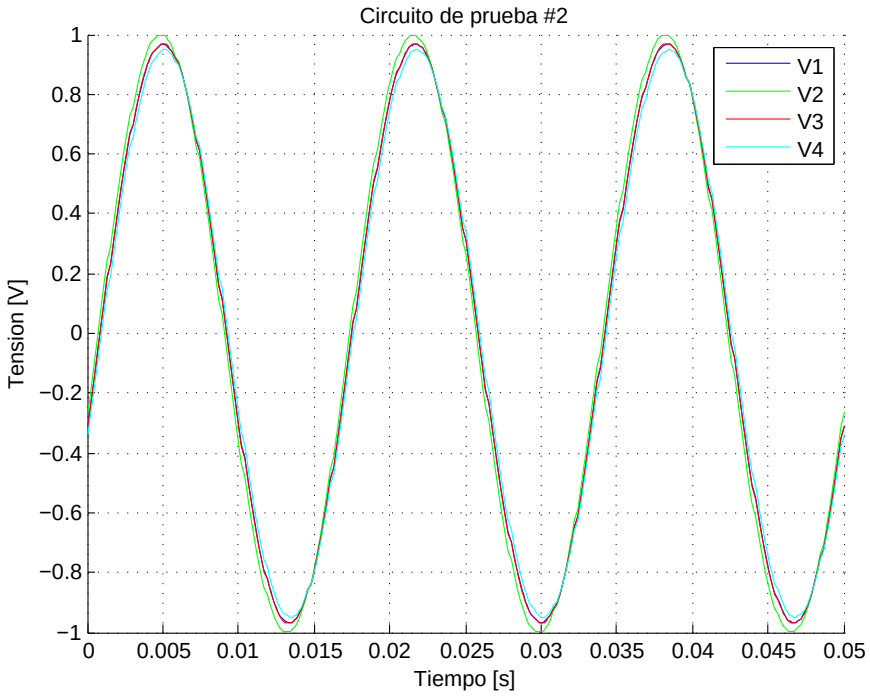


Figura 4.15: Resultados circuito de prueba 2

| Tiempo [s] | Híbrido[s] |
|---|---|
| 10.56 | 36.76 |
| 10.34 | 37.20 |
| 9.62 | 36.81 |
| 10.13 | 37.13 |
| 9.47 | 36.87 |

Tabla 4.3: Circuito 13, comparación de tiempos entre tiempo e híbrido (SASP)

| Tiempo [s] | Híbrido[s] |
|---|---|
| 268.2 | 33.68 |
| 268.52 | 33.83 |
| 267.46 | 34.26 |
| 267.92 | 33.45 |
| 268.6 | 33.93 |

Tabla 4.4: Circuito 19, comparación de tiempos entre tiempo e híbrido (SASP)

pequeña del circuito, el ahorro de tiempo no es significativo, y por el contrario el costo del análisis armónico y las inyecciones, hacen que los tiempos sean mayores, esto es cierto para circuitos con sistemas lineales pequeños y un gran número de cargas.

Para probar que el método híbrido puede presentar una ventaja en tiempos de simulación, se diseñó un circuito con un sistema lineal grande (comparativamente a la cargas) y dos cargas simples, lo opuesto al ejemplo anterior; este circuito se muestra en el figura 4.16. Esta vez la simulación híbrida tiene una gran ventaja sobre la simulación en tiempo, la tabla 4.4 muestra los tiempos tomados para cinco simulaciones de cada circuito. Podemos observar de la tabla que los tiempo híbridos son casi 8 veces menores que los tiempos de la simulación en tiempo.

Se simuló el circuito de la figura 4.13 tal como se ve en la figura, y con una resistencia de desbalanceo con el objeto de analizar las componentes armónicas.

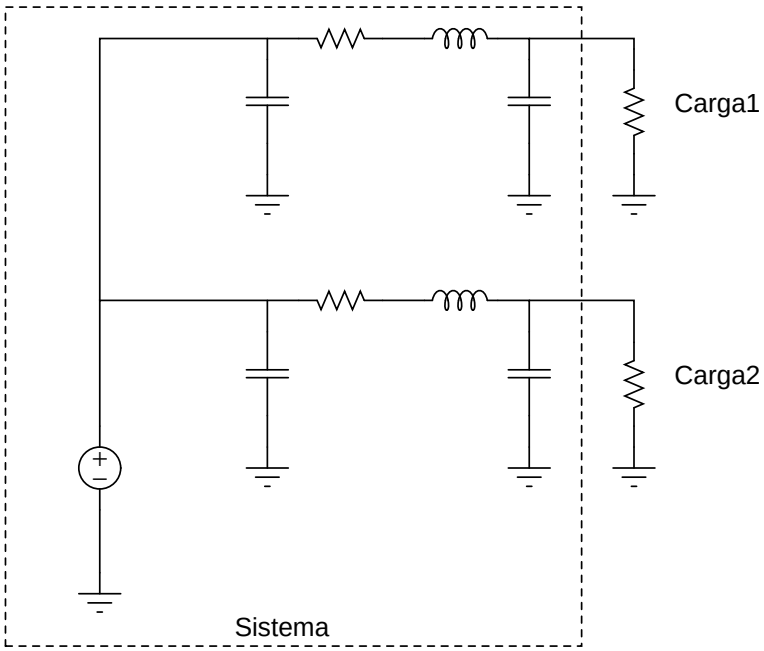


Figura 4.16: Circuito de prueba 19 en tiempo

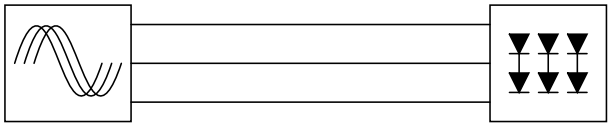

Figura 4.17: Esquema circuito de prueba 4

Los figuras 4.17 y 4.18 muestra los esquemas de los circuitos con y sin balanceo, tal como se ve en SASP. La forma de onda del circuito de prueba 4 que se ve en la figura 4.19 no presenta distorsión armónica, sin embargo el circuito de prueba 6 converge en la cuarta iteración y presenta distorsión armónica introducida por el desbalance de cargas, su espectro se puede ver en la figura 4.21, como los colores de la gráfica tienen transparencia, los colores diferentes a rojo, azul y verde, es superposición de las columnas, esto con el fin de poder ver todas las componentes sin que una oculte a otra.

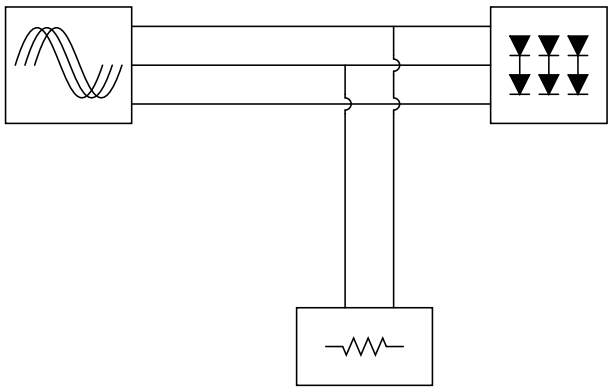

Figura 4.18: Esquema circuito de prueba 6

Los tensiones V(4) y V(5) son las tensiones en los nodos a los cuales se encuentra conectada la carga del rectificador trifásico de onda completa.

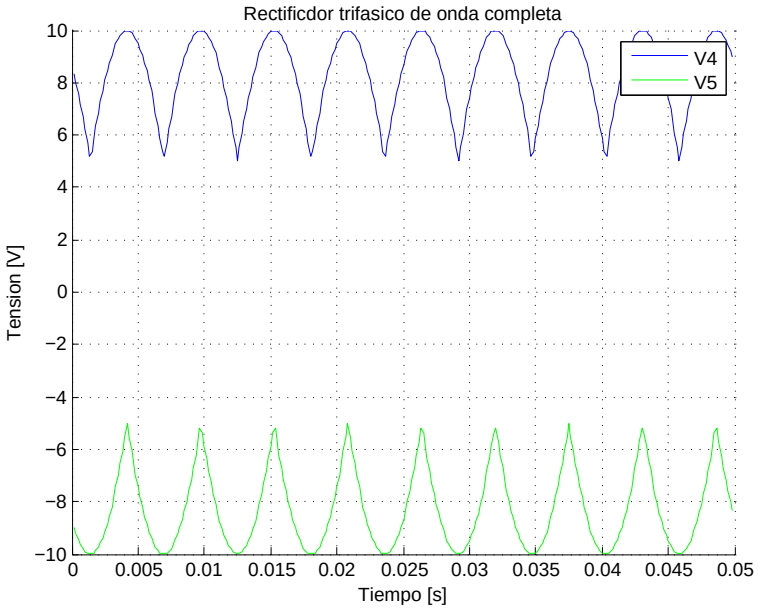


Figura 4.19: Resultados circuito de prueba 4

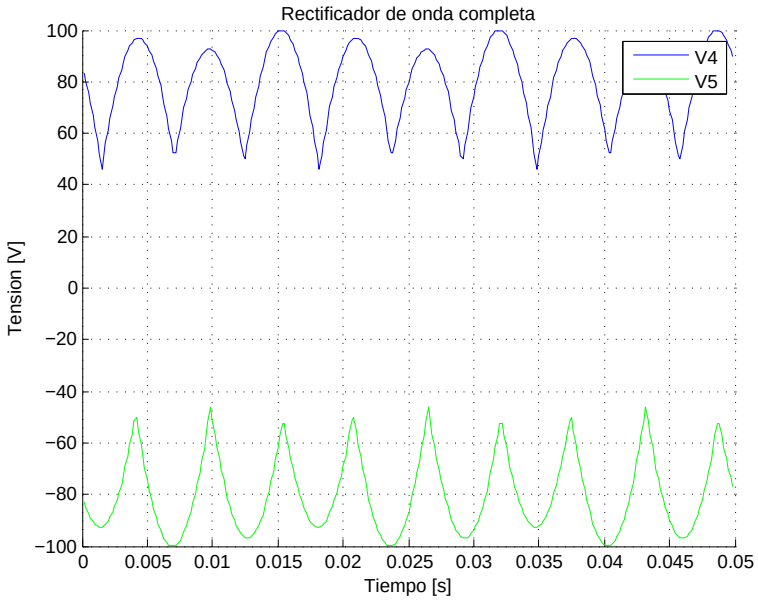


Figura 4.20: Resultados circuito de prueba 6

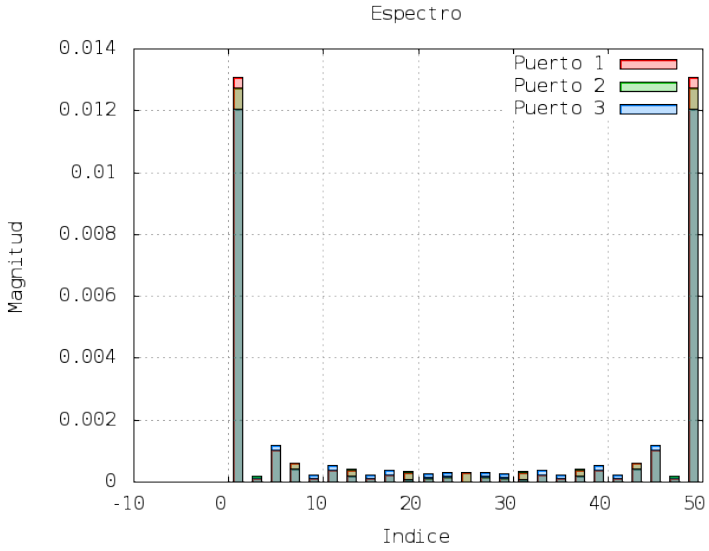


Figura 4.21: Transformada de Fourier de la corriente consumida por el circuito de prueba 6

# Capítulo 5

# Observaciones y conclusiones

## 5.1. Observaciones

Matlab no está diseñado para hacer interfaces gráficas muy elaboradas, un buen ejemplo de funcionalidades faltantes son por ejemplo, la diferenciación del click derecho de un click izquierdo, también falta una forma directa de acceder a los métodos de los objetos, actualmente se llama una función y se pasa como argumento una instancia del objeto.

Las funciones de dibujo están muy limitadas, matlab está enfocado hacia gráficas de funciones, no gráficas arbitrarias, dependiendo del graficador (OpenGL, zbuffer, painters) que se escoja el dibujo del circuito se ve diferente en pantalla.

Matlab permite hacer software que corra en linux y en windows. Aunque el desarrollo empezó en windows, terminó en linux, gracias a las herramientas presentes en linux como grep, un buscador de cadenas, una plataforma para LaTeX mejor integrada, el poder cerrar Matlab si se congelaba, etc.

En el método híbrido la estabilidad depende fuertemente del paso de tiempo (dt) usado, la selección de un dt muy grande comparado con el periodo de la señal introduce errores en la transformada de Fourier, una frecuencia fundamental de 60 Hz puede ser identificada como de 59.999Hz, esto es un problema ya que al inyectar la fuente de corriente que representa la corriente consumida por la carga, obtenemos un circuito con dos fuentes de magnitud alta, una a 60Hz y otra a 59.999 que deberían estar a 60Hz e interactuar entre ellas en la simulación del circuito lineal a 60Hz, para evitar esto SASP redondea las frecuencias hasta el tercer decimal.

La cantidad de simulaciones necesarias para el método híbrido es proporcional al número de frecuencias a tener en cuenta y al número de cargas que se tienen, como un ejemplo, si el sistema posee 10 cargas con 6 frecuencias importantes, la cantidad de simulaciones en frecuencia sería de 60 para cada iteración del método híbrido.

## 5.2. Conclusiones

Las interfaces gráficas se elaboran usando la herramienta guide de Matlab, esta herramienta está implementada sobre java y tiene serios problemas de estabilidad tanto en windows como en linux, en windows puede congelar todo el sistema operativo, en linux puede congelar el servidor X e incluso parece tener una fuga de memoria que con el tiempo puede congelar ambos sistemas operativos. Matlab tiende a guardar el estado anterior de una figura, de manera que si se usa creación de objetos en tiempo de ejecución se termina con una gran cantidad de objetos en memoria, la única solución para esto es borrarlos

antes del final de la ejecución de la figura, no hay documentación sobre este aspecto.

Matlab es un lenguaje interpretado, los lenguajes interpretados se carácterizan por ser lentos, Matlab intenta solucionar esto con optimizaciones JIT (just in time) y precompilación, pero el SASP tiene características como aceptar expresiones matemáticas para los valores de los elementos, que impiden la optimización, y debido al bajo número de primitivas, hay que leer muchos datos de disco y organizarlos, así que el SASP es realmente lento en comparación con otros simuladores, es el precio de la flexibilidad.

En la simulación en computador, siempre aparecen errores numéricos, ya sea por aproximaciones, parámetros escogidos o por los modelos usados. En SASP actualmente la mayor fuente de errores son los modelos. El diodo por ejemplo es un conjunto de resistencias, así que los resultados de otros simuladores con modelos más exactos siempre dará una diferencia de al menos 0.7 V en la tensión de los diodos. El modelo de compañia de la bobina depende solamente de la corriente anterior, lo que forza una corriente muy grande en el siguiente paso aunque no exista un camino de conducción, esto crea picos de tensión en las simulaciones con elementos de comuntación, por ejemplo tiristores, el propósito del proyecto no era desarrollar modelos de los componentes, de hecho, esto es bastante difícil e incluso en algunos elementos depende del tipo de simulación que se desee hacer, como el de los diodos que pueden ser de pequeña o gran señal, de altas o bajas frecuencias, etc.

Los ahorros de tiempo en el método híbrido son debidos a la simulación del sistema en frecuencia, debido a esto, el método es recomdable para sistemas grandes con un número pequeño de cargas, donde su rapidez es notoria, por

el contrario en sistemas pequeños con un gran número de cargas, el ahorro de tiempos puede no ser significativo o incluso la simulación puede ser mas lenta que una simulación en tiempo.

## 5.3. Trabajo Futuro

El programa es una prueba de concepto, el trabajo futuro sobre éste debería ir encaminado a la inclusión de modelos, inclusión de primitivas que aceleren el tiempo de simulación y optimización de los algoritmos, Matlab tiene sus propios métodos de optimización pero está basado en suposiciones. Alguien con un buen conocimiento sobre el funcionamiento interno de Matlab y podría hacer mejoras substanciales de velocidad al código.

Arreglar el algoritmo para encontrar las corrientes en los elementos, el que existe actualmente es empírico, la comprobación del punto de operación se hace para todos los elementos, esto se debería hacer solo para elementos definidos a pedazos. La iteración sobre los puntos de operación se hace usando el método del punto fijo, en circuitos con alta conmutación esto puede ser lento, la implementación de otro método para encontrar el punto de operación podría ser útil para estos casos.

En cuanto a trabajos extras generales, se pueden programar más exportadores para poder utlizar los datos con otros software, se puede incluir circuitos típicos, de control, potencia y electrónicos para que la creación de sistemas complejos sea más fácil. Agregar opciones para introducir modelos con otros datos por ejemplo potencia y tensión.

# Apéndice A

# Manual de usuario

Este es un breve manual de usuario encaminado a dar las nociones básicas sobre la simulación en SASP, además de este material, existen vídeos que serán incluidos con el software que muestran como hacer circuitos y correr simulaciones en SASP. Para correr SASP este debe estar correctamente instalado. La instalación es muy directa, se copia la carpeta "SASP" que contiene el software y se cambia la ruta de trabajo de Matlab con el comando *cd*.

## A.1. Crear un nuevo circuito

Para crear un nuevo circuito se presiona el botón de 'nuevo', el número uno que se ve en la Figura A.1, este abrirá un diálogo para escoger que tipo de circuito se desea crear, tal como se muestra en la Figura A.2.

Cuando se ha escogido el tipo de circuito, aparecen los elementos en la barra de iconos de la izquierda de acuerdo al archivo de configuración del método de simulación, a partir de aquí, se puede empezar a construir el circuito.

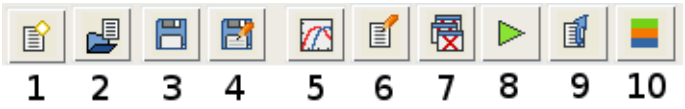


Figura A.1: Botones de herramientas


Figura A.2: Diálogo nuevo

## A.2. Construcción del circuito

Para agregar un elemento al circuito hay que hacer click sobre el ícono del elemento en los botones de modelos en la parte izquierda de la ventana, el mouse se verá ahora como una cruz. Para ubicar el elemento basta con hacer click sobre el área de trabajo.

Las conexiones se crean con el menú contextual de los elementos, se hace click derecho sobre el elemento y en la opción cablear, el cursor cambia a una cruz, se hace click sobre las dos terminales que se desean conectar y aparece la conexión, la Figura A.3 muestra una conexión entre dos elementos.

Para introducir valores en los componentes del sistema se hace click derecho sobre el elemento gráfico y en el menú contextual se escoge 'valores', despues de

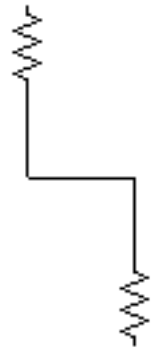

Figura A.3: Conexión entre elementos como se ve en el software

introducir los parámetros de todos los elementos, el siguiente paso es configurar la simulación del circuito.

## A.3. Simulación y visualización de resultados

Para correr una simulación se necesita configurar sus parámetros. Para acceder a los parámetros de simulación del área de trabajo se hace click sobre el ícono seis de la Figura A.1. En el caso de una simulación en tiempo, sus parámetros son: Paso, Tiempo final, Titulo, Nodos Graficador y Grid. Solo los parámetros Paso y Tiempo Final son necesarios para correr la simulación los parámetros restantes son para la visualización de los resultados. La figura A.4 muestra el dialogo de propiedades de tiempo con valores de ejemplo.

Los diálogos de simulación en frecuencia e híbrida son similares al de tiempo y se acceden de la misma manera. Cuando se han definido los parámetros de simulación, se puede simular el circuito haciendo click sobre el botón ocho de la figura A.1.

Para visualizar los resultados se hace click sobre el botón cinco que se mues-

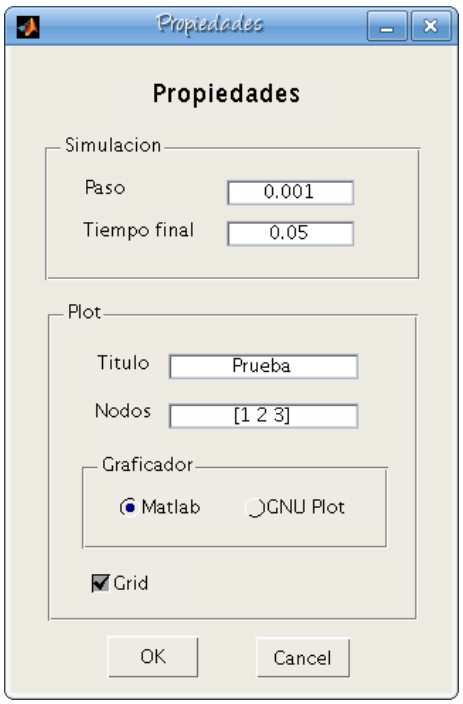


Figura A.4: Ejemplo propiedades de tiempo


Figura A.5: Diálogo exportar

| Tipo | Extensión |
|---|---|
| gnuplot | data |
| matfile | matn |
| excel | xls |
| spice | raw |

Tabla A.1: Extensión de los datos exportados

tra en la figura A.1. Adicionalmente existe la posibilidad de exportar los datos para ser visualizados, o analizados por software externo, esto se realiza haciendo click sobre el botón nueve de la figura A.1, el dialogo exportar de la figura A.5 se abrirá.

El archivo exportado se llama 'simulacion' y dependiendo del tipo de archivo que se escoja la extensión del archivo en disco cambia, la tabla A.1 relaciona el tipo de archivo con la extensión.

# Bibliografía


[1] Watson N.R. Wood A.R Arrillaga J., Smith B.C. *Power System Harmonic Analysis*. John Wiley & sons, 1998.

[2] Jordi Bonet Dalmau. *Análisis del régimen permanente y la estabilidad de circuitos no lineales con parámetros distribuidos mediante técnicas de tiempo discreto*. Tesis Doctoral, UPC España, 1999.

[3] Albert T. Davis. Gnucap: The gnu circuit analysis package user manual, Abril 2007.

[4] Hong Xu. *Transient sensitivity analysis in circuit simulation*. Tesis de Maestria, Technische universiteit Eindhoven, Noviembre 2004.

[5] Alan V. Oppenheim y Alan S. Willsky. *Señales y sistemas*. Pearson Educación, segunda edición, 1998.

[6] Jhon J. Grainger y William D. Stevenson Jr. *Análisis de sistemas de potencia*. Segunda edición, 1996.

[7] William Carvajal Carreño. *Armónicos: Método híbrido tiempo-frecuencia para el análisis de sistemas eléctricos con elementos no lineales*. Tesis de Maestria, Universidad Industrial de Santander, En ejecución.

[8] Albert T. Davis. *Implicit Mixed-Mode Simulation of VLSI Circuits*. Tesis Doctoral, University of Rochester, 1991.

[9] Jose R. Martí y Jiming Lin. Suppression of numerical oscillations in the EMTP. *IEEE transactions on Power systems*, Mayo 1989.

[10] Westinghouse Electric Corporation. *Electrical transmission and distribution Reference book*. Cuarta edición, 1950.