



REPÚBLICA BOLIVARIANA DE VENEZUELA
MINISTERIO DEL PODER POPULAR PARA LA
EDUCACIÓN SUPERIOR
INSTITUTO UNIVERSITARIO EXPERIMENTAL
DE TECNOLOGÍA DE LA VICTORIA
LA VICTORIA- ESTADO ARAGUA
Comisión Académica del Programa
Nacional de Formación de Electricidad



CÁLCULO NUMÉRICO

PARA

SISTEMAS ELÉCTRICOS

PRÁCTICA I: INTRODUCCIÓN A LA COMPUTACIÓN

Realizado por: Prof. DORIS E. SEVILLA
Revisado por: Prof. JESUS A. PEREZ

La Victoria Octubre de 2007

TEMARIO:

- I.** Definición de las computadoras y sus componentes.
- II.** Hardware y Software
- III.** Descripción y conexión de los dispositivos periféricos en una computadora.
- IV.** Componentes internos más importantes del computador.
- V.** Decreto presidencial relacionado al uso del software libre.

OBJETIVOS:

- 1.** Identificar los componentes de una computadora.
- 2.** Manejo del lenguaje básico usado con mayor frecuencia en la informática.
- 3.** Capacidad para conectar los componentes de una computadora a sus respectivos puertos.
- 4.** Identificar los componentes internos de un computador.
- 5.** Conocerá algunos los paquetes disponibles en el mercado.
- 6.** Conocerá el Decreto Presidencial referente al uso de Software Libre.
- 7.** Conocerá algunas las ventajas y desventaja ofrecidas por la Software Libre.

I. DEFINICIÓN DE LAS COMPUTADORAS Y SUS COMPONENTES:

Las Computadoras: son máquinas que son un dispositivo electrónico que acepta datos de entrada, los procesa, los almacena y los emite como salida para su interpretación. Se puede usar para redactar documentos, enviar correo electrónico y navegar la Internet. La computadora es parte de un sistema de computación.

Componentes de un sistema de computación (Figura No.1): Todas las computadoras, desde el más pequeño microsistema hasta los más complejos, están compuestas de tres componentes básicos.

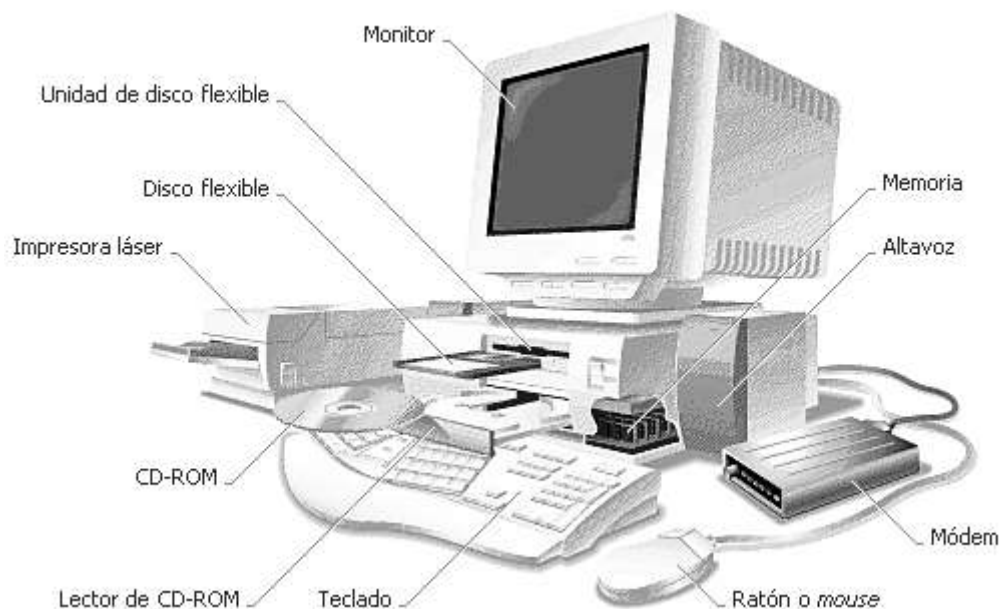


Figura No.1: Componentes de un sistema de computación

Estos componentes son:

1. Unidad Central de Proceso:

También llamada **CPU** o **UCP**, está formado por dos unidades principales:

La unidad de Control: representa el "corazón" de un computador, encargándose de controlar y coordinar toda la actividad del procesamiento de datos, incluyendo el control de todos los dispositivos de Entrada/Salida (en adelante E/S), coordinar la entrada y salida de datos e información de las diferentes memorias, determinar las direcciones de las operaciones aritméticas y lógicas, y seleccionar, interpretar y enviar a ejecutar las instrucciones de los programas.

La unidad aritmética y lógica: es la encargada de ejecutar todos los cálculos matemáticos (Suma, resta, multiplicación y división) y todas las comparaciones lógicas.

2. Unidades de Memoria:

Un computador personal posee básicamente dos tipos de memoria:

La Memoria Principal: dividida a su vez en memoria sólo de lectura (**ROM: Read Only Memory**), y la memoria que puede leerse, borrarse y actualizarse (**RAM: Random Access Memory**).

La ROM es el área de la memoria donde el fabricante de la computadora graba todos los datos e instrucciones necesarias para el funcionamiento del computador. El usuario tiene acceso a esta memoria para leerla pero no puede grabar ni cambiar absolutamente nada en ella. El contenido de esta memoria es permanente y con la ausencia del flujo electrónico no desaparece.

La RAM es el área de memoria principal disponible para satisfacer las necesidades de programación del usuario, es allí donde se guardan los datos y los programas a ejecutarse en un momento determinado. Esta memoria es volátil, significa que su contenido se pierde al apagarse el computador. Generalmente el tamaño de memoria de los computadores está determinado por la cantidad de memoria RAM que posea.

La Memoria Auxiliar: La capacidad de almacenamiento en memoria principal es limitada, situación diferente se presenta para la memoria auxiliar que es ilimitada, todo medio magnético, diskette, zip, cassette, cinta, disco, cd, etcétera, se considera como memoria auxiliar sirviendo este para guardar todos los datos y programas deseados.

3. Dispositivos de Entrada y /o Salida:

Una computadora debe tener al menos un dispositivo de entrada y otro de salida o uno con las dos funciones, los más habituales son las unidades de disco (el disquete, el CD-ROM y el DVD-ROM) o el módem que conectado a una línea telefónica y permite intercambiar información con otros computadores, por ejemplo, utilizando la red Internet.

Dispositivos de Entrada: son periféricos cuya función es la de reunir y traducir los datos de entrada a una forma que sea aceptable para la computadora. Los dispositivos de entrada más comunes son el teclado y el ratón o "mouse" (ver Figura 1)

Dispositivos de Salida son periféricos que representan, imprimen o transfieren los resultados del procesamiento, extrayéndolos de la Memoria Principal de la computadora. Entre los dispositivos de salida más utilizados se encuentran el Monitor o Pantalla, altavoces y la Impresora

II. HARDWARE Y SOFTWARE: Los componentes mencionados en la sección anterior constituyen el hardware de un sistema de computación, pero para que un computador funcione se necesita darle una serie de órdenes lógicas que permitan procesar datos, los software son los encargados de realizar esas tareas

Hardware: incluye todos los dispositivos eléctricos, electrónicos y mecánicos (que se pueden ver y tocar) que se utilizan para procesar los datos. La computadora es parte del hardware del sistema de computación.

Software o programas: es el conjunto de instrucciones electrónicas para controlar el hardware de la computadora.

Algunos Programas existen para que la Computadora los utilice como apoyo para el manejo de sus propias tareas y dispositivos.

Cuando se habla de Datos se refieren a los elementos crudos (materia prima) que la computadora puede manipular, para convertirlos en resultados o datos procesados, conocidos como información (producto terminado).

III. DESCRIPCIÓN Y CONEXIÓN DE LOS DISPOSITIVOS PERIFÉRICOS O COMPONENTES EXTERNOS DE UNA COMPUTADORA

Los Dispositivos Periféricos son componentes de hardware que acompañan a una computadora para incrementar su funcionalidad para introducir, extraer y almacenar datos. Están ubicados “alrededor de la máquina”. Estos son:

Gabinete de la computadora (CHASIS): es la caja de metal y plástico que aloja a los componentes principales.

Este tiene normalmente en la parte de adelante el botón para encender o apagar la computadora, ranuras para introducir disquete, CD-ROM y DVD-ROM, en la parte posterior todas las ranuras para realizar las conexiones de los dispositivos periféricos del computador.

Fuente de Alimentación: Como cualquier aparato electrónico, la computadora necesita energía eléctrica para su operación. Para tal efecto, tu equipo cuenta con un cable de corriente, con un extremo a conectar a la parte posterior del chasis, y el otro a la fuente de energía.



Figura No.2 Cable de alimentación

Monitor: que se parece a un televisor, es el componente en el que se visualizan texto e imágenes. Este requiere de dos cables: el cable de señal, que sale del monitor y termina en un conector tipo DB15, que se inserta en la entrada indicada del chasis. Es necesario asegurar los tornillos que tiene.



Figura No.3 Conector del monitor

El otro cable es de alimentación y va de la parte posterior de tu monitor, a una fuente de energía.

Teclado: le permite ingresar datos en la computadora. Se parece al teclado de una máquina de escribir. Este dispositivo requiere un sólo cable de señal que debe insertarse en la entrada o puerto indicado. Frecuentemente se encuentra

junto a la entrada del ratón y puedes diferenciarlos por sus símbolos y colores. Algunos teclados tienen un conector USB.



Figura No.4 Conectores de teclado

Ratón: también conocido como "mouse", es un dispositivo de mano que le permite seleccionar con un clic o trasladar objetos por la pantalla del monitor. Este dispositivo requiere de un cable de señal con un conector que debe ser insertado en la computadora en la entrada o puerto indicado.

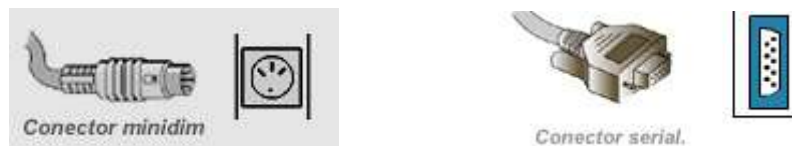


Figura No.5 Conectores de Ratón

Al igual que el teclado, el conector del ratón también puede ser de tipo USB o serial.

Impresora: es un dispositivo mecánico que genera una copia impresa de lo que aparece en el monitor. La impresora requiere de dos cables para su funcionamiento: el cable de señal, también llamado "paralelo" o "RS232" que tiene un conector DB25, el cual debe ser insertado en la entrada indicada del chasis, y el cable de alimentación, que se conecta a una fuente de energía.

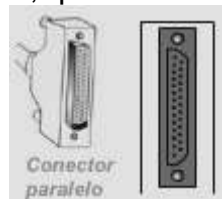


Figura No.6. Conector de la impresora

El cable de señal, en las impresoras actuales, también puede ser del tipo USB.

Bocinas Externas: Las bocinas externas requieren conectar el cable de tipo "plug" macho a la parte posterior del chasis, en la entrada que se indica.



Figura No.7 Conector de bocinas externas

Cabe aclarar que algunos equipos cuentan con bocinas internas, las cuales no requieren conexión.

Los Dispositivos Periféricos se instalan con base en las necesidades del usuario y sus disponibilidades económicas. Un Equipo Básico típico que se compre, incluye: Procesador, Memoria RAM, Disco Duro, Unidad de Disquete, Monitor, CD-ROM, Impresora, Teclado y Ratón. Cualquier Periférico adicional que se requiera "se paga aparte". Se podría sustituir el mouse con una "Esfera Rastreadora" o Track Ball. También es posible aumentar las capacidades de la computadora agregándole un Digitalizador o Escáner para capturar imágenes.

Para la conexión a Internet se requiere: Tarjeta de Red, Modem y una Cuenta de Servicio de Acceso a Internet a través de un Proveedor de Servicios Internet (ISP). De manera que las posibilidades de configuración del Equipo de Computación Básico, son muy amplias, dependiendo de cuanto dinero se esté dispuesto a invertir.

Ejemplo 1: Indique paso a paso como armar un computador dado los dispositivos periféricos (componentes externos) usados con mayor frecuencia:

1. Se debe elegir el lugar adecuado para instalar la computadora, que permita trabajar cómodamente y que resguarde la seguridad física del equipo.
2. Nunca forzar la conexión de un cable, si una parte no acopla, seguramente se esta poniendo en una posición incorrecta o en un lugar equivocado.
3. Se procede a conectar cada uno de los elementos del equipo, tal y como indica la siguiente figura 8:

Los componentes a instalar son: fuente de alimentación, monitor, teclado y el ratón.

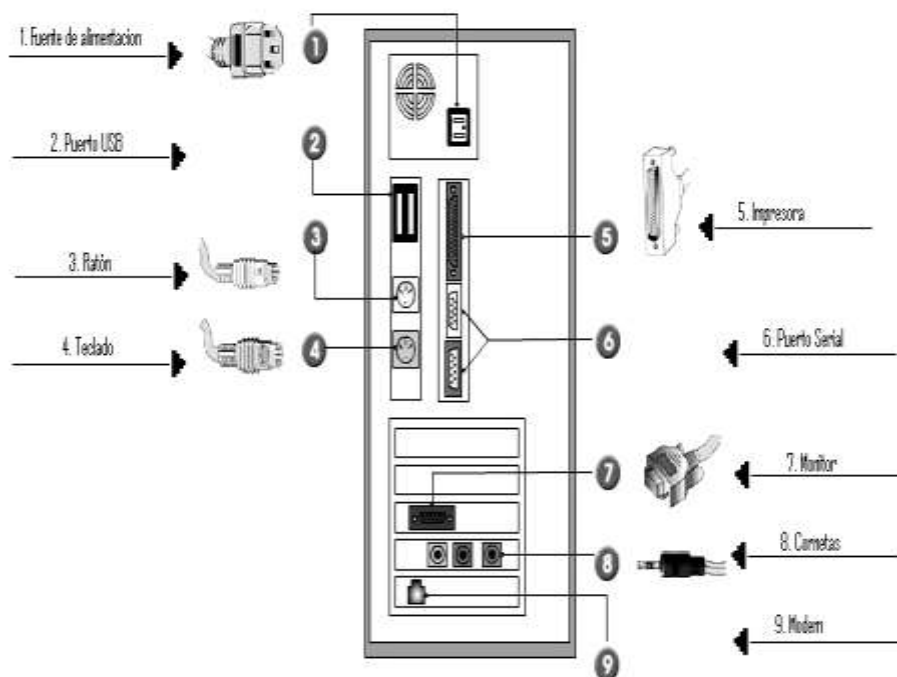


Figura No. 8: Conexión de los componentes externos al computador

IV. COMPONENTES INTERNOS MÁS IMPORTANTES DE UN COMPUTADOR:

Placa Base o Madre (Mainboard o Motherboard): Es uno de los elementos más importantes, a él se conectan todos los componentes del computador. Físicamente es una lámina fina fabricada con materiales sintéticos. Dicha lámina contiene circuitos electrónicos y conexiones para los distintos dispositivos. (Figura 9)

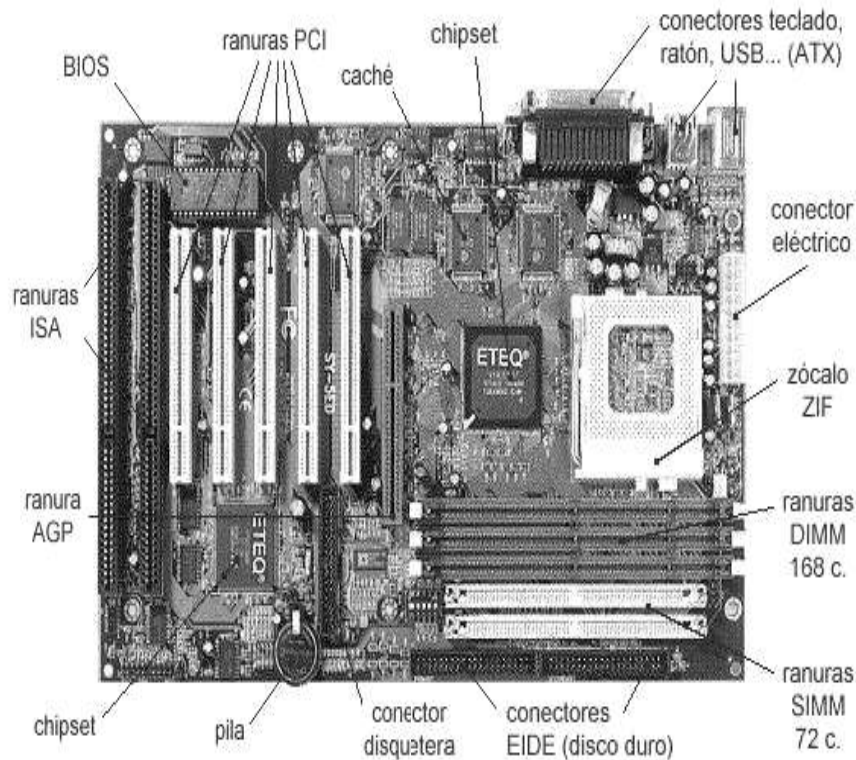


Figura No.9:

Algunos de los componentes y conexiones que forman parte de la placa son:

1. Microprocesador y Zócalo (Socket) del microprocesador: es el elemento más importante del computador, es el cerebro de la máquina, se encarga de controlar todo el sistema. Un parámetro importante es la velocidad del procesador que se mide en mega-hertzios (Mhz), es decir cantidad de "órdenes" por segundo que pueden ser ejecutadas por el procesador.

El zócalo o socket es el lugar en la placa donde se conecta el procesador.

2. Memoria y Ranuras de Memorias: Físicamente son pequeñas láminas finas de materiales sintéticos compuestas de varios chips soldados, cada una de ellas se denomina módulo. Existen diferentes tipos de memorias SIMM DIMM o RIMM entre otras que se diferencian en tamaño físico, velocidad de acceso, número de conectores etc.



Figura No.10: Procesador INTEL

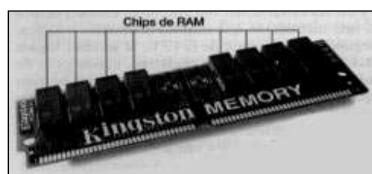


Figura No.11: Memoria RAM

Memoria CACHE: Almacena los programas y datos referenciados más reciente y frecuentemente. Más rápida que la memoria RAM. Está ubicada directamente en el CPU o entre el CPU y la RAM.

Se denominan **ranuras de memoria** al lugar en la placa donde se colocan las memorias. El número de ranuras no es fijo depende de la placa madre.

3. Bios: "Basic Input-Output System: Es un pequeño Programa incorporado en un chip de la placa base. Su finalidad es mantener cierta información básica en el arranque de la computadora. Esta información puede ser la configuración de nuestro disco duro, fecha hora del sistema prioridad de arranque, arranque desde la red etc.

Una de las características de esta memoria es que es una memoria ROM es decir no se borra cuando apagamos el computador. Cuando apagamos, la configuración permanece grabada gracias a una **pila de 3 voltios** que incorpora el computador.

4. Ranuras de expansión. Son las ranuras donde se conectan diversas tarjetas en el sistema. Ejemplos de tarjetas que se pueden instalar son tarjetas de video, audio, o red.

Existen diferentes tipos de ranuras, las más habituales en los computadores son las siguientes:

a. ISA: Son las más antiguas, aunque hoy en día casi no se utilizan algunas placas las incorporan para insertar dispositivos antiguos.

b. PCI: Son las habituales en los computadores actuales.

c. AGP: Normalmente solo hay una porque estas ranuras son de uso exclusivo para tarjetas de video: Estas ranuras son aceleradoras de gráficos 3d.

5. Conectores internos y Conectores eléctricos: Hay dos tipos de conectores, los conectores o interfaces de "datos" y los conectores propiamente eléctricos. Las interfaces de datos conectan los dispositivos a la placa y las conexiones eléctricas conectan la fuente de alimentación a los dispositivos incluida la placa.

Todos los dispositivos excepto las tarjetas de las ranuras de expansión se conectan a la fuente de alimentación. Las tarjetas reciben la tensión a través de las ranuras de expansión.

La fuente de alimentación proporciona la tensión al computador.

V. DECRETO PRESIDENCIAL RELACIONADO AL USO DEL SOFTWARE LIBRE:

El software libre siempre se le ha intentado ver como algo que no sirve para nada, y que sólo lo hacen los piratas, estos mitos han sido sembrados por transnacionales o individuos que no comparten la misma admiración por ser libres. Por lo tanto urge la necesidad de dejarles ver de una manera más clara

y concisa, como ganar dinero y a la vez colaborando y enseñando a una comunidad ansiosa de aprender.

Ya no hay "peros" para no usar software libre, es flexible, portable, es gratis y libre. Además puedes hacer dinero ahorrándote la inversión en software que posee licencia.

El 23 de diciembre de 2004, HUGO CHÁVEZ FRÍAS Presidente de la República Bolivariana de Venezuela, decreto:

Decreto 3.390: de conformidad con lo dispuesto en los artículos 110 y 226 de la Constitución de la República Bolivariana de Venezuela, 12 y 47 de la Ley Orgánica de la Administración Pública y, 2º, 19 y 22 del Decreto con Rango y Fuerza de Ley Orgánica de Ciencia, Tecnología e Innovación, en Consejo de Ministros, **Decreta:** La Administración Pública Nacional empleará prioritariamente Software Libre desarrollado con Estándares Abiertos, en sus sistemas, proyectos y servicios informáticos. A tales fines, todos los órganos y entes de la Administración Pública Nacional iniciarán los procesos de migración gradual y progresiva de éstos hacia el Software Libre desarrollado con Estándares Abiertos.

Se entiende por:

Software Libre: Un Programa de computación cuya licencia garantiza al usuario acceso al código fuente del programa y lo autoriza a ejecutarlo con cualquier propósito, modificarlo y redistribuir tanto el programa original como sus modificaciones en las mismas condiciones de licenciamiento acordadas al programa original, sin tener que pagar regalías a los desarrolladores previos.

Estándares Abiertos: a las Especificaciones técnicas, publicadas y controladas por alguna organización que se encarga de su desarrollo, las cuales han sido aceptadas por la industria, estando a disposición de cualquier usuario para ser implementadas en un software libre u otro, promoviendo la competitividad, interoperatividad o flexibilidad.

Software Propietario: al Programa de computación cuya licencia establece restricciones de uso, redistribución o modificación por parte de los usuarios, o requiere de autorización expresa del Licenciador.

Distribución del Software Libre desarrollado con Estándares Abiertos para el Estado Venezolano: es un paquete de programas y aplicaciones de Informática elaborado utilizando Software Libre con Estándares Abiertos para ser utilizados y distribuidos entre distintos usuarios.

Fuente: Decreto 3.390 de fecha 23-12-04, publicado en Gaceta Oficial 38.095 de fecha 28-12-04

El Estado Venezolano Busca:

1. Reducir la Dependencia Tecnológica
2. Garantizar la Seguridad de la Información y los Procesos
3. Favorecer el Trabajo Cooperativo
4. Garantizar la Interoperabilidad de los sistemas
5. Garantizar el Intercambio de Información
6. Adoptar Estándares no Dependientes
7. Reducir Costos de Replicación de Aplicaciones

8. Favorecer la inversión del componente de desarrollo, soporte y servicio

Nacional.

9. Fortalecer el Capital Humano

10. Garantizar la Transferencia Tecnológica

Desventaja:

1. Fortalecimiento de la Industria Nacional

2. Adaptación de la Academia a los cambios paradigmáticos

3. Mayor preparación y equipamiento del Ciudadano

4. Adaptación del personal de la administración pública al uso de la nueva plataforma.

EJERCICIOS:

1. Discutir en clase las ventajas ofrecidas por el uso del software libre.

2. Identifique los componentes entregados de un sistema de computación, anote la marca de cada componente, busque un lugar seguro y en grupo de 2 o 3 personas arme el sistema de computación. Recuerde todas las sugerencias señaladas en esta guía. Encienda el computador si lo considera necesario.

3. Identifique los componentes de la tarjeta madre entregada por el profesor.

TAREA:

1. Realice un mapa mental referente a los componentes externos de un computador y su utilidad.

2. Busque 3 presupuestos diferentes de un equipo de computación, comente la diferencia de precios que existe entre ellos.

3. Indique en el siguiente cuadro No.1:

* A que dispositivo pertenece el conector.

* Cuantos pines posee y en cuantas hileras están agrupados.

4. Investigue cuantos artículos posee el Decreto 3.390 y comente los aspectos que considere mas importantes.


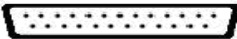
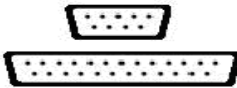



5. Investigue algunos software libre que ofrezcan aplicaciones para la electricidad que se pueden encontrar en la red (Internet) y baje el programa que lleva por nombre Scilab, lea acerca de su creación y utilidad, si posee algún dispositivo de almacenamiento grábelo y tráigalo para la próxima clase.

6. Visitar las siguientes páginas:

www.lug.fi.uba.ar

www.scilab.org

www.cdlibre.org

NOMBRE	FIGURA	DESCRIPCION
Teclado		
		
		Son conectores macho, suelen ser dos: uno estrecho de unos 17 mm, con 9 pines (habitualmente "COM1"), y otro ancho de unos 38 mm, con 25 pines (generalmente "COM2").
		
		
		Bus Serial Universal. USB 2.0

Cuadro No.1

BIBLIOGRAFIA

- La Tarjeta Madre o Mainboard - Monografias_com.htm, Paco Junior Alcoser Serrano
- Programa Nacional de Formación en Sistemas e Informática, MISION SUCRE
- Taller I: Uso básico de equipo de cómputo y periféricos, Secretaría General, Dirección de Informática, Gobierno de México
- Gaceta oficial N° 38.095, Decreto N° 3.390 de fecha 28/ 12/ 2004
- Decreto Presidencial 3.390, Software Libre en la Administración Pública Nacional, José Alfredo Atay, Director General de la Oficina de Tecnologías de Información.
- Software Libre, Hacia la soberanía, independencia y efectividad tecnológica del Instituto Venezolano de Investigaciones Científicas.
- Paginas de Internet:
www.scilab.org
www.cdlibre.org
www.cnice.mec.es
<http://chaslug.org.ve/wiki/index.php>
www.lug.fi.uba.ar/documentos/lista-herramientas/
www.lawebdelprogramador.com/cursos

Estaré muy agradecida por los comentarios, sugerencias o correcciones enviadas a: dsevilla@cantv.net



REPÚBLICA BOLIVARIANA DE VENEZUELA
MINISTERIO DEL PODER POPULAR PARA LA
EDUCACIÓN SUPERIOR
INSTITUTO UNIVERSITARIO EXPERIMENTAL
DE TECNOLOGÍA DE LA VICTORIA
LA VICTORIA- ESTADO ARAGUA
Comisión Académica del Programa
Nacional de Formación de Electricidad



CÁLCULO NUMÉRICO
PARA
SISTEMAS ELÉCTRICOS
**PRACTICA II: INSTALACIÓN E INTRODUCCIÓN A
SCILAB**

Realizado por: Prof. DORIS E. SEVILLA
Revisado por: Prof. JESUS A. PEREZ

La Victoria Octubre de 2007

TEMARIO: INTRODUCCION E INSTALACION DEL PROGRAMA SCILAB:

- I. Introducción.
- II. Instalación del programa.
- III. Menú, Barra de herramientas y Ventana de Comandos.
- IV. Comandos de utilidad.
- VI. Manejo de instrucciones básicas
- VII. Expresiones Algebraicas, Operadores Y Funciones para números reales e imaginarios

OBJETIVOS:

- 1. Instalar el programa Scilab a un computador.
- 2. Reconocer el ambiente en que trabaja Scilab, sus ventanas y directorios.
- 3. Utilizar a Scilab como una poderosa calculadora.
- 4. Resolver ejercicios haciendo uso del programa Scilab.
- 5. Realizar operaciones algebraicas y lógicas con números reales y complejos

I. INTRODUCCIÓN:

Scilab fue desarrollado en el INRIA, Institut National de Recherche en Informatique et Automatique, un excelente instituto francés de investigación. Posteriormente colaboro la escuela de ingenieros ENPC, Ecole Nationale de Ponts et Chaussees.

Es un software de cálculo científico orientado a la computación numérica. Posee una extraordinaria versatilidad y capacidad para resolver problemas de matemática aplicada, física, ingeniería, procesamiento de señales y otras muchas aplicaciones. Su base la constituye un sofisticado intérprete formado por cientos de rutinas de cálculo matricial, análisis numérico y visualización gráfica. El programa está concebido como un software abierto.

El sitio oficial de Scilab es: www.scilab.org. Allí se encuentra información general, manuales, FAQs (frequent asked questions), referencias sobre reportes, diferencias con Matlab, lista de errores. Allí se puede "bajar" la versión binaria o las fuentes para las diferentes plataformas.

II. INSTALACIÓN DEL PROGRAMA SCILAB:

Instalación en Windows

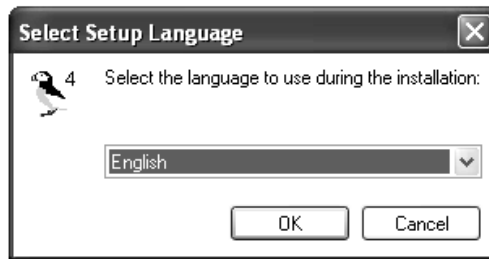
En el caso de plataformas Windows 98, 2000 o XP, el procedimiento de instalación es el estándar: al ejecutar el "Setup", el software se instala automáticamente como cualquier programa Windows, apareciendo el icono de Scilab en el menú de inicio.

Ejemplo No1: Instale Scilab en su computador, el procedimiento que se ilustra es para computadoras que trabajen bajo Windows

Paso 1: Abrir la carpeta donde se tenga grabado el programa y luego:



Paso 2. Se abrirá una ventana, donde se podrá seleccionar el idioma a usar durante la instalación: con dos opciones Ingles o francés, se selecciona English con un clic en OK.



Paso 3: Se abrirá una ventana dándole la bienvenida a la instalación de Scilab, pulsara Next si desea continuar con la instalación o Cancelar si desea salir de ella. Pulsar Next.

Paso 4: Se abrirá una ventana donde aparecen las definiciones y objetivos de los creadores de la licencia de Scilab, aparecerán dos opciones que indican si usted acepta o no los acuerdos, deberá hacer un clic con el Ratón en: **"I accepts the agreements"**, y hacer clic en Next.

Paso 5: Debe seleccionar en que carpeta dentro de su computador se copiara el programa. Por defecto se copia en: C:\Archivos de programa\scilab-4.0, pero si desea otra carpeta con un clic en Browse se desplegara una ventana donde se podrá elegir otra carpeta.



En este ejemplo se copiara el programa en C:\Archivos de programa\.

Paso 6: Aparecerá una ventana podrá seleccionar los componentes que desea instalar de Scilab, seleccionamos Full Instalación. Pulsamos Next

Paso7: Selecciona la carpeta donde se guardaran los archivos creados por scilab, pulsamos Next.

Paso 8: Seleccionamos los objetivos adicionales al programas que necesitamos se instalen como por ejemplo crear un icono en el escritorio para entrar a Scilab, pulsamos Next.

Paso 9: Pulsamos **Install** para comenzar la instalación del programa.

Paso 10: Esperamos unos minutos hasta que se copie el programa, al finalizar la instalación aparecerá una ventana que ofrece información referente a las carpetas de trabajo del programa, pulsar Next.

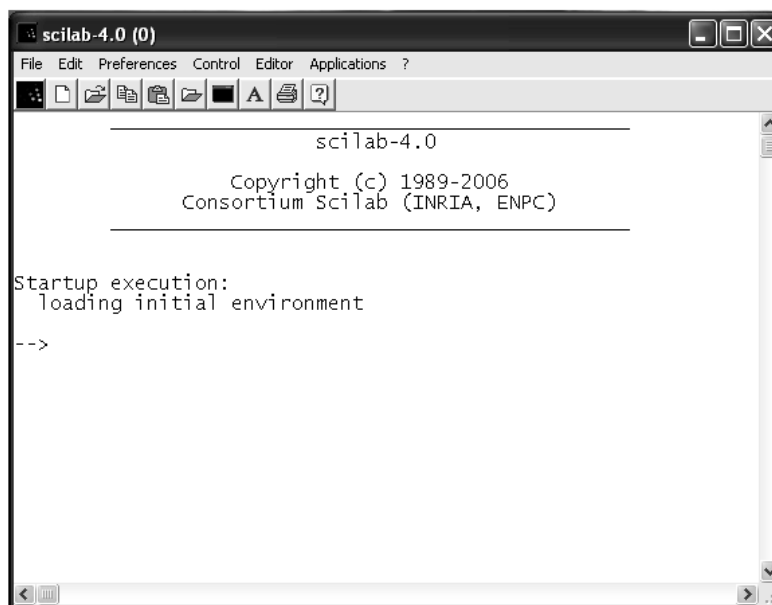
Paso 11: Aparecerá una mensaje que nos indica que la instalación de Scilab se ha completado, al pulsa Finísh, se abrirá el programa.

Paso 12: Para arrancar Scilab, basta con hacer un clic en el icono que aparece en el escritorio:



III. VENTANA DE COMANDOS, MENU Y BARRA DE HERRAMIENTAS:

Al realizar el Paso 12 aparecerá una pantalla como esta:



A partir de ese momento se puede escribir al frente de --> las _ordenes de Scilab. Estas deben ser acabadas oprimiendo la tecla Enter, denominada también Intro.

Esta ventana se llama Ventana de comandos y es donde el usuario opera, aquí se ejecutan los comandos.

La Ventana principal de Scilab cuenta con 7 menús en su parte superior y posee una barra de herramientas que si el usuario lo desea puede verse o no, con pulsar F3, esta se encuentra debajo del menú principal:



Donde:

File: en este menú se abre una ventana dándole al usuario la oportunidad: de abrir otra ventana de scilab (New Scilab), de abrir archivos, de guardar, de imprimir y entre otros de salir del programa.

Edit: seleccionar lo escrito en el Ventana de comandos, copiar líneas del Ventana de comandos, Pegar, etc.

Preferences: el usuario selecciona el aspecto visual del Ventana de comandos, colores, tipos y tamaños de las letras, etc.

Control: resume, aborta una ejecución e interrumpe.

Editor: El editor de ficheros

Applications: Aplicaciones

?:Muestra la ayuda que ofrece Scilab.

Barra de herramientas:

1. Abre otra ventana con Scilab.
2. Abre el editor de texto.
3. Abre archivos previamente grabados.
4. Copia líneas de comandos.
5. Pega líneas previamente copiadas.
6. Para cambiar de directorio.
7. Consola
8. Cambia tamaño y color de la letra usada en el Ventana de comandos.
9. Imprime.
10. Ayuda

IV. COMANDOS DE UTILIDAD

El comando demo nos muestra, de modo interactivo, un amplio abanico de ejemplos de aplicación de Scilab y es de gran ayuda durante nuestros inicios con el programa. El comando help función _ deseada muestra una librería de ayuda con respecto a la función.

who: Para obtener información sobre las variables utilizadas.

whos: proporciona una información más detallada de who.

Con las teclas-flechas `↑` `↓` `←` `→` `y` `↵`: para recuperar comandos anteriores de Scilab.

clc: se utiliza para borrar todas las salidas anteriores de Scilab y dejar limpia la **Ventana de comandos**.

quit y exit: Si se desea salir de Scilab

dir: igual que el de DOS, lista el directorio actual.

Ejemplo 2: Help

--> help plot

Esta instrucción abre una librería de ayuda donde aparecen parámetros, descripción y ejemplos aplicados para el comando PLOT.

--> help pdiv

Ejemplo 3: who

-->who

your variables are...

```
ged_select_axes      ged_paste_entity    ged_copy_entity     ged_move_entity
ged_delete_entity
DestroyGlobals      ReLoadTicks2TCL     Subtickstoggle      LoadTicks2TCL
setTicksTList
GetTab2  GetTab  GEDeditvar      GEDeditvar_get      CloseEditorSaveData
EditData
setchampntlistXYFXFY      setGrayplottlist    set3dtlistXYZC      set3dtlistXYZ
setZb
setA2val setA1val setHval setWval setZval setYval setXval getparaxe.....
```

Ejemplo 4: Dir

-->dir

ans =

Application Data Favoritos Mis documentos WINDOWS
Escritorio Menú Inicio Scilab

-->

Ejercicio 1: Tecleen en su pantalla los siguientes comandos precedidos por help:

Poly, Plot2d, symbols, roots, simbolos. Analice los resultados de su búsqueda.

VI. MANEJO DE INSTRUCCIONES BÁSICAS:

Al escribir en el Ventana de comandos la siguiente operación matemática y pulsar

Enter:

-->3+5

ans = 8.

-->

Scilab realiza la operación y genera un variable por defecto llamada ans donde guarda el valor.

Ahora se creara una variable y se escribirá la misma operación:

-->a=3+5

a = 8.

guarda en la variable el resultado y lo muestra, si no es necesario esta ultima acción se escribe después de la orden punto y coma (;):

-->a=8+4;

Scilab diferencia las letras mayúsculas de las minúsculas. En una línea pueden escribirse varias órdenes, estas deben estar separadas por comas o punto y comas.

Ejemplo 5: En este ejemplo se pondrá a trabajar a Scilab como una calculadora básica, defina 4 variables y asigne valores diferentes, realice las operaciones de suma, resta, multiplicación y división:

-->a=2; b=65 ;c=1; d=0; (ENTER)

-->S=a+b (ENTER)

```

S = 67.
-->R=d-c (ENTER, a partir de ahora se obviara enter, tendrá que sobreentenderse)
R = - 1.
-->M=a*c*b
M = 130.
-->k=8*a
k = 16.
-->f=d-12+1
f = - 11.
-->F=d*9
F =
0.
-->u=a+b+c ;i=d*a*b;
-->u
u =
68.
-->i
i = 0.

```

```

-->exp(X)^2
ans = 7.3890561

```

```

-->max(X,Y)
ans = 1.

```

```

-->min(X,Y,Z)
ans = - 4.

```

```

-->h=sqrt(X^2-Y^2)
h = 3.8729833i

```

```

-->cos(%pi)
ans = - 1.

```

```

-->cos(0)
ans = 1.

```

```

-->sin(%pi/2)
ans = 1.

```

Ejemplo 8: Manejo de los números complejos

Forma de ingresar los números complejos: se le antepone a la parte imaginaria el símbolo del % (porcentaje)

```

-->a=-5*%i; b=2+4*%i; c=-3+%i;
-->S=a+b+c
S = - 1.
-->R=a-b
R = - 2. - 9.i
-->Q=sqrt(R)
Q = 1.8999401 - 2.3684958i
-->sin(Q)
ans = 5.0982651 + 1.7111448i

```

```
-->Re=real(Q),Im=imag(Q)
Re = 1.8999401
Im = -2.3684958
-->
```

Ejercicios:

1. Complete la siguiente la tabla aplicando los comandos de Scilab a las variables indicadas:

x	y	ceil(x)	fix(x)	floor(x)	max(x,y)	min(x,y)	round(x)
1.0	2.2						
0.8	0.85						
0.5	-0.5						
2.1	2.0						
-4.1	4.5						
-4.5	2.5						

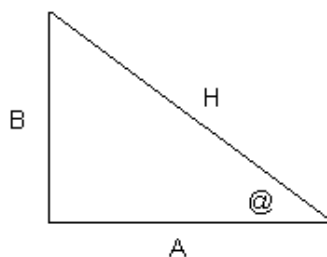
2. Complete la siguiente tabla aplicando las funciones trigonometricas que se les indica usando Scilab:

t	sin(t)	cos(t)	tan(t)	tanh(t)
0				
$\pi / 2$				
π				
$3 \pi / 2$				
2π				

3. Calcule el área de un triangulo equilátero, haciendo uso del programa scilab, sabiendo que uno de sus lados mide 5cm.

4. Calcule la hipotenusa del siguiente triangulo recto:

Donde: A=5mts, B=5mts, @=?, H=?



5. Calcule el área de un rectángulo cuyas medidas son: base=3cms, altura=5cms.

6. Encuentre la respuesta a las siguientes operaciones matemáticas:

A ^b	Resultado
$-3^4 = -3x-3x-3x-3$	
$5^3 =$	
$(\frac{1}{2})^{-3} =$	
$4^{-1} =$	
$(\frac{2}{5})^{-6} =$	
$3^5 \times 3^2 =$	
$(7+1)^2 =$	
$(\frac{3}{7})^4 =$	
$\sqrt[3]{2^3} =$	
$\sqrt[3]{41}^{\frac{3}{2}} =$	

Ejercicios propuestos por el Profesor

BIBLIOGRAFIA

1. Introducción a SCILAB, Héctor Manuel Mora Escobar Departamento de Matemáticas, Universidad Nacional de Colombia Bogotá mayo del 2005.
2. Fundamentos Scilab y aplicaciones, de Andrés Alfonso caro, Cesar Valero.
3. Introducción A Scilab (Programa De Cálculo Numérico) Febrero/Marzo 2005, Laboratorio De Computación de Alto Desempeño (Icad), Facultad de ingeniería, Universidad nacional de asunción.
4. Manual de Introducción al Tratamiento de Señales con SCILAB para usuarios de MATLAB. Programa de Doctorado en Automática E Informática Industrial, por Bernardo A. Delicado
5. Scilab. Computación Numérica Bajo Linux Y Windows, por Andrés Jiménez Jiménez. Titulado Superior de Apoyo a Investigación, Universidad de Cádiz.

Estaré muy agradecida por los comentarios, sugerencias o correcciones enviadas a: dsevilla@cantv.net



REPÚBLICA BOLIVARIANA DE VENEZUELA
MINISTERIO DEL PODER POPULAR PARA LA
EDUCACIÓN SUPERIOR
INSTITUTO UNIVERSITARIO EXPERIMENTAL
DE TECNOLOGIA DE LA VICTORIA
LA VICTORIA- ESTADO ARAGUA
Comisión Académica del Programa
Nacional de Formación de Electricidad



CALCULO NUMÉRICO

PARA

SISTEMAS ELÉCTRICOS

PRÁCTICA III: MANEJO DE EXPRESIONES ALGEBRAICAS, OPERADORES Y FUNCIONES PARA POLINOMIOS, VECTORES Y MATRICES

Realizado por: Prof. DORIS E. SEVILLA
Revisado por: Prof. JESUS A. PEREZ

La Victoria Octubre de 2007

TEMARIO:

I. POLINOMIOS:

I.1 Sintaxis

I.2 Operaciones básicas

II. VECTORES Y MATRICES:

II.1 Creación

II.2 Notación y operaciones

II.3 Funciones elementales

II.4 Ejercicios

OBJETIVOS:

- 1. Definir polinomios de dos formas diferentes.**
- 2. Trabajar las operaciones básicas de los polinomios:**
 - Calculo de raíces,**
 - Evaluación de polinomio en un punto,**
 - Multiplicación y división de polinomio.**
- 3. Crear vectores y matrices**
- 4. Realizar álgebra de matrices.**
- 5. Manejar diferentes funciones elementales.**

I. POLINOMIOS:

Los polinomios pueden definirse de dos formas diferentes utilizando la primitiva `poly()`:

a) Especificando sus raíces

```
--> p = poly([1 2], 'x')
```

$$2 - 3x + x^2$$

```
--> roots(p)
```

```
ans = 1.
```

```
2.
```

b) Especificando sus coeficientes con la letra c:

```
--> q = poly([1 2], 'x', 'c')
```

$$q = 1 + 2x$$

```
--> roots(q)
```

```
ans = -0.5
```

La función `roots()` se aplica para calcular las raíces reales o complejas del polinomio.

La función `coeff` se aplica para obtener el valor del coeficiente que acompaña a la variable que posee el grado especificado, posee dos parámetros, el primero es el polinomio y el segundo la potencia. La siguiente orden asigna a la variable `z` el valor +13, el coeficiente de x^2 en el polinomio `b`.

```
--> z = coeff(b, 2)
```

```
z = 13.
```

Si se utiliza simplemente la variable que identifica al polinomio se obtendrán todos los coeficientes:

```
--> c = coeff(b)
```

```
c = 4. - 12. 13. - 6. 1.
```

La función `horner` se aplica para evaluar un polinomio, tiene dos parámetros, el primero es el polinomio y el segundo el valor donde se evalúa el polinomio:

```
--> x1 = horner(b, 1)
```

```
x1 = 0.
```

```
--> x0 = horner(b, 0)
```

```
x0 = 4.
```

Las operaciones algebraicas elementales suma, resta, producto y cociente se realizan con los operadores habituales $p+q$, $p-q$, $p*q$, p/q .

También se puede elevar un polinomio a una potencia, por ejemplo:

```
--> b = p^2
```

$$b = 4 - 12x + 13x^2 - 6x^3 + x^4$$

Ejercicio I. 1: Defina los siguientes polinomios en Scilab:

Polinomio	Scilab
$1 \leq 6 - S + S^2$	
$2 \leq D^3 - D^2 - 1$	
$3 \leq [X - 4][X + 2]$	

Ejercicio I.2: Con los siguientes polinomios, realice las siguientes operaciones:

$$p(x) = x^2 - 2$$

$$q(x) = x^3 - 2x^2 + 3x - 1$$

$$r(x) = 2 + x^2 - x + 5$$

$S(x) = p(x) + r(x)$	$R(x) = q(x) - p(x) - r(x)$	$H(x) = p(x) + 2$

Ejercicio I.3: Realice las siguientes operaciones aplicando los métodos estudiados en matemática y compruebe sus resultados con Scilab:

$$p(x) = 3x^3 - 2x^2 + 5x - 3$$

$$t(x) = x + 3$$

$$s(x) = x - 2$$

$$h(x) = 6x^3 - \frac{3}{2}x + 2$$

$$w(x) = 5x^4 - 3x^2 + 1$$

$$g(x) = 2x - 1$$

$F(x)$	Resultado
$p(x) / s(x)$	
$s(x) \times g(x)$	
$w(x) / t(x)$	
$g(x) \times t(x)$	
$h(x) / g(x)$	

Ejercicio I.4: Calcule las raíces a los siguientes polinomios y evalúe en $x=b$:

$f(x)$	Raíces	$x=b$
$x + 3$		$b=0, f(b)=$
$2x^4 - 10x^2 + 8$		$b=2, f(b)=$
$6x^3 - 7x^2 - x + 2$		$b=-1/2, f(b)=$

II. VECTORES Y MATRICES:

II.1 Creación: En Scilab no hay vectores como tales, los vectores se deben asimilar a matrices de una sola fila o vectores fila (tamaño $1 \times n$) o a matrices de una sola columna o vectores columna (tamaño $n \times 1$).

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

se puede definir por medio de:

$$A = [1 \ 2 \ 3; 4 \ 5 \ 6; 7 \ 8 \ 9]$$

o también por medio de:

$$A = [1,2,3;4,5,6;7,8,9]$$

II.2 Notación y operaciones

Ejemplo II.1: Escriba en la ventana de comandos

```
-->A=[1 2 3;4 5 6;7 8 9]
```

```
A =
```

```
1.  2.  3.
4.  5.  6.
7.  8.  9.
```

```
-->A=[1,2,3;4,5,6;7,8,9]
```

```
A =
```

```
1.  2.  3.
4.  5.  6.
7.  8.  9.
```

Ejemplo II.2. Cree dos vectores fila con 4 elementos y dos vectores columna del mismo tamaño:

```
-->F1=[14 0 5 -8*%i]
```

```
F1 =
```

```
14.  0  5. - 8.i
```

```
-->F2=[2 9 -1 0]
```

```
F2 =
```

```
2.  9. - 1.  0.
```

```
-->C1=[2;5;4*%i;3]
```

```
C1 =
```

```
2.
5.
4.i
3.
```

```
-->C2=[-1*%i;3*%i;8;0]
```

```
C2 =
```

```
- i
3.i
8.
0
```

```
-->
```

Ejemplo II.3: Realice operaciones algebraicas con los vectores creados:

-->S=F1+F2

S = 16. 9. 4. - 8.i

-->R=F2-F1

R = - 12. 9. - 6. 8.i

-->Sc=C1+C2

Sc =

2. - i

5. + 3.i

8. + 4.i

3.

-->Df=F1/F2; //la división se realizan con vectores de igual tamaño

-->Df

Df = 0.2674419

-->M=C1*F1

M =

28. 0 10. - 16.i

70. 0 25. - 40.i

56.i 0 20.i 32.

42. 0 15. - 24.i

-->M1=F2*C1

M1 = 49. - 4.i

-->M1=C1*F2

M1 =

4. 18. - 2. 0

10. 45. - 5. 0

8.i 36.i - 4.i 0

6. 27. - 3. 0

-->M1=F1*C1

M1 = 28. - 4.i

II.3 Funciones Elementales: Scilab permite crear rápidamente algunos tipos especiales de matrices:

ones(4,5) es una matriz de unos de tamaño 4 x 5

zeros(4,5) es una matriz de ceros de tamaño 4 x 5

rand(20,30) es una matriz aleatoria de tamaño 20 x 30

eye(4,4) es la matriz identidad de orden 4

La multiplicación (*) es el producto matricial. Para realizar estas mismas operaciones pero elemento a elemento debemos utilizar los operadores (.*) ./ .^).

Por supuesto también existen las funciones propias del tratamiento matricial algunas de estas son: inversa de una matriz cuadrada "inv()",

determinante de una matriz cuadrada "*det()*", descomposición singular "*svd()*", rango de una matriz *rank()* etc.

La operación denominada transposición consiste en convertir las filas en columnas y las columnas en filas de un vector o matriz (*A'*) se realiza situando el apóstrofe " ' "

Ejercicio II.1:

Genere en Scilab las siguientes matrices y complete el cuadro No.1, cuando las funciones no sean aplicables a la matriz, explique porque:

$$A = \begin{bmatrix} 1 & 0 & -9i \\ 3 & 8 & 2+6i \end{bmatrix} \quad B = \begin{bmatrix} 2+1i & 5 \\ 3 & 10 \end{bmatrix} \quad C = \begin{bmatrix} 2 & 6 & 5 \\ 9 & 8 & 0 \\ 1 & 3 & -1 \end{bmatrix}$$

Matriz:	det	inv()	N^2	sort()	svd()	rank	triu()
A							
B							
C							
B*A*C							
rand(2,3)							
eye(3,3)*C							

Cuadro No.1 Funciones propias del tratamiento matricial

Tareas:

1. Encuentre las raíces de los siguientes polinomios y compruebe el resultado evaluando cada una de las raíces generadas:

$$a = x + 6$$

$$b = x^2 - 9x - 20$$

$$c = x^2 - 7x - 10$$

$$d = x^2 - 21x + 20$$

$$e = x^3 - 2x^2 - x - 2$$

$$f = x^4 - 5x^3 - 8x^2 - 7x - 3$$

$$g = x^4 - 11x^2 - 18x - 8$$

2. Dadas las raíces, encuentre el polinomio característico:

$$a = -5, \pm 2$$

$$b = +9, \pm 6$$

$$c = +2, -2, \pm 3$$

$$d = 1, -2, -1/3, -2/5$$

$$e = 1, -1$$

$$f = -1, \pm 1, -2, -2$$

$$g = 0, 1$$

3. Realice las siguientes operaciones en Scilab:

$$a. \quad 2m^4 - 3m^3 \pm 5m^2 - 6m \pm 10 \pm m - 2 \pm$$

$$b. \quad 4x^4 - 8 \pm x + 1 \pm$$

$$c. \quad a^3 - 512 \pm a - 8 \pm$$

$$d. \quad 2/3x^3 - 1/6x^2 \pm 2/5x - 1/3 \pm x - 1 \pm$$

4. Genere las siguientes matrices:

$$A = \begin{bmatrix} 4 & 5 & 0 \\ 9 & -3 & 1 \\ -5 & 0 & 6 \end{bmatrix} \quad B = \begin{bmatrix} 5 \\ 2 \\ 0 \end{bmatrix}$$

Realice las siguientes operaciones:

- Encuentre el determinante de A
- Inversa de A
- $C = A \times B$

BIBLIOGRAFIA

- Introducción a SCILAB, Héctor Manuel Mora Escobar Departamento de Matemáticas, Universidad Nacional de Colombia Bogota mayo del 2005.
- Fundamentos Scilab y aplicaciones, de Andrés Alfonso caro, Cesar Valero.
- Introducción A Scilab (Programa De Cálculo Numérico) Febrero/Marzo 2005, Laboratorio De Computación de Alto Desempeño (Icad), Facultad de ingeniería, Universidad nacional de asunción.
- Scilab. Computación Numérica Bajo Linux Y Windows, por Andrés Jiménez Jiménez. Titulado Superior de Apoyo a Investigación, Universidad de Cádiz.

Estaré muy agradecida por los comentarios, sugerencias o correcciones enviadas a: dsevilla@cantv.net

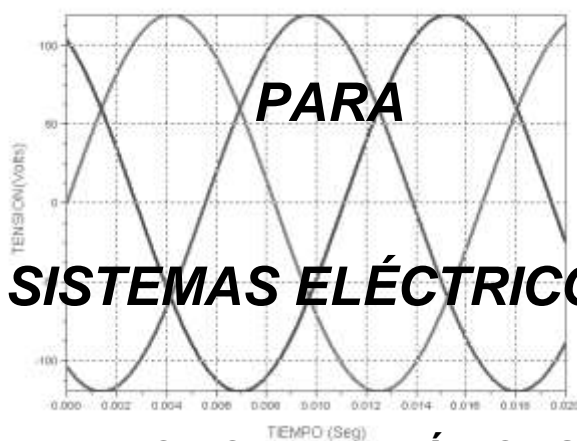


REPÚBLICA BOLIVARIANA DE VENEZUELA
MINISTERIO DEL PODER POPULAR PARA LA
EDUCACIÓN SUPERIOR
INSTITUTO UNIVERSITARIO EXPERIMENTAL
DE TECNOLOGIA DE LA VICTORIA
LA VICTORIA- ESTADO ARAGUA
Comisión Académica del Programa
Nacional de Formación de Electricidad



CÁLCULO NÚMÉRICO

FORMAS DE ONDA DE TENSION DE UN GENERADOR TRIFASICO



SISTEMAS ELÉCTRICOS

PRACTICA IV: GRÁFICOS

Realizado por: Prof. DORIS E. SEVILLA
Revisado por: Prof. JESUS A. PEREZ

La Victoria Octubre de 2007

TEMARIO:

I. Gráficos en dos dimensiones:

I.1 Comando plot.

I.2 Comando plot2d

I.3 Comando gca

I.4 Grafica de vectores

II. Gráficos en tres dimensiones

OBJETIVO:

1. Introducir al estudiante en el manejo de los comandos básicos de Scilab para graficar curvas.

2. Reconocer a Scilab como una herramienta útil para visualizar la geometría analítica en plano y el comportamiento de algunas funciones vectoriales.

3. Realizar gráficos en dos dimensiones.

4. Realizar gráficos tres dimensiones.

5. Graficar las funciones básicas usadas en la electricidad.

6. Utilizar a Scilab como una herramienta de comprobación de resultados.

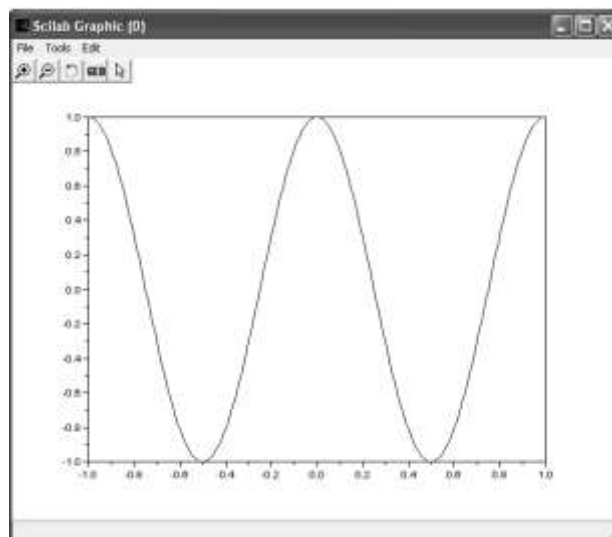
GRAFICOS:

La gama de gráficos científicos que Scilab puede realizar es enorme, comprende gráficos en dos y en tres dimensiones, en coordenadas cartesianas y en paramétricas, en escalas decimal, logarítmica y semilogarítmica, además de representaciones específicas para datos estadísticos, sistemas de control (Bode, Niquist, etc.), animaciones, etc.

I. Gráficos en dos Dimensiones: El caso más sencillo de gráfico 2D es el de una función $y=f(t)$, en el ejemplo siguiente se muestra como utilizar la función "*plot()*" para construir graficos sencillos. También se usa el comando "*plot2d2*" para graficas de dos dimensiones con mejores características.

Ejemplo I.1.1

```
--> t=(-1:0.01:1);           // Definición del rango de t.  
--> ft = cos(2*%pi*t);       // Definición de f(t)  
--> plot( t, ft);            // Ejecución del gráfico del tipo y = f(t)
```



Lo primero que podemos observar en las figuras es que el dibujo se realiza sobre una ventana gráfica. La ventana consta de un menú superior con cuatro botones:

- 3D Rot.: Permite aplicar rotaciones a los gráficos tridimensionales empleando el ratón.

- 2D Zoom: Amplia un gráfico 2D. Este comando puede ser ejecutado recursivamente.

- Un Zoom: Recupera el gráfico inicial.

- File: Abre un panel que permite imprimir el gráfico, exportarlo a un formato específico (Postscript, Postscript- LATEX, Xfig), salvarlo como gráfico Scilab para luego cargarlo.

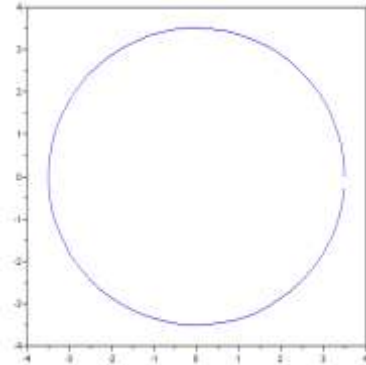
I.1 Sintaxis del comando Plot: **plot(Variable ind., variable dep.)**

* Para cambiar parámetros del gráfico: color, tipo de líneas, fondo, espesor de líneas, etc., ver: `--> xset()`

- * Para abrir una nueva ventana de gráfico: --> `xset('window', número de ventana)`
- * Borrar el contenido de la ventana actual: --> `xbasc()`

Ejemplo I.1.2 Grafica de una función vectorial en el plano

```
deff('[x]=f1(t)','x=3.5*cos(t)');//Define una función x=f(t)
deff('[y]=f2(t)','y=3.5*sin(t)');//Define una función
t = [0:0.1:2*pi]; //Define el dominio del parámetro
x = f1(t); //Vector imagen en X
y = f2(t); //Vector imagen en y
clf() //Limpia la ventana de dibujo
plot(x,y) //Dibuja la función paramétrica
```

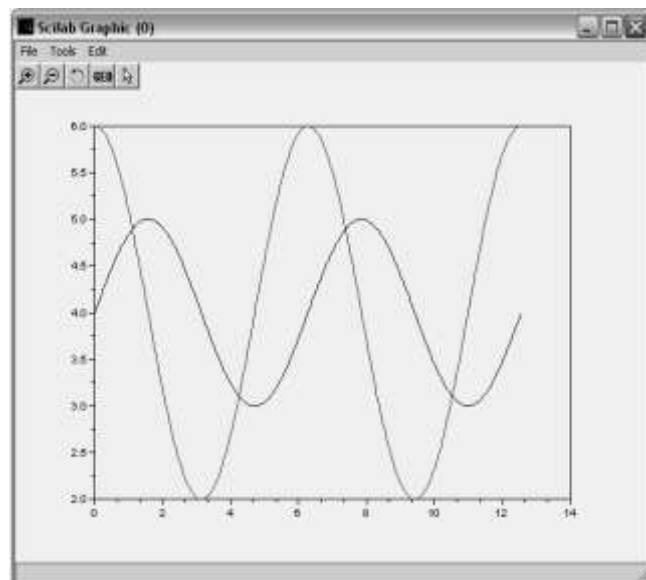


Ejemplo I.1.3: Grafique las siguientes funciones en una sola grafica:
 $a=4+\sin(t)$,

$b=4+2\cos(t)$

Scilab:

```
t=0:0.05:4*pi;
a=4+sin(t);
b=4+2*cos(t);
plot(t,a,t,b)
```

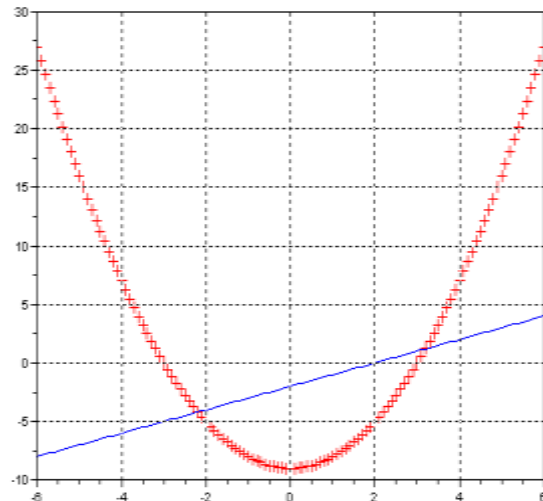


Ejemplo I.1.4

Grafique las siguientes funciones y colóquele una malla: $Y(x)=x^2-9$; $Y1(x)=x-2$
 Para colocar un malla a la grafica se escribe el comando **xgrid**.

Scilab:

```
x=-6:0.1:6; //se define el vector
//dominio de la función
Y=x^2-9; //parábola
Y1=x-2; //recta
plot(x,Y,'r.+',x,Y1); //la grafica de Y
//será roja y marcada con +
xgrid //malla
```

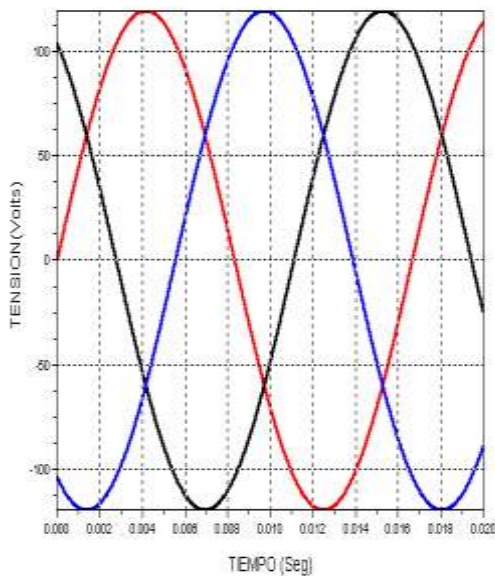


Ejemplo II.1.5: Grafique las formas de onda de un Sistema trifásico equilibrado tomando como referencia $V_1=120\text{Sen}(wt)$, con una frecuencia de 60Hz.

Scilab:

```
t=[0:0.00001:0.02];
w=2*pi*60;
Q=120*pi/180;
v1=120*sin(w*t);
v2=120*sin(w*t+Q);
v3=120*sin(w*t-Q);
plot(t,v1,t,v2,t,v3)
xtitle('FORMAS DE ONDA DE
TENSION DE UN GENERADOR
TRIFASICO',
'TIEMPO(Seg)','TENSION(Volts)');
xgrid
//Para modificar el aspecto de la
grafica en la venta donde aparece la
grafica, en Edit y luego figure
properties (propiedades de la grafica)
```

FORMAS DE ONDA DE TENSION DE UN GENERADOR TRIFASICO



I.2 Comando Plot2d: (variable ind, variable dep, "argumentos")

--> plot2d(x,y,[style,strf,leg,rect,nax])

Donde:

x, y matrices o vectores a graficar

style: vector conteniendo números que definen el color. Para graficar usando símbolos (+, *, o, etc.) usar números negativos.

strf = "xyz" donde:

x = 1, muestra leyenda de líneas

y = 1, usa rect;

y = 2, calcula bordes usando xmax y xmin

y = 3, similar a y = 1 pero con escala isométrica

y = 4, similar a y = 2 pero con escala isométrica

z = 1, ejes graficados de acuerdo a especificaciones en nax

z = 2, marco del gráfico sin grilla
leg = "nombrelínea1@| nombrelínea2@..."

rect = [xmin, ymin, xmax, ymax]

nax = [nx, Nx, ny, Ny] donde:

nx,ny = sub-graduaciones de x,y;

Nx,Ny = graduaciones de x,y.

- Para colocar título a un gráfico:

--> xtitle('Nombre_del_gráfico', 'Nombre_eje_x', 'Nombre_eje_y')

- Creando sub-ventanas:

➔ xsetech(wrect)

Donde wrect es un vector de 4 elementos [x, y, "ancho", "alto"] donde "ancho" y "alto" definen en cuantas ventanas estará dividida la ventana, "x" e "y" definen cual de las ventanas activar.

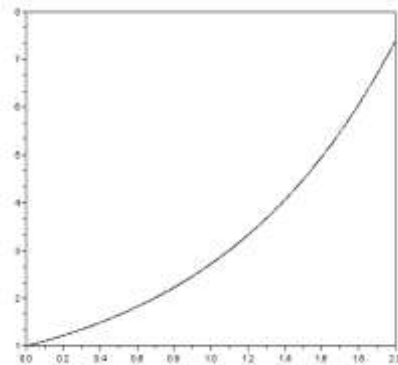
Ejemplo I.2.1 En la siguiente grafica se explicara paso a paso los efectos de los argumentos mas usados. Grafique: $f(x)=e^x$

Paso 1: No se utiliza argumentos

-->x=[0:0.05:2];

-->y=exp(x);

-->plot2d(x,y)

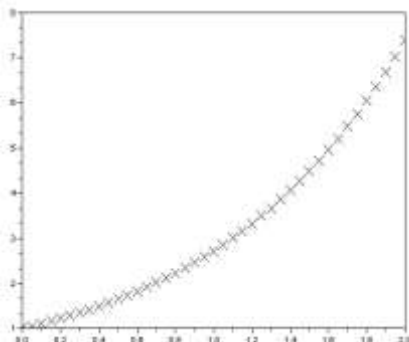


Paso 2: Definimos el estilo de la grafica, la curva se marcara con x, para ello se coloca como 1^{er} argumento el -2. Si queremos cambiar el color de la curva usamos números positivos.

-->x=[0:0.05:2];

-->y=exp(x);

-->plot2d(x,y,-2)

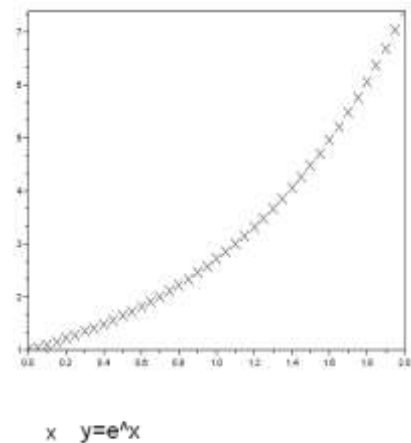


Paso 3: Se define la legenda de la grafica.

-->x=[0:0.05:2];

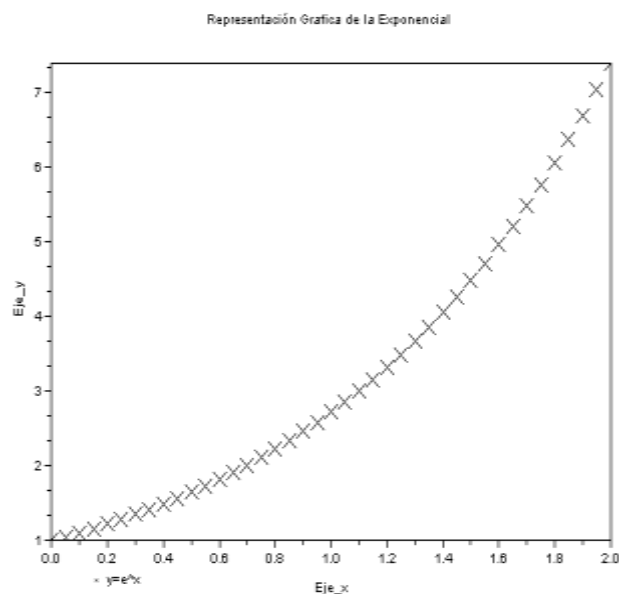
-->y=exp(x);

-->plot2d(x,y,-2,"121",leg="y=e^x")



Paso 4: Se completa la grafica definiendo el titulo y los ejes:

```
x=[0:0.05:2];
y=exp(x);
plot2d(x,y,-2,"121",leg="y=e^x")
xtitle('Representación Grafica de la Exponencial', 'Eje_x', 'Eje_y')
```



1.3 Comando “gca()”: es un comando que nos permite mejorar notablemente las características de nuestros gráficos creados previamente.

Uso de **gca()**:

Después de haber creado una grafica con una función o modelo determinado es necesario crear una variable e igualarla a gca(), así:

```
a=gca();
```

Para mejorar o hacer cambios a los parámetros, se escribe de la siguiente forma: a.variable a cambiar.

Los parámetros que se modifican con mas frecuencia son:

1. Ejes de coordenadas: a.x_location="ubicación";
a.y_location="ubicación"

Ubicación:

Centrados= "middle"

Izquierda = "left"

Derecha = "right"

Superior = "top"

Inferior = "bottom"

2. Tamaño de las letras de los títulos:

```

xtitle("Titulo","Eje x", "Eje y");
a.title.font_style=5;
a.title.font_size=4;

```

- Colocar notas adicionales: `xstring(x,y,"texto",angulo, box)`
`x,y` = son las coordenadas donde se ubique la nota.
`Texto` = nota adicional
`ángulo` = ángulo en que se escribirá la nota.
`Box` = colocar la nota en una caja, si=1, no=0.

Ejemplo I.2.1:

Grafique las siguientes funciones:

$$G(x) = x^2 - 16$$

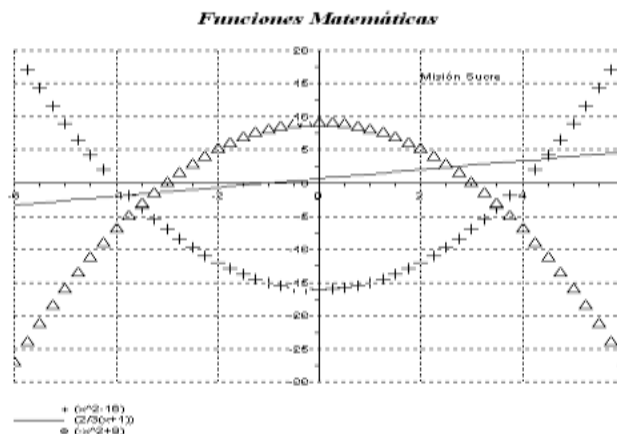
$$F(x) = \frac{2}{3}x + 1$$

$$H(x) = -x^2 + 9$$

```

x=[-6:0.25:6];
G=(x^2-16);
F=2/3*(x+1);
H=-x^2+9;
plot2d(x',[G',F',H'],[-1 1 -6],leg="(x^2-16)@(2/3(x+1))@(-x^2+9)")
a=gca();
a.x_location="middle";
a.y_location="middle";
xtitle("Funciones Matemáticas");
a.title.font_style=5;
a.title.font_size=4;
xstring(2,15,"Misión Sucre",0,0);
xgrid

```



1.4 Representación gráfica de vectores:

Los vectores se pueden representar en coordenadas cartesianas(x,y), coordenadas polares(r,θ).

El ángulo se debe llevar de grados a radianes: $\theta(\text{rad}) = \theta(\text{grados}) * \pi / 180^\circ$

Dado r , el vector viene expresado por $V = r * \cos(\theta) + r * \sin(\theta) * i$

Para representar gráficamente el vector se utiliza el comando `plot2d`:

```
plot2d(real(V),imag(V),rect=[0,0,10,10])
```

Cuando se desea realizar la operaciones algebraicas con vectores de forma gráfica es de gran utilidad el comando plot2d4 que permite la unión de curvas unidas por flechas (vectores)

Para graficar los vectores es necesario especificar las coordenadas iniciales en x y y.

```
x=[inicial,final], y=[inicial, final]
plot2d4(x,y,"parámetros")
```

Para graficar varios vectores es necesario que por cada vector se especifique sus valores iniciales y finales en x y y, generando dos matrices:

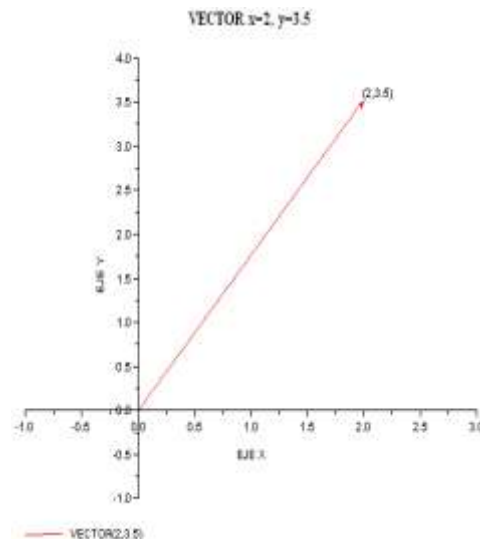
```
x= | inicio(vector1) ... inicio(vectorN)|
   | inicio(vector1) ... inicio(vectorN)|
```

```
y= | fin(vector1)      ... fin(vectorN) |
   | fin(vector1)      ... fin(vectorN) |
```

Ejemplo I.4.1 Grafique los siguientes vectores: $x=[0,2]; y=[0,3.5];$

Scilab:

```
x=[0,2];y=[0,3.5];
plot2d4(x,y,5,leg="VECTOR(2,3.5)",rect=[-
1,-1,3,4]);
xtitle("VECTOR x=2, y=3.5", "EJE X","EJE
Y");
xstring(2,3.5,"(2,3.5)");
c=gca();
c.x_location="middle";
c.y_location="middle";
c.title.font_style=4;
c.title.font_size=3;
```



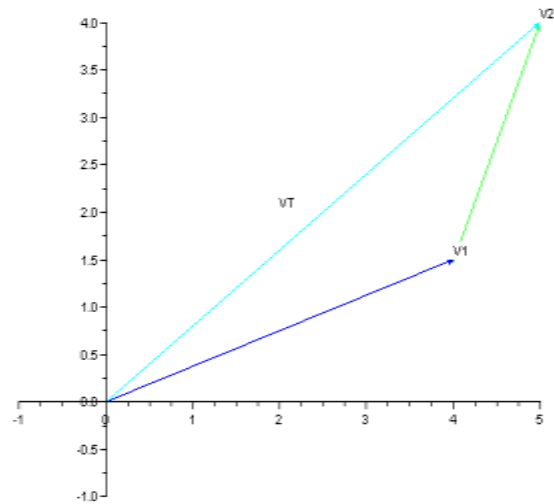
Ejemplo I.4.2 Grafique los siguientes vectores V_1, V_2 y V_T :

```
x=| 0 4 0|      y=| 0  1.5  0|
   | 4 5 5|      | 1.5  4  4|
```

```

V1x=[0;4];V2x=[4;5];VTx=[0;5];
V1y=[0;1.5];V2y=[1.5;4];
VTy=[0;4];
x=[V1x V2x VTx];
y=[V1y V2y VTy];
plot2d4(x,y,[2,3,4],rect=[-1,-1,5,4]);
z=gca();
z.x_location="middle";
z.y_location="middle";
xstring(4,1.5,"V1")
xstring(5,4,"V2")
xstring(2,2,"VT")

```



II. Gráficos en tres dimensiones:

Se construyen de forma análoga con la orden `"plot3d(x,y,z,argumentos)"`. Puede observarse que ahora hace falta especificar el rango de variación de dos parámetros y si se desea los ángulos de rotación de los ejes.

Estos gráficos son realmente una superficie representada por los valores de x, y, z

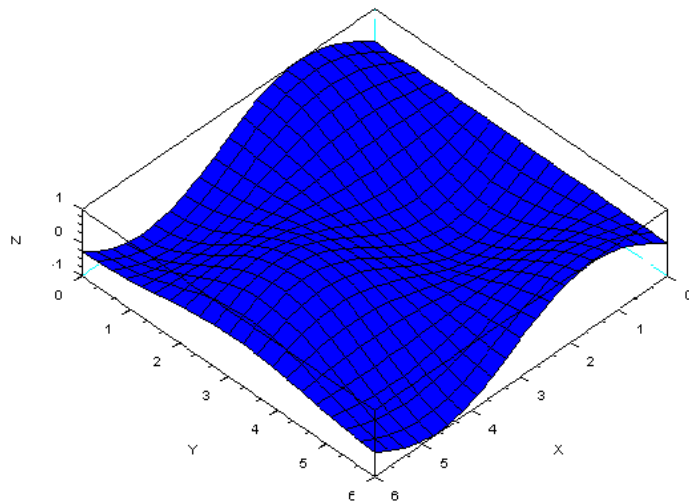
Para realizar graficos en 3D hay que tener en cuenta lo siguiente: "x" y "y" son vectores x_m y y_n dimensiones, z es una matriz de dimensión (x_m, y_n) .

Ejemplo II.1

```

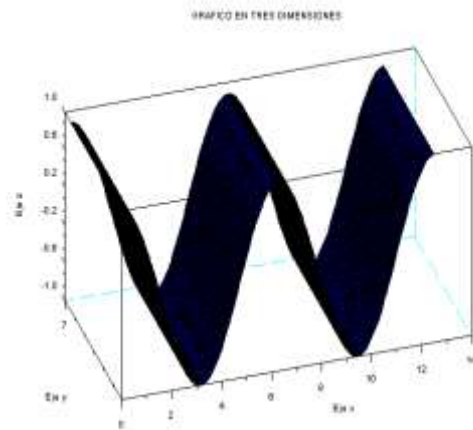
t=[0:0.1:2*pi]';
Z=sin(t)*cos(t');
Plot3d(t,t,Z)

```



Ejemplo II.2:

```
x=[0:0.1:4*%pi];//vector con 126
componentes
y=cos(x);
z=y*ones(1,126);//se forma una matriz
cuadrada
plot3d(x,x/2,z,leg="Eje x@Eje y@Eje
z")
xlabel("GRAFICO EN TRES
DIMENSIONES")
```



Ejercicios:

1. Genere los siguientes gráficos: $t=0:0.05:2*\pi$

$$F(t)=4+\cos(t)$$

$$G(t)=-2+\sin(t)$$

$$H(t)=2+2\sin(t/2)+2\cos(t/2)$$

$$K(t)=120\sin(120t)$$

2. Al siguiente circuito se le realizaron varios ensayos para observar el comportamiento de la corriente:

1. Cuando se varía la fuente de tensión de [0-50v] y la resistencia se deja fija en 1200 ohmios, los datos arrojados se encuentran en la Tabla No.1

2. Con la fuente de tensión fija a 20V, se varía R_x desde 1200 ohm hasta 10 ohm en pasos de 100, las lecturas tomadas se encuentran en la Tabla No.2

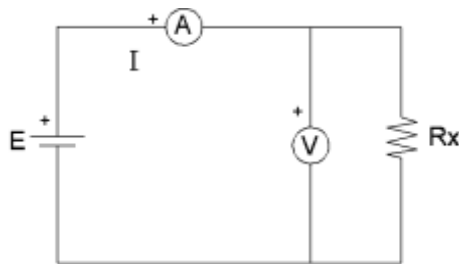


Tabla No.1:

V(volts)	0	2	4	6	8	10	15	20	25	30	35	40	45	50
I(ma)	0	1,67	3,33	5,00	6,67	8,33	12,50	16,67	20,83	25,00	29,17	33,33	37,50	41,67

Tabla No.2:

R(Ω)	1200	1100	1000	900	800	700	600	500	400	300	200	100
I(ma)	16,7	18,2	20	22	25	28,6	33,3	40	50	66,7	100	200

Análisis de ensayo No.1:

El circuito presentado consta de una fuente de tensión en serie con una resistencia, si aplicamos la ley de malla de Kirchhoff nos queda:

$$E = I \cdot R_x, \text{ de donde } I = E/R_x, \\ \text{como } R_x = 1.200, \text{ entonces} \\ I = E/1200, E \text{ será el valor del voltímetro } V$$

Análisis de ensayo No.2:

Es el mismo circuito del primer ensayo es decir se cumple que: $I = E/R_x$ como $E = 20V$

entonces $I = 20/R_x$

Ejercicios Propuestos por el Profesor

Bibliografía:

1. Fundamentos Scilab y aplicaciones, de Andrés Alfonso caro, Cesar Valero.
2. Introducción A Scilab (Programa De Cálculo Numérico) Febrero/Marzo 2005, Laboratorio De Computación de Alto Desempeño (Icad), Facultad de ingeniería, Universidad nacional de asunción.
3. Manual de Introducción al Tratamiento de Señales con SCILAB para usuarios de MATLAB. Programa de Doctorado en Automática E Informática Industrial, por Bernardo A. Delicado
4. Scilab. Computación Numérica Bajo Linux Y Windows, por Andrés Jiménez Jiménez. Titulado Superior de Apoyo a Investigación, Universidad de Cádiz.

Páginas de Internet: www.scilab.org

Estaré muy agradecida por los comentarios, sugerencias o correcciones enviadas a: dsevilla@cantv.net



REPÚBLICA BOLIVARIANA DE VENEZUELA
MINISTERIO DEL PODER POPULAR PARA LA
EDUCACIÓN SUPERIOR
INSTITUTO UNIVERSITARIO EXPERIMENTAL
DE TECNOLOGIA DE LA VICTORIA
LA VICTORIA- ESTADO ARAGUA
Comisión Académica del Programa
Nacional de Formación de Electricidad



CÁLCULO NUMÉRICO

PARA

SISTEMAS ELÉCTRICOS

PRÁCTICA I: INTRODUCCIÓN A LA COMPUTACIÓN

Realizado por: Prof. DORIS E. SEVILLA
Revisado por: Prof. JESUS A. PEREZ

La Victoria Octubre de 2007

TEMARIO:

1. Conceptos generales de la Algoritmia:

1.1 Algoritmo

1.2 Tipos de algoritmos

1.3 Partes de un algoritmo.

1.4 Características.

1.5 Formas de representar un algoritmo.

2. Programación en Scilab:

2.1 El Editor

2.2 Guiones (script)

3. Herramientas de programación:

3.1 Comparaciones y operadores lógicos

3.2 Ordenes de control

4. Ejemplos.

5. Ejercicios.

OBJETIVO:

1. Manejar conceptos generales de la algoritmia

2. Manejar herramientas básicas de programación

3. Resolver problemas haciendo uso de algoritmos con Scipad

1. CONCEPTOS GENERALES DE LA ALGORITMIA:

1.1 Algoritmo: es un conjunto de pasos precisos, definidos y finitos que conducen a la solución de un problema. Por ejemplo: el desarrollo de las actividades diarias en nuestra vida cotidiana corresponde a un algoritmo.

Ejemplo: algoritmo para comprar las entradas a un concierto.

1. Inicio
2. Llegar al sitio donde se venden las entradas al concierto.
3. Seleccionar el sector donde se desea presenciar el concierto.
4. Hacer la cola
5. Esperar el turno
6. Solicitar las entradas:
 - Si las hay
 7. Entregar el dinero
 8. Esperar entradas y diferencia de pago.
 9. Retirarse.
 - Si no hay
 10. Escoger otro sector e ir a 6 o retirarse

1.2 Tipos de algoritmos:

1.2.1. Computacionales: Ejecutados por una computadora.

1.2.2. No computacionales: ejecutados por el ser humano.

Para ejecutar un algoritmo computacional se utilizan los lenguajes de programación, el algoritmo expresado en un determinado lenguaje se denomina **programa**.

1.3 Partes de un algoritmo: todo algoritmo debe obedecer a la estructura básica de un sistema es decir, entrada, proceso y salida.

1.3.1 Entrada: corresponde a los datos necesarios que requiere el proceso para ofrecer unos resultados esperados.

1.3.2 Proceso: pasos necesarios para obtener la solución del problema.

1.3.3 Salida: resultados arrojados por el proceso como solución.

1.4 Características:

Las características fundamentales que debe cumplir un algoritmo son:

1.4.1 Precisión: Indica el orden de realización de cada paso dentro del proceso.

1.4.2 Definición: Indica la exactitud y consistencia de los pasos descritos en el proceso, si el algoritmo se prueba dos veces, en estas dos pruebas, se debe obtener el mismo resultado.

1.4.3 Finitud: Indica el número razonable de pasos, los cuales deben conllevar a la finalización del proceso y producir un resultado en un tiempo finito.

Ejemplo: Realice un algoritmo para realizar una arepa (sin medidas), especifique entrada, proceso y salida.

Entrada:	Harina, agua, sal y recipiente
Proceso:	1.Vierta en el recipiente agua y sal al gusto 2. Añada lentamente la harina. 3. Amase continuamente hasta compactar la masa. 4. Deje reposar por 5 minutos. 5. Déle forma con la mano. 6. Colóquelas en un budare, horno o sartén caliente. 7. De vueltas hasta obtener el cocimiento adecuado.
Salida:	Arepa

Ejercicios:

1. Realice un algoritmo para realizar una torta (sin medidas), especifique entrada, proceso y salida, estudie si cumple con las características.
2. Realice un algoritmo para comprar las entradas a un juego de fútbol especifique entrada, proceso y salida, estudie si cumple con las características.
3. Realice un algoritmo para llegar a su centro de estudio, especifique, entrada, proceso y salida, estudie si cumple con las características.

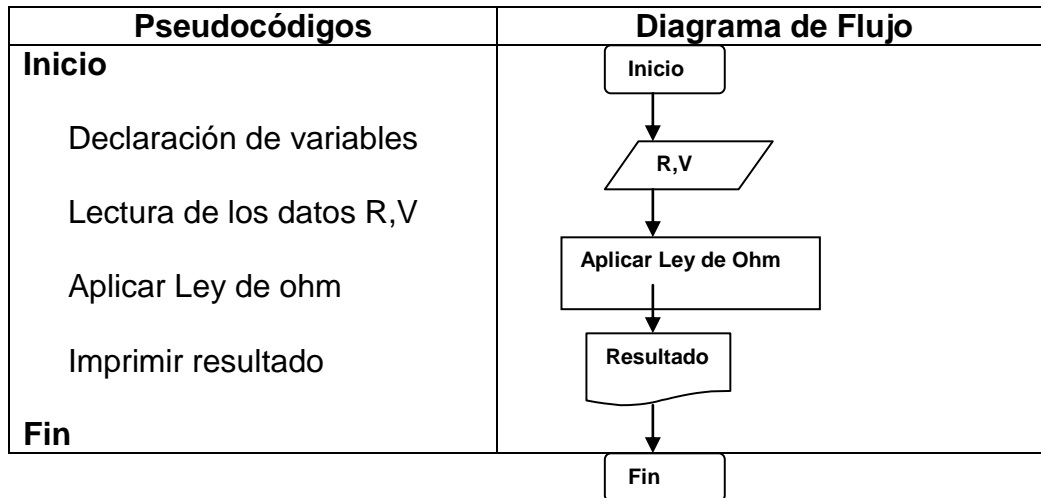
1.5 Formas de representar un algoritmo:

1.5.1 Pseudocódigo: significa escribir las instrucciones del algoritmo en lenguaje natural, tal y como lo expresamos de manera cotidiana, este procedimiento facilita su escritura en los lenguajes de programación.

1.5.2 Diagramas de flujo o flujogramas: son representaciones gráficas de los pasos necesarios que conllevan a la solución algorítmica de un problema. Para diseñarlos se utilizan determinados símbolos o figuras que representan una acción dentro del procedimiento. Estos símbolos se han normalizado o estandarizado para representar los pasos del algoritmo. Cada paso se representa a través del símbolo adecuado, que se van uniendo con

flechas, denominadas líneas de flujo, que a su vez indican el orden en que los pasos deben ser ejecutados.

Ejemplo: Realice un algoritmo en sus dos representaciones para el cálculo de la corriente en una resistencia dado el voltaje en la misma.



Ejercicios. Diseñe un algoritmo en sus dos representaciones que resuelva los siguientes problemas:

1. Pagar un recibo de teléfono.
2. Enviar un mensaje a través de un teléfono celular.
3. Calcular el valor de una resistencia dado I, V.
4. Sumar dos números enteros.

2. PROGRAMACIÓN EN SCILAB

En Scilab hay dos tipos de programas: los guiones o libretos (scripts) y las funciones. Un guión es simplemente una secuencia de órdenes de Scilab. No tiene parámetros ("argumentos") de entrada ni de salida. En cambio una función si los tiene.

Por otro lado, las variables definidas en un guión son globales, es decir, después del llamado del guión estas variables siguen existiendo. En cambio en una función, las variables definidas dentro de la función dejan de existir una vez analizada la ejecución de la función, son variables locales.

2.1 El Editor de Scilab: Scilab trae un editor de texto llamado Scipad incluido en el entorno de Scilab, se activa mediante la opción Editor de la barra de menú de Scilab o también puede iniciarse desde la línea de comandos de Scilab, ejecutando `scipad()` o `scipadf`.

* La tecla F2 permite guardar el archivo.

* La tecla F5 guarda el archivo y ejecuta las líneas del programa escrito en Scipad dentro de Scilab.

- * Doble click en el botón izquierdo del ratón selecciona la palabra que este cubriendo el cursor.
- * Con el botón derecho del ratón aparecerá un menú con varias opciones de edición.
- * Shift + botón derecho del ratón aparece el menú que permite seleccionar la ejecución del programa.
- * Ctrl. + botón derecho del ratón aparece un menú que permite modificar algunas características de formato y visualización.
- * Ctrl. + la tecla (+) incrementa el tamaño de la fuente en todo el entorno de Scipad.
- * Ctrl. + la tecla (-) disminuye el tamaño de la fuente en todo el entorno de Scipad.
- * Para cargar el programa o script dentro de Scilab se puede usar Ctrl.+Alt+L y es análogo a llamar el archivo con el contenido del script usando el comando exec desde Scilab.

2.2 Guiones (script): Un guión es simplemente un archivo ASCII en el que hay una sucesión de órdenes de Scilab. Generalmente tiene la extensión .sce pero eso no es obligatorio. Puede estar colocado en cualquier carpeta. Ya sabiendo lo que se va a hacer, con Scipad o con un editor cualquiera, creamos el archivo:

Ejemplo 2.2.1

1. Abra un Scipad, con un click en Editor de la barra de menú del Ventana de comandos.

2. Diseñe un programa que calcule la hipotenusa de un triangulo:

```
// MI PRIMER GUION
```

```
// PROGRAMA QUE PERMITE CALCULAR LA HIPOTENUSA
```

```
// DE UN TRIANGULO, DADO SUS LADOS, A Y B
```

```
A=input("Introduzca el valor de A");
```

```
B=input("Introduzca el valor de B");
```

```
disp("El valor de la Hipotenusa es:")
```

```
H=sqrt(A^2+B^2)
```

3. Guarde el programa: ir al menú File, luego Save as, se elige la carpeta donde se desea guardar, se escribe el nombre seguido del la extensión .sce, por ejemplo

```
c:\misdocumentos\hipotenusa.sce
```

4. Dar la orden exec c:\misdocumentos\hipotenusa.sce también se puede hacer en Windows por medio de la barra de menú con las opciones File y después Exec. Subiendo y bajando de nivel se busca la carpeta adecuada y se hace doble clic con el botón derecho del ratón en el archivo hipotenusa.sce

En Linux se hace de manera muy parecida: en la barra de menú File, enseguida File Operations y finalmente Exec después de haber escogido el archivo deseado. Si se usa SciPad, se obtiene el mismo resultado con la barra de menú (de SciPad) mediante la opción Execute y después Load into Scilab.

Si se desea, se puede editar el archivo, hacer algunos cambios, guardarlos y de nuevo activar el guión mediante exec.

3. Herramientas de programación

3.1 Comparaciones y operadores lógicos

<	menor
<=	menor o igual
>	mayor
>=	mayor o igual
==	igual
~=	diferente
<>	diferente
&	y
	o
~	no

3. **Ordenes de control:** Las principales estructuras de control de Scilab son: if, select y case, for, while. Otras ordenes relacionadas con el control de flujo son: break, return equivalente a resume, abort.

3.2.1 If: La estructura condicional simple, *Si-Entonces* (o IF-THEN, en inglés), permite evaluar una condición determinada y si se cumple la condición ejecuta una o varias instrucciones. Si la condición es falsa, entonces no se realizará ninguna acción.

Una forma sencilla de la escritura de la escritura if es la siguiente:

if condición then

...

end

La palabra then puede ser reemplazada por una coma o por un cambio de línea.

Obviamente también existe la posibilidad else(If-Then-Else/ Si-Entonces-Si no): permite evaluar una condición, la cual puede tener dos acciones, cuando se cumple, y cuando no se cumple.

if condición

...

else

...

end

3.2.2 Selec y case: La estructura de decisión múltiple o selectiva evaluará una expresión que podrá tomar un conjunto de valores distintos 1, 2, 3, 4, n, es decir hasta n valores. Según la elección del valor de la condición establecida, se realizará un conjunto de instrucciones.

La forma general es:

select variable

case valor1 then

...

```

case valor2 then
...
case valor3 then
...
case valor4 then
...
else
...
end

```

Al igual que la estructura if, la palabra then puede ser reemplazada por una coma o por un cambio de línea.

3.2.3 For: Se realizará una secuencia de acciones un número predeterminado de veces.

La estructura for tiene la siguiente forma:

```
For n = lim1:incr:lim2
```

```

...
end

```

Esto quiere decir que la variable n empieza en el límite inferior lim1 , después va incrementar el valor en el incremento incr (puede ser negativo). Si el incremento es 1, este se puede suprimir.

3.2.4 While: Mientras se cumpla la condición que desencadena el proceso, las instrucciones que se encuentran dentro del ciclo While se realizarán.

La forma general es:

```
while condición
```

```

...
end

```

La orden break permite la salida forzada (en cualquier parte interna) de un bucle for o de un bucle while.

La orden return permite salir de una función antes de llegar a la última orden. Todos los parámetros de salida o resultados deben estar definidos con anterioridad.

Otra orden que sirve para interrumpir una función, en este caso interrumpiendo la evaluación, es abort.

4. Ejemplos:

4.1 Dado dos números determine cual es el mayor y muéstrelo.

```

//Ejemplo No.1: Dado dos números determine cual es mayor
disp('Programa que permite encontrar el numero mayor entre dos números
introducidos por el usuario a través del teclado')
A=input('Introduzca el Primer Numero, A=');
B=input('Introduzca el Segundo Numero, B=');
if A==B
disp('Los Números introducidos son Iguales')
elseif A>B
disp('A es el numero MAYOR')
else

```

```
disp('B es el numero MAYOR')
end
```

4.2 Ordene de mayor a menor valor el conjunto de 3 resistencias de carbón introducidos por teclado.

//Ejemplo No.2: Introducidas 3 valores de resistencias ordénelas de mayor a menor

```
disp('Programa que permite ordenar 3 valores de resistencias diferentes introducidos por el usuario a traves del teclado')
```

```
R1=input('Introduzca el valor de R1=');
```

```
R2=input('Introduzca el valor de R2=');
```

```
while R1==R2
```

```
disp('R1 y R2 son iguales redefina los valores de las Resistencias')
```

```
R1=input('Introduzca el valor de R1=');
```

```
R2=input('Introduzca el valor de R2=');
```

```
end
```

```
R3=input('Introduzca el valor de R3=');
```

```
while R3==R1 | R3==R2
```

```
disp('Redefina el valor de R3 ya que coincide con uno de los valores anteriormente introducido')
```

```
R3=input('Introduzca el valor de R3=');
```

```
end
```

```
disp('Se ordenaran la mayor a menor de la siguiente forma Ra,Rb,Rc')
```

```
if R1>R2 & R1>R3
```

```
    Ra=R1
```

```
    if R2>R3
```

```
        Rb=R2
```

```
        Rc=R3
```

```
    else
```

```
        Rb=R3
```

```
    end
```

```
else
```

```
    if R2>R3 & R2>R1
```

```
        Ra=R2
```

```
        if R1>R3
```

```
            Rb=R1
```

```
            Rc=R3
```

```
        else
```

```
            Rb=R3
```

```
            Rc=R2
```

```
    end
```

```
else
```

```
    Ra=R3
```

```
    if R1>R2
```

```
        Rb=R1
```

```
        Rc=R2
```

```
    else
```

```
        Rb=R2
```

```
        Rc=R1
```

```
    end
```

```
end
```

end

4.3 Desarrolle un algoritmo que calcule la comisión a los empleados, dado su sueldo base de 480.000 Bs. y venta del día, con las siguientes condiciones si la venta es mayor a 500.000Bs. obtendrá una comisión del 7% y si es menor del 4%

```
disp('Calculo de la comisión a los empleados según su venta')
SB=480000;
Venta=input('Introduzca la venta del día, Venta=')
if Venta>=500000
    Comisión=0.07*Venta
    Sueldo=SB+Comisión
else
    Comisión=0.04*Venta
    Sueldo=Comisión+SB
end
```

5. Ejercicios:

1. Dada la siguiente ecuación: $I(t) = 100 \sin(628t - 60^\circ)$ desarrolle un algoritmo que dibuje cuando esta toma valores positivos y cuando toma valores negativos pero en graficas separadas.
2. Desarrolle un algoritmo que permita determinar si N es par o impar.
3. Desarrolle un algoritmo que determine cuantos dígitos enteros tiene un número.
4. Desarrolle un algoritmo que resuelva N/M.
5. Desarrolle un algoritmo que encuentre los instante de tiempos donde el voltaje alcanza su valor máximo de 120V, trabajando a una frecuencia a 60Hz, el intervalo de estudio esta comprendido [0-2] seg.

$$V(t) = 120 \sin(\omega t - \phi)$$

6. Ejercicios propuestos por el Profesor

BIBLIOGRAFIA

1. Introducción a SCILAB, Héctor Manuel Mora Escobar Departamento de Matemáticas, Universidad Nacional de Colombia Bogota mayo del 2005.
2. Fundamentos Scilab y aplicaciones, de Andrés Alfonso caro, Cesar Valero.
3. Programa Nacional de Formación en Sistemas e Informática, MISION SUCRE

Estaré muy agradecida por los comentarios, sugerencias o correcciones enviadas a: dsevilla@cantv.net



REPÚBLICA BOLIVARIANA DE VENEZUELA
MINISTERIO DEL PODER POPULAR PARA LA
EDUCACIÓN SUPERIOR
INSTITUTO UNIVERSITARIO EXPERIMENTAL
DE TECNOLOGIA DE LA VICTORIA
LA VICTORIA- ESTADO ARAGUA
Comisión Académica del Programa
Nacional de Formación de Electricidad



CÁLCULO NUMÉRICO

PARA

SISTEMAS ELÉCTRICOS

PRÁCTICA I: INTRODUCCIÓN A LA COMPUTACIÓN

Realizado por: Prof. DORIS E. SEVILLA
Revisado por: Prof. JESUS A. PEREZ

La Victoria Octubre de 2007

TEMARIO:

1. Sistemas de Ecuaciones Lineales

1.1 Sistema de múltiples ecuaciones con coeficientes iguales.

2. Una Ecuación No Lineal

3. Sistema de Ecuaciones No Lineales

4. Integración Numérica

5. Derivación Numérica

6. Interpolación

7. Ecuaciones Diferenciales Ordinarias

8. Sistema de Ecuaciones Diferenciales

OBJETIVO:

1. Resolver sistemas de ecuaciones lineales.

2. Derivar e integrar funciones aplicando Scilab.

3. Resolver circuitos eléctricos de corriente continua aplicando Scilab.

4. Resolver ecuaciones diferenciales de primer orden de un circuito RC y RL.

1. Sistemas de Ecuaciones Lineales:

La función linsolve en SCILAB, nos permite dar solución a un sistema de ecuaciones lineales, resolviéndolo como un sistema matricial, la forma general de esta función es:

[x0,kerA]=linsolve(A,B,[x0])

Donde:

A es la matriz de n_a filas por m_a columnas,

B es un vector de n_a filas,

x0 el vector solución,

kerA es la matriz $m_a * k$ soluciones.

Todas las posibles soluciones de linsolve son de la forma:

$$\mathbf{A} \cdot \mathbf{x} + \mathbf{B} = \mathbf{0}$$

Dependiendo del un numero de incógnitas y ecuaciones, la función linsolve producirá diferentes resultados.

Ejemplo1.1: Resuelva el siguiente sistema de ecuaciones:

a) $\begin{cases} 2x + y = 1 \\ 6x - 3y = 21 \end{cases}$

Lo primero que debemos hacer es ordenar e igualar todas las ecuaciones a cero (0).

$$\begin{cases} 2x + y - 1 = 0 \\ 6x - 3y - 21 = 0 \end{cases}$$

Ahora lo transformamos a la forma **(Ax + B = 0)**

$$A = \begin{pmatrix} 2 & 1 \\ 6 & -3 \end{pmatrix} \quad X = \begin{pmatrix} x \\ y \end{pmatrix} \quad B = \begin{pmatrix} -1 \\ -21 \end{pmatrix}$$

En SCILAB:

```
-->A=[2 1;6 -3];B=[-1;-21];
```

```
-->[x0,KerA]=linsolve(A,B)
```

KerA =

[]

x0 =

2.

- 3.

-

La solución es $x=2$; $y=-3$, como $\text{KerA}=0$, hay una única solución.

□

b) $\begin{cases} 2x - 3y = -2 \end{cases}$

$$\begin{aligned} z + 2y - x/2 &= 3 \\ 4x + 5y - 2z &= 12 \end{aligned}$$

Se ordena las ecuaciones y se igualan a cero:

$$\begin{cases} 2x - 3y + 0z + 2 = 0 \\ -x/2 + 2y + z - 3 = 0 \\ 4x + 5y - 2z - 12 = 0 \end{cases}$$

En SCILAB:

-->A=[2 -3 0;-1/2 2 1;4 5 -2]; B=[2;-3;-12];

-->[x0,KerA]=linsolve(A,B)

KerA =

[]

x0 =

1.3333333

1.5555556

0.5555556

La solución única es x=1.333; y=1.555; z=0.555.

$$\text{c) } \begin{cases} S + 4R - 8T - 49 = 0 \\ 8T - 39 + S = \\ 8 + T = 0 \end{cases}$$

Se ordenan las ecuaciones:

$$\begin{cases} S + 4R - 8T - 49 = 0 \\ S + 0R + 8T - 39 = 0 \\ 0S + 0R + T + 8 = 0 \end{cases}$$

En SCILAB:

-->A=[1 4 -8;1 0 8;0 0 1]; B=[-49;-39;8];

-->[x0,KerA]=linsolve(A,B)

KerA =

[]

x0 =

103.

- 29.5

- 8.

La solución única es S=103; R= -29.5; T= -8

Ejercicios 1.1: Encuentre solución a los siguientes sistemas de ecuaciones:

a.- $\begin{cases} A+B = 4 \\ -B - A = -4 \end{cases}$	b.- $\begin{cases} R + 4S - T = 13.5 \\ -2S + R/2 = -23/4 \\ 3R - 2S + T = 11/2 \end{cases}$	c.- $\begin{cases} B + Z = A \\ B - A - 1 = 0 \\ 2B - 4 + A = 2Z \end{cases}$

1.1 Sistema de múltiples ecuaciones con coeficientes iguales: para encontrar la solución de un sistema de ecuaciones lineales que cuenta con múltiples ecuaciones de coeficientes iguales, se debe realizar como primer paso la transformación de los sistemas a la forma $Ax=B$, luego se crea la matriz resultante escribiendo $x=\text{inv}(A)*B$.

Ejemplo1.1.1: Encuentre la solución a los siguientes sistemas de ecuaciones:

$$\begin{cases} 2x+5y-3z= 12 \\ 6x+6y+z= 13 \\ x+y+z= 2 \end{cases} ; \quad \begin{cases} 2x+5y-3z= 3 \\ 4x+6y+z= 27 \\ x+y+z= 9 \end{cases} ; \quad \begin{cases} 2x+5y-3z= -6 \\ 4x+6y+z= 2 \\ x+y+z=2 \end{cases}$$

Se transforman los sistemas a la forma $Ax=B$, es decir:

$$A = \begin{pmatrix} 2 & 5 & -3 \\ 6 & 6 & 1 \\ 1 & 1 & 1 \end{pmatrix}; \quad B = \begin{pmatrix} x1 & x2 & x3 \\ y1 & y2 & y3 \\ z1 & z2 & z3 \end{pmatrix}; \quad C = \begin{pmatrix} 12 & 3 & -6 \\ 13 & 27 & 2 \\ 2 & 9 & 2 \end{pmatrix}$$

En SCILAB:

```
-->A=[2 5 -3;4 6 1;1 1 1]; B=[12 3 -6;13 27 2;2 9 2];
```

```
-->x=inv(A)*B
```

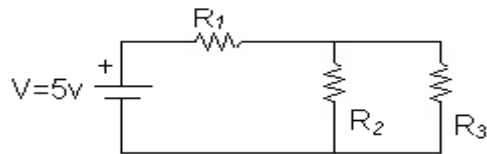
```
x =
```

```
2.  6.  0.
1.  0.  0.
-1.  3.  2.
```

```
-->
```

Las soluciones son: $x1=2, y1=1, z1=-1$; $x2=6, y2=0, z2=3$; $x3=0, y3=0, z3=2$

Ejemplo1.1.2: Hallar el valor de las corrientes, del circuito de la siguiente figura: $R1=10\Omega$, $R2=30\Omega$, $R3=60\Omega$



Aplicando leyes de kirchoff (métodos de las mallas):

I: $5 - V_{R1} - V_{R2} = 0 \rightarrow V_{R1} + V_{R2} - 5 = 0$

II: $5 - V_{R1} - V_{R3} = 0 \rightarrow V_{R1} + V_{R3} - 5 = 0$

III: $V_{R2} = V_{R3} \rightarrow V_{R2} - V_{R3} = 0$

Scilab:

$A=[1 \ 1 \ 0; 1 \ 0 \ 1; 0 \ 1 \ -1]; B=[-5;-5;0];$

$[x0]=\text{linsolve}(A,B)$

//////////

$[x0]=\text{linsolve}(A,B)$

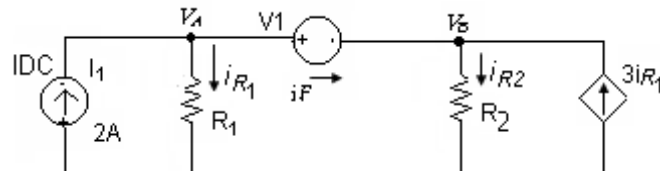
$x0 =$

3.3333333

1.6666667

1.6666667

Ejemplo1.1.3: Encuentre las corrientes en el circuito, cuando $R_1=10\Omega$, $R_2=20\Omega$, $V_1=3V$, $I_{DC}=2A$:



Por nodos: Se tienen dos incognitas: los voltajes de los nodos V_A y V_B

Panteando las ecuaciones en los nodos:

Nodo A: $i_1 = i_F + i_{R1}$

Nodo B: $i_F = i_{R2} + 3i_{R1}$

Fuente V1: $V_1 = V_A - V_B$

Se obtiene el siguiente sistema de ecuaciones lineales:

$4 V_A - V_B = -40$

$V_A - V_B = 3$

Usando scilab:

--> $c=[4 \ -1; 1 \ -1]; r=[-40; 3];$

--> $[V, \text{kerA}]=\text{linsolve}(c,r)$

$\text{kerA} =$

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

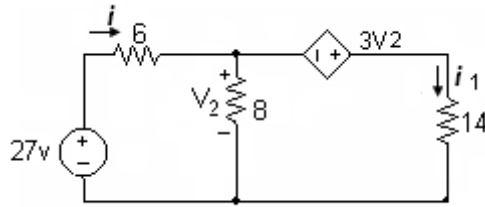
$V =$

14.333333

17.333333

Solución: $V_A=14.3V$ y $V_B=17.3V \Rightarrow i_{R1}=1.43A$, $i_F=3.43A$ y $i_{R2}=1.73^a$

Ejemplo 1.1.4: Dado el siguiente circuito, encuentre i , i_1 y v_2



Aplicando las leyes de corriente de kirchhoff:

$$27 = 6i + V_2$$

$$V_2 = -3V_2 - 14i_1$$

$$i = V_2/8 + i_1$$

Ordenando las ecuaciones:

$$\begin{cases} 6i + V_2 + 0i_1 - 27 = 0 \\ 0 + 4V_2 - 14i = 0 \\ i - 1/8V_2 - i_1 = 0 \end{cases}$$

Scilab:

```
--> c=[6 1 0;0 4 -14;1 -1/8 -1];r=[-27;0;0];
```

```
--> [S,kerA]=linsolve(c,r)
```

```
kerA =
```

```
[]
```

```
S =
```

```
3.2010309
```

```
7.7938144
```

```
2.2268041
```

Solución: $i=3,20A$; $V_2=7,79V$; $i_1=2,22A$

2. Una Ecuación No Lineal:

Para resolver una ecuación o un sistema de ecuaciones no lineales en SCILAB, se puede usar el comando: **fsolve**

```
[x [,v [,info]]]=fsolve(x0,fct [,fjac] [,tol])
```

Donde:

x0 : vector de valores reales conteniendo valores iniciales del argumento de la función

fct : definición de la función cuya solución se desea encontrar

fjac : definición del jacobiano de la función a resolver (Opcional)

tol : tolerancia del proceso iterativo de búsqueda de la solución. El proceso iterativo termina cuando el error relativo de la solución es menor o igual que tol. (1×10^{-10} es el valor por defecto) (Opcional)

x : vector real que contiene el valor final de la solución aproximada.

v : vector real que contiene el valor de la función en x, (Opcional).

info : indicadores de culminación, (Opcional)

0 : parámetros de entrada inapropiados

1 : error relativo satisface condiciones de tolerancia

2 : número máximo de iteraciones alcanzado

3 : tol muy pequeña. No se puede mejorar aproximación a la solución

4 : iteración no se encuentra progresando.

Fsolve está basado en el método de resolución iterativo Newton-Raphson.

Ejemplo 2.1: Encuentre el valor de x, para satisfacer la ecuación:

$$\frac{x - e^x}{1 + x^2} - \sin x + 1 = 0$$

En Scilab realizamos un archivo.sce, lo cargamos y obtenemos el resultado

```
function fx=func21(x)
fx=(x-exp(x))/(1+x*x)-sin(x)+1// definimos la ecuación no lineal
endfunction
r=fsolve(0,func21) // encuentra el valor de x que satisface la ecuación en 0.
```

Cargamos la función:

```
-->exec('C:\Archivos de programa\scilab-4.0\bin\EJEMPLO2.1.sce');disp('exec
done');
r =
0.
```

Ejemplo 2.2: Encuentre el valor de x, para que la ecuación tome los siguientes valores: a=-1, 1, 2.

$$\cos x + 0.1 + \frac{e^x}{x} = a$$

En Scilab realizamos un archivo.sce, lo cargamos y obtenemos el resultado

```
function fx=func22(x)
fx=cos(x)+0.1+(exp(x)/x)// definimos la ecuación no lineal
endfunction
r1=fsolve(-1,func22) // encuentra el valor de x que satisface la ecuación en -1
r2=fsolve(1,func22) // encuentra el valor de x que satisface la ecuación en 1.
r3=fsolve(2,func22) // encuentra el valor de x que satisface la ecuación en 2.
```

Cargamos la función:

```
-->exec('C:\Archivos de programa\scilab-4.0\bin\EJEMPLO2.2.sce');disp('exec
done');
r1 =
- 1.5290305
r2 =
1.5000392
r3 =
- 1.5290305
exec done
```

3. Sistema de Ecuaciones No Lineales:

Dada una función $f: \mathbb{R}^n \rightarrow \mathbb{R}^n$ se desea encontrar un vector x tal que $f(x)=0$. Aquí 0 es el vector nulo. Por ejemplo,

$$\begin{aligned} x_1^2 + x_1 x_2 + x_3 - 30 &= 0, \\ 2x_1 + 3x_2 x_3 &= 0, \\ x_1 + x_2 + x_3^2 - 10x_3 + 1 &= 0. \end{aligned}$$

La función f debe estar definida en una función de Scilab, donde, dado un vector columna x, calcule el vector columna $f(x)$. Una vez escrita la función y cargado el archivo donde esta f, se utiliza **fsolve**.

En Scilab realizamos un archivo.sce, lo cargamos y obtenemos el resultado:

```
function fx=f31(x)
    fx=zeros(3,1)
    fx(1)=x(1)^2+x(1)*x(2)+x(3)-30
    fx(2)=2*x(1)+3*x(2)*x(3)
    fx(3)=(x(1)+x(2)+x(3))^2-10*x(3)+1
endfunction
[x,fx,info]=fsolve([3 4 5]',f31)
```

Cargamos la función:

```
-->exec('C:\Archivos de programa\scilab-4.0\bin\EJEMPLO31.sce');disp('exec
done');
-->[x,fx,info]=fsolve([3 4 5]',f31)
info =
    1.
fx =
    1.0D-10 *
    - 0.0344258
    0.2518874
    0.0115374
x =
    16.257922
    - 14.458775
    0.7496219
```

4. Derivación:

La derivada de una función f es aquella función, denotada por f' , cuyo valor en un numero cualquiera x del dominio de f esta dado por:

$$f'(x) \equiv \lim_{\Delta \rightarrow 0} \frac{f(x+\Delta) - f(x)}{\Delta}, \text{ si este límite existe.}$$

Para encontrar la derivada de un polinomio en Scilab, se debe utilizar el comando, **derivat** de la siguiente forma:

pd=derivat(p)

Donde: p, es el polinomio a derivar.

Ejemplo 4.1: Derive la siguiente función: $f(x) \equiv x^2 + x$

```
-->x=poly(0,'x');// definimos la variable de la función, en este caso "x"
-->Df=derivat(x^2+x)
Df =
    1 + 2x
-->
```

Ejemplo 4.2: Derive la siguiente función: $g(x) \equiv \frac{x+8}{x^3}$

```
-->x=poly(0,'x');// definimos la variable de la función, en este caso "x"
```

```
-->Dg=derivat(x+8/x^3)
```

$$Dg = \frac{-24x^2 + x^6}{x^6}$$

Ejemplo 4.3: Derive la siguiente función: $Z(a) \equiv \frac{a^8 + a^5 + a^2 - 8}{-a^{10} + a^3}$

```
-->a=poly(0,'a');// definimos la variable de la función, en este caso "a"
```

```
-->Dz=derivat((a^8+a^5+a^2-8)/(-a^10+a^3))
```

$$Dz = \frac{24a^2 - a^4 + 2a^7 - 80a^9 + 5a^{10} + 8a^{11} + 5a^{14} + 2a^{17}}{a^6 - 2a^{13} + a^{20}}$$

```
-->
```

Ejemplo 4.4: Derive la siguiente función: $z(x) \equiv \frac{1}{x^5} + x^2 \cdot \frac{8}{x^2}$

```
-->x=poly(0,'x');// definimos la variable de la función, en este caso "x"
```

```
-->Dz=derivat((1/x^5)+x^2+(8/x^2))
```

$$Dz = \frac{-5x^{-4} - 16x + 2x}{x^{10}}$$

```
-->
```

5. Derivación numérica:

Para obtener una aproximación numérica de la derivada de una función en un punto se debe usar **derivative**.

Ejemplo 5.1 Encuentre el valor de la derivada en el punto $x=\pi$.

```
-->deff('[y]=f51(x)', 'y=sin(x)');
```

```
-->d=derivative(f51,%pi)
```

$$d = -1.$$

```
-->
```

Ejemplo 5.2 Encuentre el valor de la derivada en el punto $x=1$.

```
-->deff('[y]=f52(x)', 'y=sqrt(x+8)');
```

```
-->d=derivative(f52,1)
```

$$d = 0.1666667$$

```
-->
```

Ejemplo 5.3 Encuentre el valor de las derivadas en cada uno de los puntos:

$$a. f(x) = \arccos(3x^2 - 6x + 1) \dots \text{en} \dots x = \pi$$

$$b. f(x) = \frac{x+4}{1-2x} \dots \text{en} \dots x = 2$$

$$c. f(x) = e^{x^2} \dots \text{en} \dots x = 1$$

$$d. f(x) = \sin(2x) \dots \text{en} \dots x = \pi/2$$

Archivo .sce para definir las funciones y los puntos donde se necesita conocer la derivada:

```
deff('[y]=f531(x)', 'y=acos(3*x^2-6*x+1)');
da=derivative(f531,2*%pi)
//////////
deff('[y]=f532(x)', 'y=(x+4)/(1-2*x)');
db=derivative(f532,2)
//////////
deff('[y]=f533(x)', 'y=exp(x^2)');
dc=derivative(f533,1)
//////////
deff('[y]=f534(x)', 'y=sin(2*x)');
dd=derivative(f534,%pi/2)
```

Luego se carga el archivo:

```
-->exec('C:\Archivos de programa\scilab-4.0\bin\ejemplo 5.3.sce');disp('exec
done');
da =
    0.3878515i
db =
    1.
dc =
    5.4365637
dd =
    - 2.
exec done
-->
```

6. Integración Numérica:

Para obtener una aproximación del valor de una integral definida, por ejemplo:

$$\int_2^8 \frac{1}{x+8} dx$$

Se utiliza el comando **intg**. Para eso es necesario definir en Scilab la función que se va a integrar.

En Scilab:

```
-->deff('[y]=f2(x)', 'y=sqrt(x+8)');
-->l=intg(2,8,f2)
l =
    21.584816
```

También se puede definir una función en un archivo .sce, se carga y se da la orden para integrar.

Ejemplo 6.1

```
//Archivo .sce para definir la función a integrar
function fx = f3(x)
fx=exp(x*x)
endfunction
//Luego se carga la función desde la ventana de comandos:
-->exec('C:\Archivos de programa\scilab-4.0\bin\EJEMPLO.sce');disp('exec
done');
exec done
-->l=intg(0.1,1,f3)
l = 1.3623174
```

Ejemplo 6.2 Integre las siguientes funciones:

$$a. \int_{-2}^2 (3x^3 - x^2 - 1) dx$$

$$b. \int_0^5 \log(\sqrt{x+2}) dx$$

$$c. \int_0^5 \frac{x+2}{x} dx$$

$$d. \int_0^{\pi} 2\sin x + 3\cos x dx$$

$$e. \int_{-2}^1 (x+1)^{\frac{2}{3}} dx$$

$$f. \int_{-\pi/2}^{\pi/2} (4\cos^3 x - 9\cos x) dx$$

$$g. \int_0^{\pi} \frac{3\tan x - 4\cos^2 x}{\cos x} dx$$

En Scilab: realizamos un archivo.sce con las funciones a integrar y luego la cargamos, para conocer los resultados:

```
deff('[y]=f53(x)', 'y=(3*x^3-x^2-1)');
la=intg(-2,2,f53)
///
deff('[y]=f54(x)', 'y=(log(sqrt(x+2)))');
lb=intg(0,5,f54)
///
deff('[y]=f55(x)', 'y=(x+2/(sqrt(x)))');

lc=intg(0,4,f55)
///
deff('[y]=f56(x)', 'y=(2*sin(x)+3*cos(x))');
ld=intg(0,%pi,f56)
///
deff('[y]=f57(x)', 'y=((x+1)^2/3)');
le=intg(-2,1,f57)
///
deff('[y]=f58(x)', 'y=(4*cos(x)^3-9*cos(x))');
lf=intg(-1*%pi/2,%pi/2,f58)
//
deff('[y]=f59(x)', 'y=((3*tan(x)-4*cos(x)^2)/(cos(x)))');
lg=intg(0,%pi,f59)
```

Cargamos este archivo:

```
-->exec('C:\Archivos de programa\scilab4.0\bin\Integrales.sce');disp('exec
done');
l1 =
```



```

- 9.3333333
l2 =
  3.6175383
l3 =
  16.
l4 =
  4.
l5 =
  1.
l7 =
- 12.666667
l8 =
  5256.5868

```

```

exec done
-->

```

7. Interpolación:

Cuando hay m puntos $(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$ se desea obtener una función interpolante, una función que pase por esos puntos, con el objetivo de evaluarla en otros valores x intermedios.

La función `interpIn` permite hacer interpolación lineal. Tiene dos parámetros, el primero es una matriz de dos filas. La primera fila tiene valores x_i . Deben estar en orden creciente. La segunda fila tiene los valores y_i . El segundo parámetro es un vector donde están los valores x en los que se desea evaluar la función interpolante (afín por trozos).

También se puede hacer interpolación utilizando funciones *spline*. Para hacer esto en Scilab, se requieren dos pasos. En el primero, a partir de una lista de $(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)$ se calculan las derivadas, en los puntos x_i , de la función *spline* interpolante. En el segundo paso se evalúa la función interpolante en una lista de puntos t_1, t_2, \dots, t_p

Ejemplo 7.1 Encuentre los valores de la función en los siguientes puntos: [-4:45], dado la siguiente tabla de valores:

x	y
1	1
10	30
20	-10
30	20
40	40

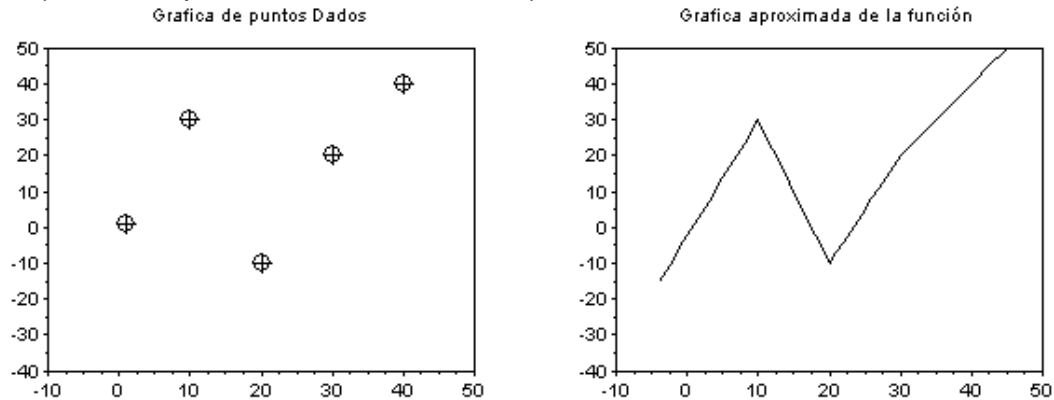
En Scilab:

```

x=[1 10 20 30 40];
y=[1 30 -10 20 40];
subplot(221)
plot2d(x',y',[-3],"011"," ",[-10,-40,50,50]);
xtitle('Grafica de puntos Dados')
yi=interpIn([x;y],-4:45);
subplot(222)
plot2d((-4:45)',yi',[1],"011"," ",[-10,-40,50,50]);

```

xtitle('Grafica aproximada de la función')



8. Ecuaciones Diferenciales Ordinarias:

SCILAB permite obtener soluciones numéricas a problemas de valor inicial de la forma:

$dy/dx = f(x,y)$, dada la condición inicial (x_0, y_0) .

El comando a ser utilizado es: **[y] = ode([type], y0, x0, x, f)**

Donde:

Type: argumento opcional que indica el método de resolución y puede ser 'adams' (Adams), 'rk' (Runge-Kutta), 'rkf' (Runge-Kutta modificado), 'fix', 'discrete', 'roots'

y_0 : vector conteniendo las condiciones iniciales

x_0 : valor inicial de la variable independiente

x: vector que contiene los valores de la variable independiente para los cuales la solución es calculada

f: definición de la función a ser integrada

La función **ode** evalúa aproximaciones de valor de y en valores t_1, t_2, \dots, t_p .

Ejemplo 8.1: Encuentre la solución a la siguiente ecuación diferencial:

$$y' = \frac{x+y}{x^2+y^2} - 4,$$

$$y(2) = 3.$$

En Scilab: realizamos un archivo.sce con las funciones a integrar y luego la cargamos desde la ventana de comando para conocer los resultados:

```
function fxy=func71(x,y)
```

```
fxy=(x+y)/(x*x+y*y)+4
```

```
endfunction
```

```
x0=2,y0=3,t=[2 2.5 2.9 3 3.5]';
```

```
yt=ode(y0,x0,t,func71)
```

Al cargar el archivo, se arrojan los resultados:

```
-->exec('C:\Archivos de programa\scilab-4.0\bin\EJEMPLO71.sce');disp('exec done');
```

```
x0 =
```

```
2.
```

```
y0 =
```

```
3.
```

```
yt =
    3.  5.148577  6.8295837  7.2467387  9.3199381
exec done
```

Ejemplo 8.2 Encuentre la solución de las siguientes ecuaciones diferenciales:

$$a \quad y' = 2x + 1, y(0) = 3$$

$$b \quad y' = x - 2x^3, y(2) = 1$$

$$c \quad y' = x(x^2 + 9)^{1/2}, y(4) = 0$$

Scilab-Scipad:

```
////////Función 8A: dy/dx=(2*x+1)////////
function fxy=func8a(x,y)
fxy=2*x+1
endfunction
x0=0;y0=3;t=[2 2.5 2.9 3 3.5]';
yt=ode(y0,x0,t,func8a)
////////Función 8b: dy/dx=(x-2)^3////////
function fxy=func8b(x,y)
fxy=(x-2)^3
endfunction
x0=2;y0=1;t=[2 2.5 2.9 3 3.5]';
yt=ode(y0,x0,t,func8b)
////////Función 8c: dy/dx=x(x^2 +9)^1/2 //////////
function fxy=func8c(x,y)
fxy=x*(x^2 +9)^1/2
endfunction
x0=-4;y0=0;t=[-4:0.5:4]';
yt=ode(y0,x0,t,func8c)
```

Al cargar el archivo, se arrojan los resultados:

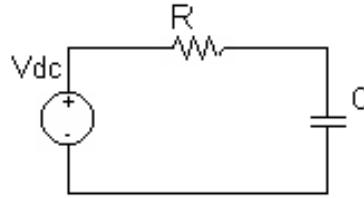
```
-->exec('C:\Archivos de programa\scilab-4.0\bin\edo.sce');disp('exec done');
yta =  9.0000006  11.750001  14.310001  15.000001  18.750001
ytb =  1.  1.0156255  1.1640255  1.2500005  2.2656255
ytc =
    0. - 21.679688 - 37.625 - 49.054688 - 57. - 62.304688 - 65.625 -
67.429688

- 68. - 67.429688 - 65.625 - 62.304688 - 57. - 49.054688 - 37.625

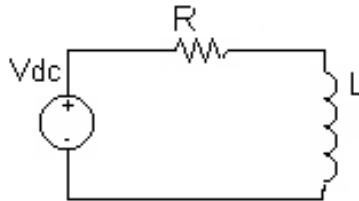
- 21.679688 - 6.952D-10
```

exec done

Ejemplo 8.3 Dado un circuito RC en serie, dibuje con ayuda de scilab la grafica de la corriente. Con los siguientes valores, $c=0.01f$; $R=10ohm$; $E=10v$



Ejemplo 8.4: Dado un circuito RL en serie, dibuje con ayuda de Scilab la grafica de la corriente:



Corriente inicial $i_0=2A$

9. Sistema de Ecuaciones Diferenciales:

Un sistema de ecuaciones diferenciales ordinarias de primer orden con sus condiciones iniciales se escribe, generalmente, en la forma: $dy/dx = f(x,y)$, $y(x_0)=y_0$, pero ahora, y , y' y y_0 son vectores en R^n ; $f: R^{1+n} \rightarrow R^n$.

Ejemplo 9.1: Resuelva el siguiente sistema de E.D:

$$y1' = \frac{2y1}{x} + x^3 y2,$$

$$y2' = -\frac{3}{x} y2$$

$$y1 \square \square - 1,$$

$$y2 \square \square 1$$

Scilab:

```
function fxy=func91(x,y)
fxy=zeros(2,1)
fxy(1)=2*y(1)/x+x^3*y(2)
fxy(2)=-3*y(2)/x
endfunction
x0=1,
y0=[-1 1]',
t=(1:0.2:2)';
yt=ode(y0,x0,t,func91)
```

```
-->exec('C:\Archivos de programa\scilab-4.0\bin\EJEMPLO91.sce');disp('exec done');
```

x0 =

1.

y0 =

- 1.

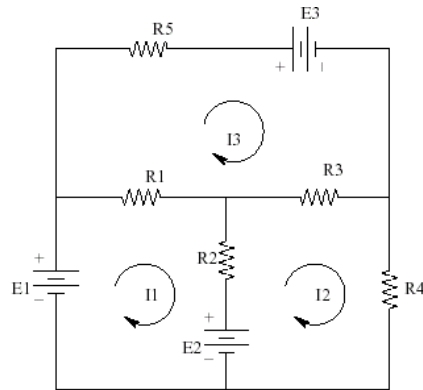
```

1.
yt =
- 1. - 1.1999999 - 1.3999999 - 1.5999998 - 1.7999997 - 1.9999996
1. 0.5787039 0.3644316 0.2441407 0.1714678 0.1250000
exec done

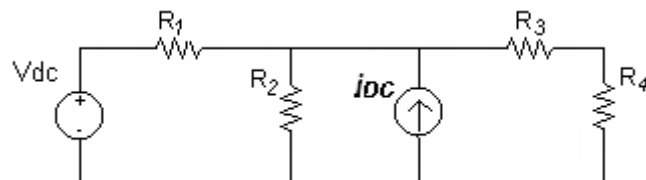
```

Ejercicios:

1. Usando el método de mallas plantee las ecuaciones para el circuito que se muestra a continuación. Resuelva el sistema de ecuaciones mediante el uso de Scilab, cuando: $E1 = 9V$, $E2 = 6V$, $E3 = 8V$, $R1 = 4\Omega$, $R2 = 8\Omega$, $R3 = 7\Omega$, $R4 = 9\Omega$, $R5 = 2\Omega$. Solución: $i1 = 0.24A$, $i2 = 0.21A$, $i3 = -0.43A$



2. Encuentre En el siguiente circuito el voltaje en las resistencias 2 y 4, cuando $V_{dc} = 25V$; $i_{DC} = 10A$, $R1 = 10\Omega$, $R2 = 9V$, $R3 = 7\Omega$, $R4 = 8\Omega$:



BIBLIOGRAFIA

1. Introducción a SCILAB, Héctor Manuel Mora Escobar Departamento de Matemáticas, Universidad Nacional de Colombia Bogota mayo del 2005.
2. Fundamentos Scilab y aplicaciones, de Andrés Alfonso caro, Cesar Valero.
3. Análisis Introductorio de Circuitos, Boylestad, R.L.

Estaré muy agradecida por los comentarios, sugerencias o correcciones enviadas a: dsevilla@cantv.net



REPÚBLICA BOLIVARIANA DE VENEZUELA
MINISTERIO DEL PODER POPULAR PARA LA
EDUCACIÓN SUPERIOR
INSTITUTO UNIVERSITARIO EXPERIMENTAL
DE TECNOLOGÍA DE LA VICTORIA
LA VICTORIA- ESTADO ARAGUA
Comisión Académica del Programa
Nacional de Formación de Electricidad



CÁLCULO NUMÉRICO

PARA

SISTEMAS ELÉCTRICOS

PRÁCTICA I: INTRODUCCIÓN A LA COMPUTACIÓN

Realizado por: Prof. DORIS E. SEVILLA
Revisado por: Prof. JESUS A. PEREZ

La Victoria Octubre de 2007

TEMARIO:

- 1. Introducción.**
- 2. Definiciones.**
- 3. Señales en Scicos.**
- 4. Editor de Scicos.**
- 5. Otras Funcionalidades de Scicos.**
- 6. Crear Subsistemas.**
- 7. Vector Entrada-Salida:**

OBJETIVOS:

- Modelar y simular el software científico Scilab.**
- Modelar sistemas físicos mediante bloques interconectados.**
- Crear sus propios bloques programado con código Scilab.**

1. Introducción:

Scicos (Scilab Connected Object Simulator) es una Toolbox (caja de herramientas) de Scilab para el modelado y simulación de sistemas dinámicos. Permite trabajar con sistemas lineales y no lineales tanto continuos como discretos o híbridos (con partes continuas y discretas). Scicos incluye un editor gráfico para la construcción de modelos mediante interconexión de **bloques** (**blocks**). Estos bloques representan funciones fundamentales predefinidas en Scilab o definidas por el usuario.

Para modelar un sistema, Scicos proporciona una interfase gráfica (*GUI: Graphic User Interface*) que permite construir modelos como diagramas de bloques, utilizando el ratón. Ofrece una biblioteca de elementos (sumideros, fuentes, componentes lineales y no lineales, conectores, etc.) que contiene todos los bloques necesarios para la construcción del modelo.

2. Definiciones:

2.1 Sistema: es un conjunto o colección de componentes interconectados los cuales interaccionan para realizar una función que no es posible obtenerla con cualquiera de los componentes por separado.

El objetivo principal del análisis de un sistema consiste en determinar como esa colección de componentes se comportan cuando están sujetos a una excitación especificada.

2.1.1. Clasificación de Sistemas: Estáticos y Dinámicos.

- Se dice que un sistema es estático si su salida para cualquier instante de tiempo t no depende de valores pasados o futuros de la señal de entrada sino, del valor de la entrada en ese mismo instante de tiempo.
- Un sistema es dinámico, cuando la salida no solamente depende de la entrada en ese instante sino también de al menos uno de los valores pasados de las entradas. En otras palabras diremos que un sistema es dinámico si existe en el una variación de la energía acumulada con el tiempo.

2.2 Modelación de sistemas físicos: para modelar sistemas físicos debemos conocer algunos términos, como:

2.2.1 Función de transferencia: es la relación entre la transformada de laplace de su salida y la transformada de laplace de la entrada, bajo la asunción de que todas las condiciones iniciales sean cero.

2.2.2 Diagrama de bloque: el diagrama de bloque de un sistema es una representación grafica de las funciones realizadas por cada componente del sistema indicando simultáneamente el flujo de las señales.

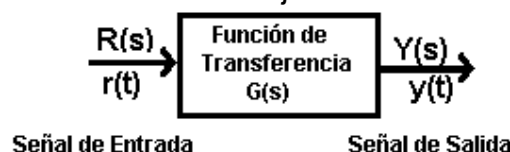


Figura No.1 Elemento del diagrama de Bloque

Cuando tenemos un diagrama de bloque la magnitud de la señal de salida vendrá dada por la multiplicación de la magnitud de la señal de entrada por la función de transferencia del bloque.

$$Y(s)=G(s).R(s)$$

Una vez formado el diagrama de bloques que representa el sistema, se procede a la simulación propiamente dicha, que consiste en la integración numérica de la ecuación (o sistema de ecuaciones) diferencial representativa del sistema.

3. Señales en Scicos.

En scicos cada señal tiene asociada un conjunto de índices de tiempos, llamados tiempo de activación, el conjunto de tiempo de activación es una unión de intervalos de tiempo y puntos aislados llamados eventos (*events*).

Las señales en Scicos son generadas por bloques (*blocks*) controlados por señales de activación, que hacen que los bloques (*blocks*) produzcan una salida a partir de su entrada y de su estado interno. La señal de salida hereda del bloque que la ha generado, el conjunto de tiempos de activación (*activation time set*), pudiéndose usar esta para controlar otros bloques.

3.1 Las entradas para las señales de activación (*activation input ports*) de los bloques, son los que se encuentran en la parte superior y son de color rojo. Un bloque que no tenga entradas de activación se encuentra permanentemente activo, sin embargo recibe sus tiempos de activación a partir de los de sus **señales de entrada** (*input signal*).

3.2 Las salidas de activación (*activation output ports*) de un bloque se encuentran en la parte inferior, las señales de salida de estos puertos se consideran por tanto señales de activación generados por los bloques.

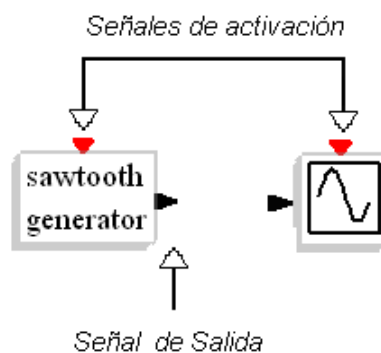


Figura No.2: Bloques (*blocks*)

4. Editor de Scicos:

Se cuenta con dos formas para abrir la herramienta Scicos:

1. Se despliega Applications en la barra de menú y se hace un clic en Scicos:

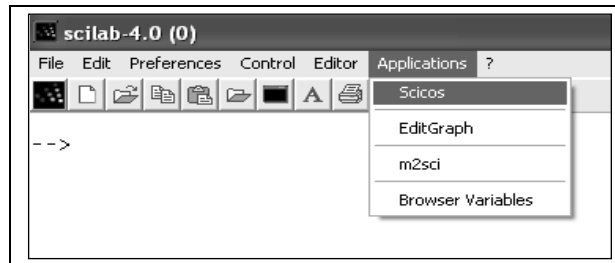


Figura No.3 Ventana de comandos de Scilab, usando la barra de herramientas

2 Se escribe scicos en la ventana de comando:

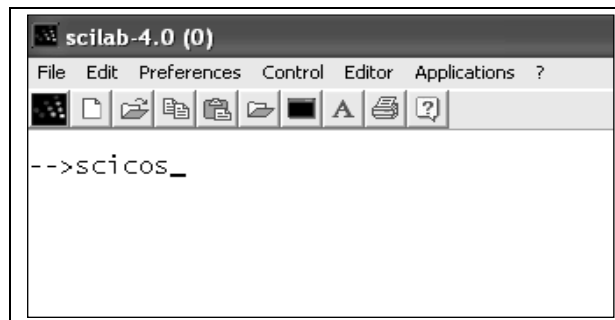


Figura No.4 Ventana de comandos de Scilab

Se abre la ventana de trabajo de Scicos:

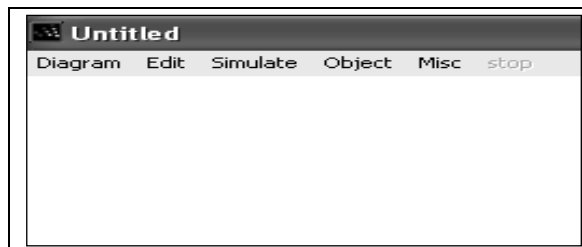


Figura No.5 Ventana del diagrama de bloque de Scicos

En la parte superior de la ventana del diagrama de bloque de Scicos aparecen una serie de botones: Diagram, Edit, Simulate, Object, Misc y Stop.

Para acceder a los iconos que permiten construir los diagramas de bloques hemos de seleccionar **Edit>Palette**. En:

- **Linear:** Contiene los bloques relacionados con los sistemas lineales (derivada, integral, función de transferencia, tiempo muerto, tiempo muerto variable, etc.).
- **Non_linear:** Contiene aquellos bloques, utilizados habitualmente en control, y que tienen un comportamiento no lineal o con discontinuidades (backslash -simulación del "juego" de válvulas-, zona muerta, relé -controlador todo/nada-, saturación -limitador-). Incluye también funciones matemáticas.
- **Others:** Contiene otros bloques tales como el backslash -simulación del "juego" de válvulas-, o el Super Block, que permite definir bloques constituidos por diagramas de bloques (esto es especialmente útil para definir controladores PID, ya que Scicos carece de un bloque controlador PID).

- **Branching:** Recoge los bloques generales que se utilizan en la construcción de diagramas (subsistemas, entrada, activación, disparo, multiplexor, salida, etc.).
- **Sinks:** Bloques receptores de datos (display, enviar a archivos, gráfico xy, gráfica en función del tiempo, etc.).
- **Sources:** Bloques generadores de datos (generador de señales, generador de señal sinusoidal, reloj, constante, función escalón, función rampa, etc.).
- **Electrical:** fuentes, componentes discretos, componentes eléctricos.
- **ThermoHydraulics:** pipa, tap, pozos, tanques.
- **OldBlocks:** contienen bloques de la antigua versión de Scicos.
- **DemoBlock:** scope

4.1 Construcción de un modelo sencillo: la construcción de un modelo depende de tres partes: copia, conexión y simulación.

Se construirá el siguiente modelo: $I = \int 5 \sin t \, dt$

a) La copia de bloques: se realiza en la ventana principal de Scicos desde las paletas, haciendo click en el botón, con el ratón los arrastramos y los soltamos sobre la hoja del nuevo modelo.

Edit:

Palettes:

Sources: se arrastran los bloques: Clock, sinusoid generator.

Linear: se arrastran los bloques: Gain, integral block.

Sinks: se arrastra el bloque: Scope (Osciloscopio)

Others: se arrastra el bloque: text

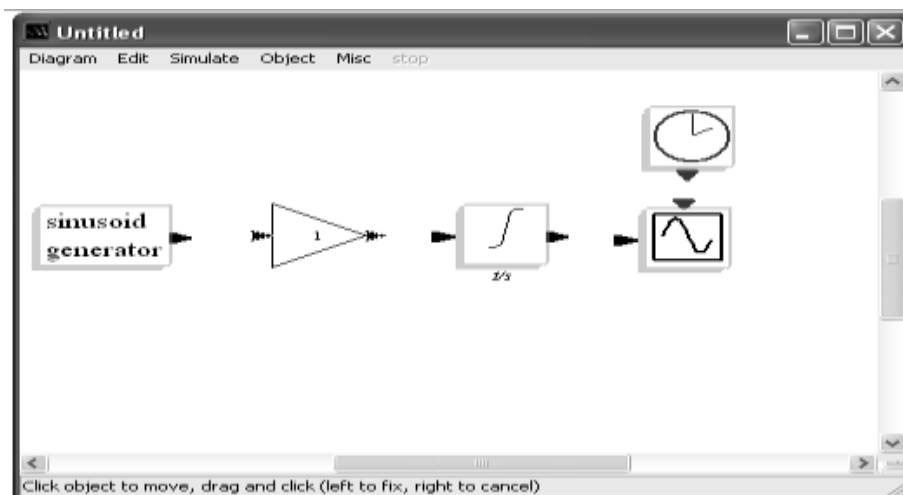


Figura No.6 Diagrama de Bloque para construir el modelo

Con el botón derecho del ratón cancelo la selección de bloque realizada, además si quiero borrar un bloque ya pegado en la ventana principal, hago click en el botón **Delete** del menú **Edit** y hago click sobre el bloque a borrar. Con el botón derecho del ratón sobre el bloque se puede modificar sus parámetros y sus propiedades.

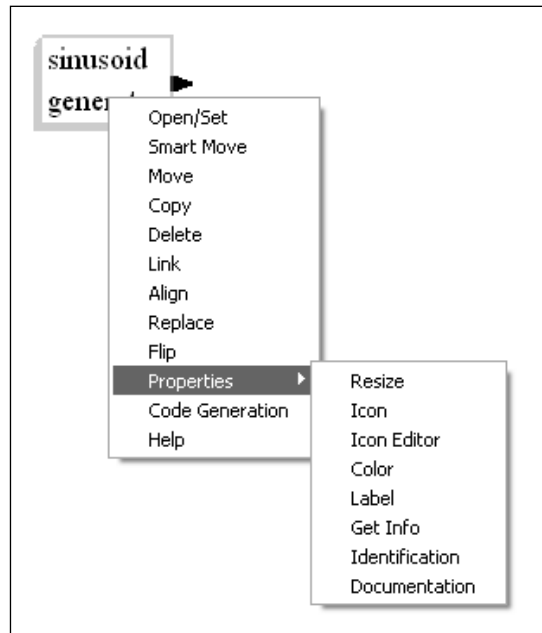


Figura No.7 Parámetros y propiedades del generador de señales senoidales

b. Conexión: con el ratón podemos conectar los puertos de entrada con los de salida. Al conectarlo, aparece una flecha dirigida desde el puerto de salida al de entrada. Debo conectar las entradas y salidas de los bloques, seleccionando primero el botón **Link** del menú **Edit**.

Una vez he hecho click en el botón **Link**, hago click sobre la salida de un bloque y luego sobre la entrada del siguiente, quedando ambos conectados. También puedo partir o terminar en puntos intermedios en los enlaces, o sea hago click sobre el botón **Link** y pincho en un punto de un enlace a partir del cual puedo obtener las bifurcaciones deseadas.

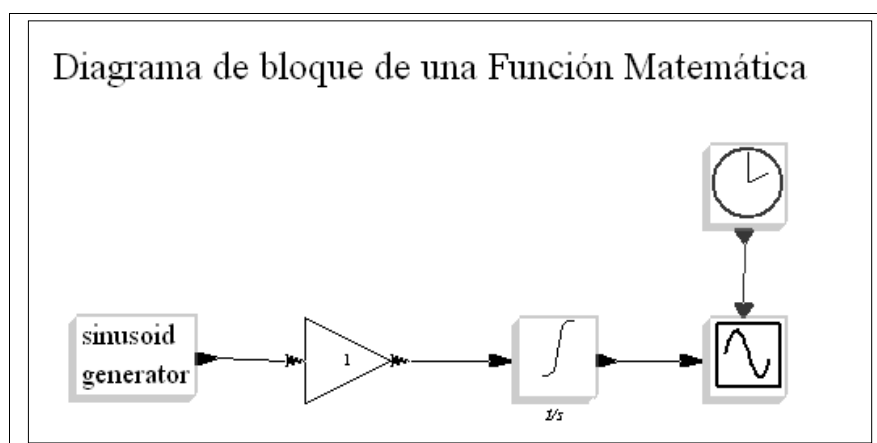


Figura No.8 Bloques conectados.

c) Simulación: Si nos situamos con el ratón sobre el nombre del bloque, podemos cambiarlo y escribir un nombre que sea más representativo del bloque. Como nombre de bloque se puede utilizar cualquiera que deseemos, con la única limitación de que no puede haber nombres repetidos. Haciendo doble click en los bloques accedemos a un cuadro de diálogo que nos muestra detalles del módulo.

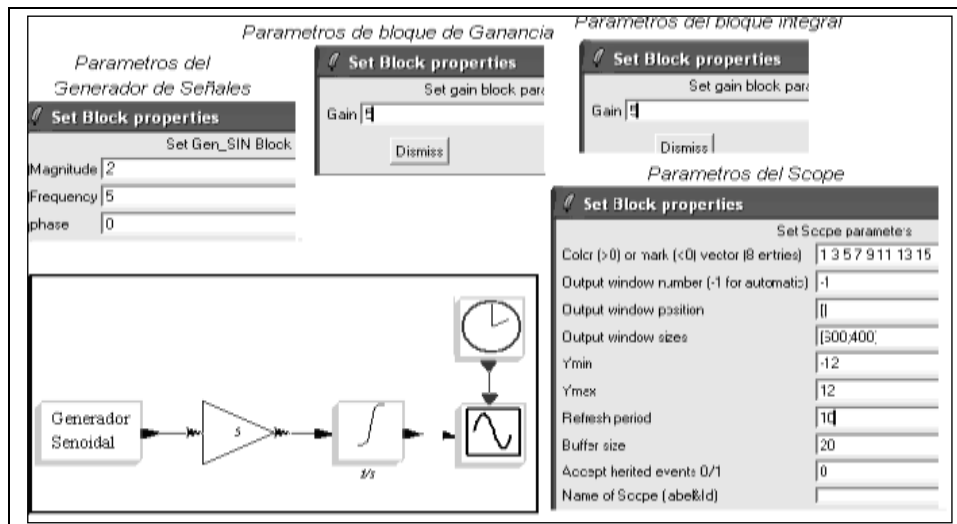


Figura No.9 Parámetros de los bloques.

Para guardar el modelo creado hay que acceder a la barra de botones: *Diagram>Save* o *Diagram>*

Desde la barra de botones, pulsando *Simulate>Setup* accedemos al cuadro de diálogo de los parámetros de simulación

Con este diálogo se puede definir:

- Tiempo final de la simulación (el inicial siempre es cero) [Final integration time].
- Tolerancias en el cálculo numérico [*Integrator absolute tolerance* e *Integrator relative tolerance*].
- Elegir el tipo de "solver" (algoritmo de integración numérica) utilizado para resolver la ecuación diferencial ordinaria representativa del diagrama de bloques.

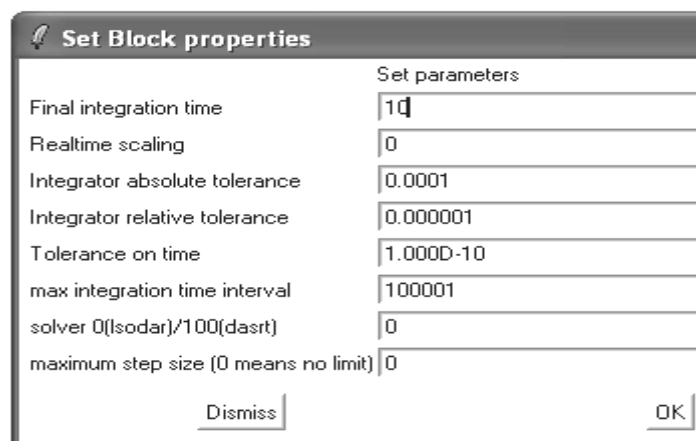


Figura No.10 Parámetros de la simulación.

Una vez seleccionado el botón **Run** se realiza la compilación del modelo (si no ha sido ya compilado) y después la simulación. La *simulación* puede ser parada en cualquier momento simplemente haciendo clic sobre el botón **Stop** en la ventana principal de Scicos.

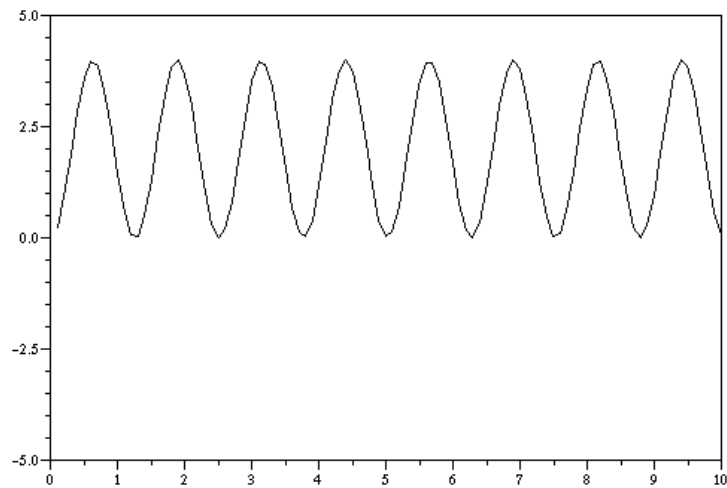


Figura No.11 Simulación

5. Otras Funcionalidades de Scicos.

El editor Scicos proporciona muchas otras funcionalidades tales como:

- Salvar y cargar diagramas de modelos en varios formatos.
- Ampliación de zonas del modelo.
- Cambio del aspecto de los bloques y colores.
- Cambio del color de fondo de los diagramas de modelos y de los bloques.
- Ubicar texto en el diagrama.
- Impresión y exportación de diagramas Scicos.

El botón de ayuda de la ventana principal de Scicos puede usarse de dos maneras: seleccionando el botón ayuda y haciendo click sobre un bloque obtengo un manual de ayuda del bloque o seleccionando el botón ayuda y haciendo click sobre otro botón, mostrándose el manual de ayuda de ese botón.

También puedo obtener información de las funciones que puedo implementar en la programación de nuevos bloques desde la ventana principal de Scilab haciendo click en el botón Help Dialog del menú Help apareciendo, la siguiente ventana en la que elijo Scicos en zona inferior Chapters y seleccionando en la zona superior la función de la que quiera ayuda y pulsando el botón Show.



Figura No.12 Ventana de ayuda de Scicos-Scilab

Si quiero ejecutar las demos de Scicos selecciono el botón Demos del menú File en la ventana principal de Scilab, apareciendo la siguiente ventana de diálogo: selecciono Scicos y hago click en el botón OK, apareciendo otra ventana con todas las demos disponibles para Scicos, seleccionando la que quiera ejecutar y haciendo click en el botón OK

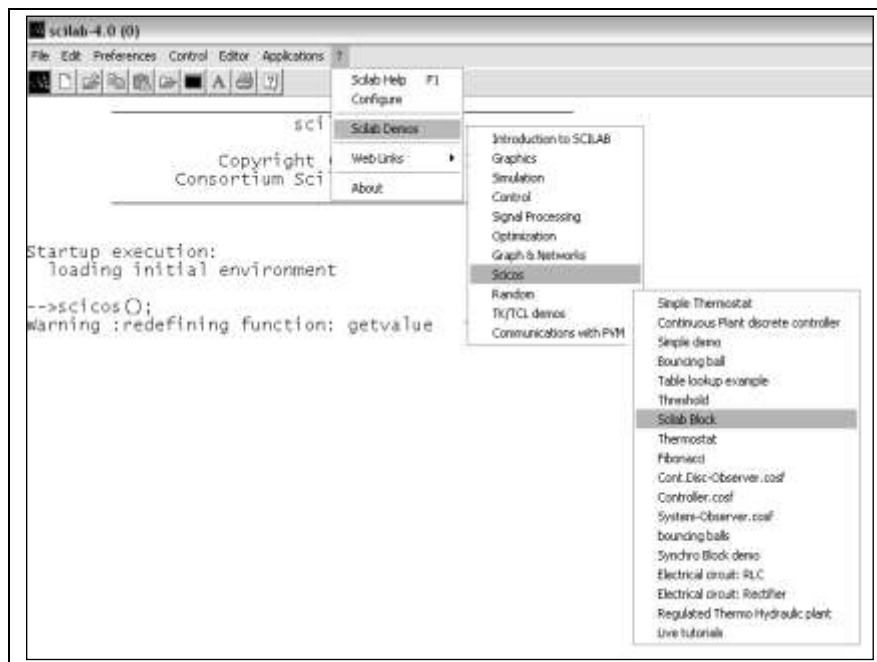


Figura No.13 Demos de Scicos-Scilab

6. Crear Subsistemas: No es deseable tener modelos complejos en Scicos con cientos de bloques en un solo diagrama, por lo que Scicos ofrece la posibilidad de hacer agrupaciones de bloques. Para esto Scicos proporciona la posibilidad de agrupamiento de bloques definiendo subdiagramas llamados **Super Bloques** (*Super Blocks*), que se comportan como cualquier otro bloque pero pueden contener un número ilimitado de bloques, e incluso otros **super bloques**. Los **super bloques** tienen la siguiente apariencia:



Figura no.14 Super Bloque (Subsistema)

Ejemplo: Subsistema

1. Para ilustrar la creación de un subsistema, se dibuja el siguiente modelo:

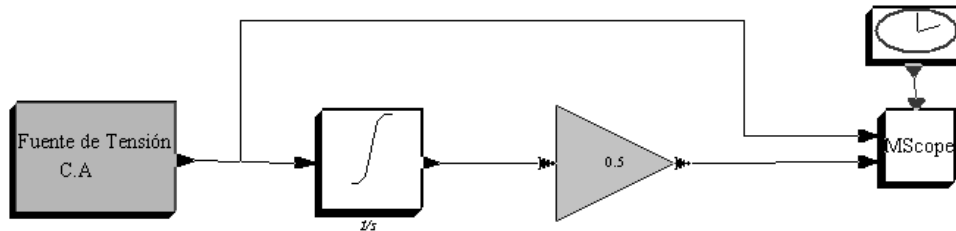


Figura No. 15 Diagrama de Bloque

2. Se realiza la selección de la región que será parte de subsistema de la siguiente forma: →Diagram→Region to Super Block: dando un click al botón izquierdo del ratón se encierra en un cuadro los bloques que pertenecerán al subsistema:

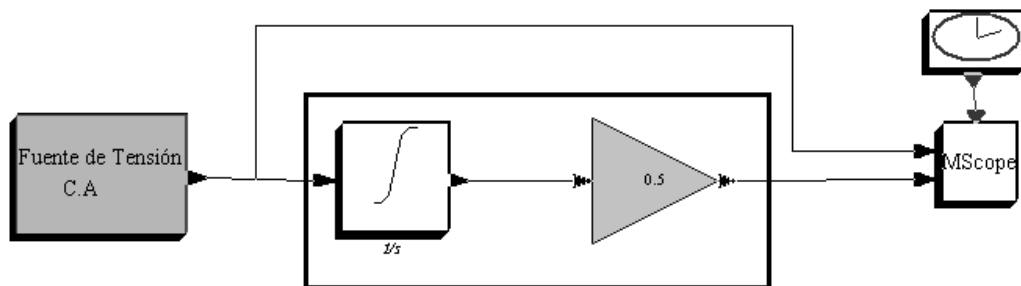


Figura No. 16 Selección de la región.

3. Con un click izquierdo en el cuadro creado aparece el Súper bloque:

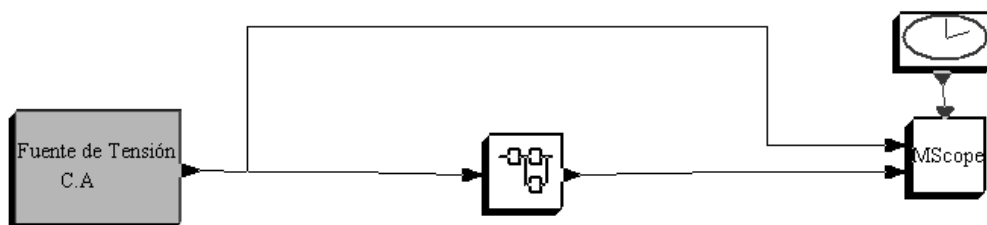


Figura No. 17 Súper Bloque.

4. Para inspeccionar el interior del super block, se hace doble click en el Super Block, se abrirá entonces una nueva ventana grafica de Scicos mostrando los bloques del sistema:

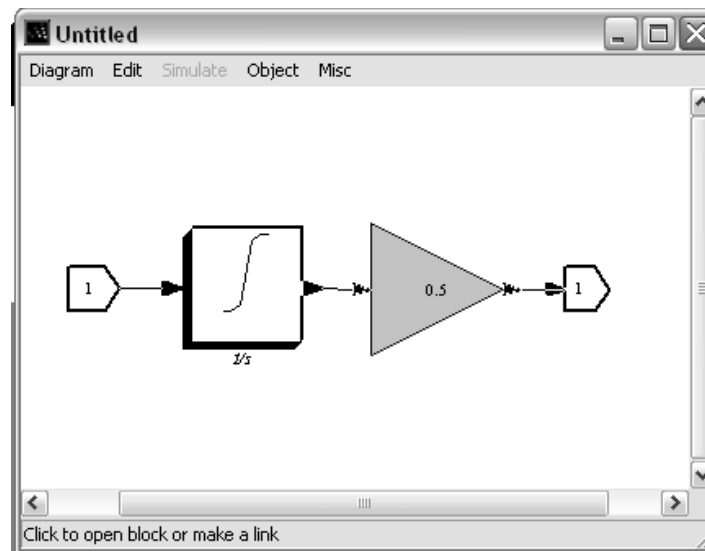


Figura No. 18 Interior del Super Bloque.

5. Simulación:

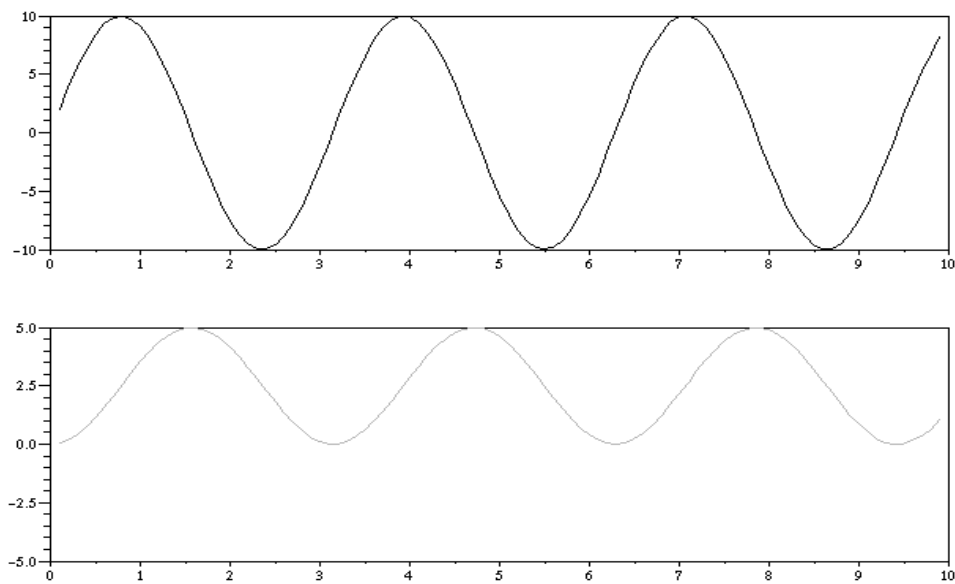


Figura No. 19 Simulación.

6.1 Otra forma de crear subsistema:

Edit → palettes → Others: selecciona el Super bloque:

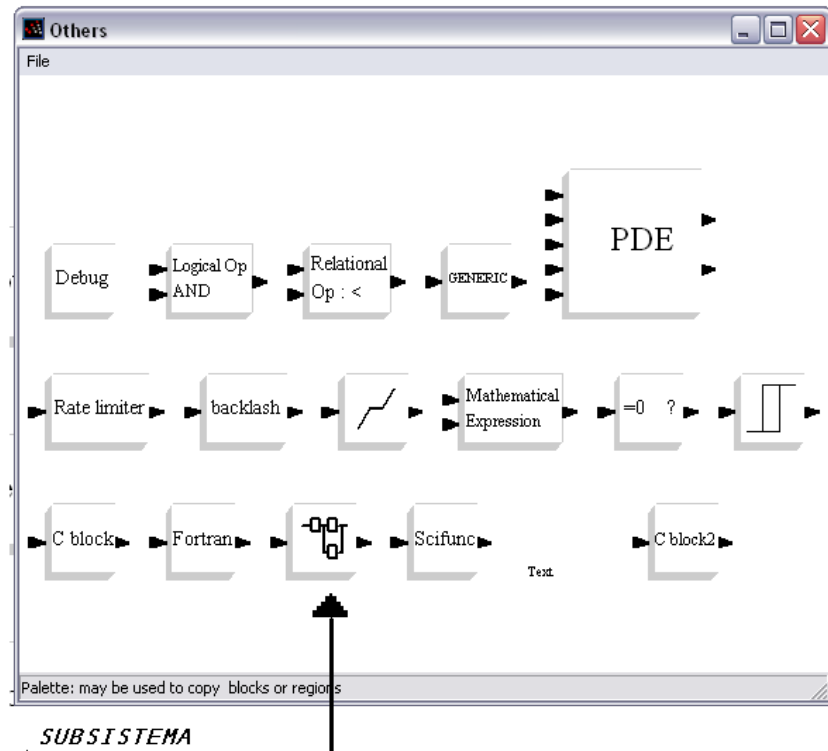


Figura No. 20 Súper Bloque o Subsistema.

7. Vector Entrada-Salida:

Scicos contiene bloques que pueden tener mas de una entrada y más de una salida, estas entradas y salidas pueden ser vectorizadas. Así cada puerto se le asocia un tamaño. Cuando se esta conectando un puerto de entrada a un puerto de salida, el tamaño de los puertos debería ser igual.

7.1 Vector Entrada:

El MUX (multiplexer) esta en palette → Branching. Este Bloque puede tener cualquier número de entrada; la salida es un vector obtenido como una concatenación de las entradas. No es necesario especificar el tamaño de cada entrada. En el siguiente ejemplo el numero d entradas son dos y el de salida es uno. Similar para el Scope (aquí no es necesario especificar el tamaño del vector de entrada, este se ajusta automáticamente).

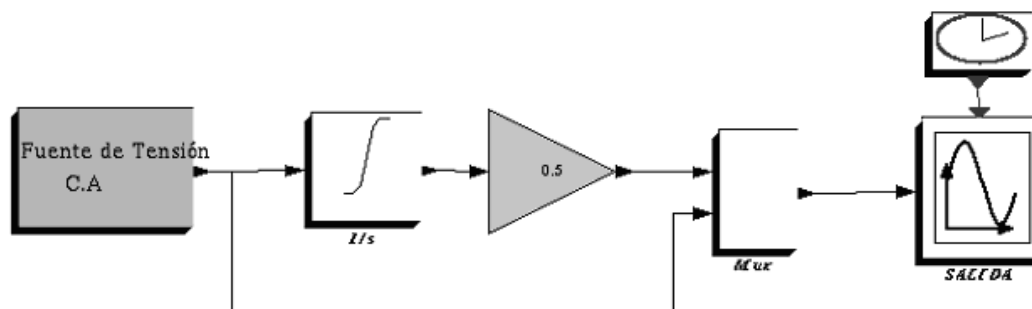


Figura No. 21 Diagrama de bloque, usando un MUX.

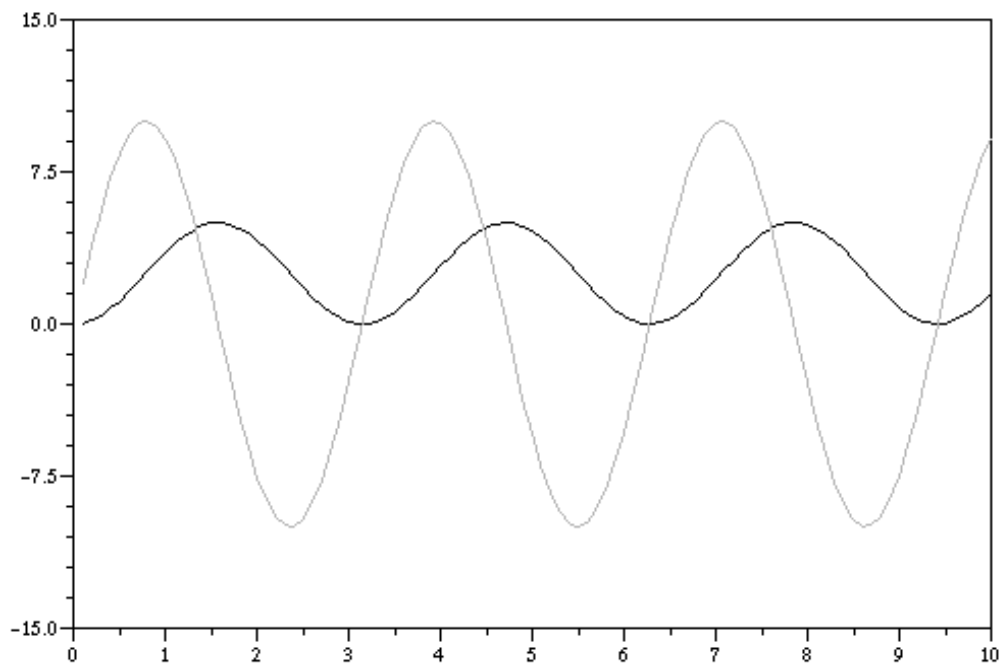


Figura No.22 Señales Superpuestas (Scope)

Cuando el Scope recibe un vector de entrada, superpone las señales.

7.2 Vector de salida:

El Bloque Demux no es el opuesto al Mux, este corta el vector de la entrada en vectores más pequeños del tamaño.

Para evitar que la salida de Scope y MScope dibuje en las graficas, es necesario introducir un conjunto de parámetros de Mscope tales que se use en una ventana diferentes gráficos.

Ejercicios:

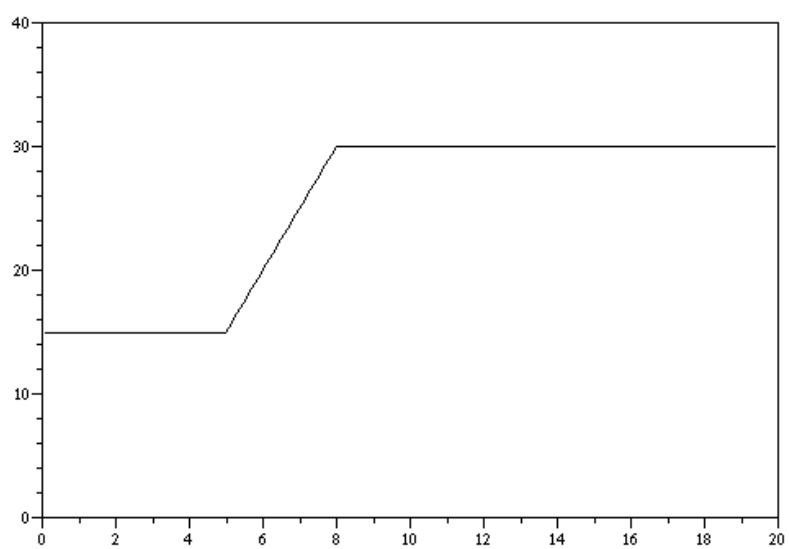
1. Grafique las siguientes funciones:

$$a) m(t) = \sin(\omega t + u(t) - 2)$$

$$b) h(t) = r_1(t) + r_2(t) - 5$$

$$c) z(t) = 3r(t) \sin 5$$

2. Encuentre la función y el diagrama de bloque para generar las siguientes graficas:



Grafica No.1

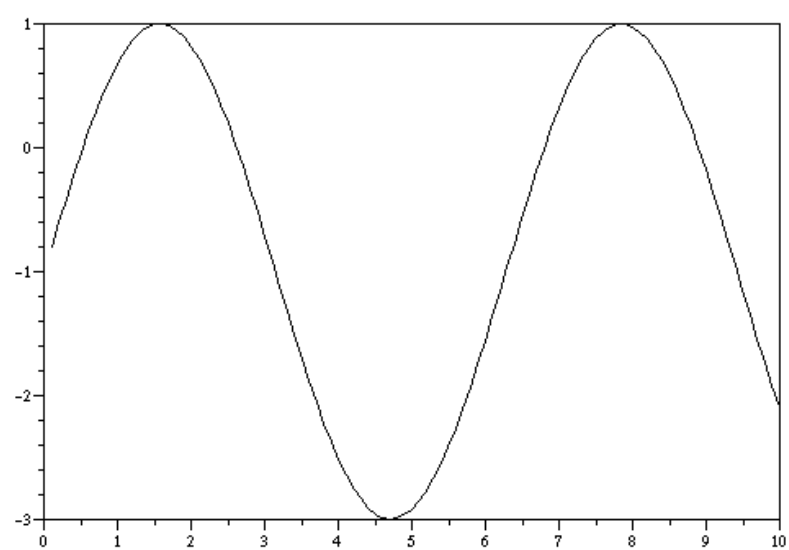
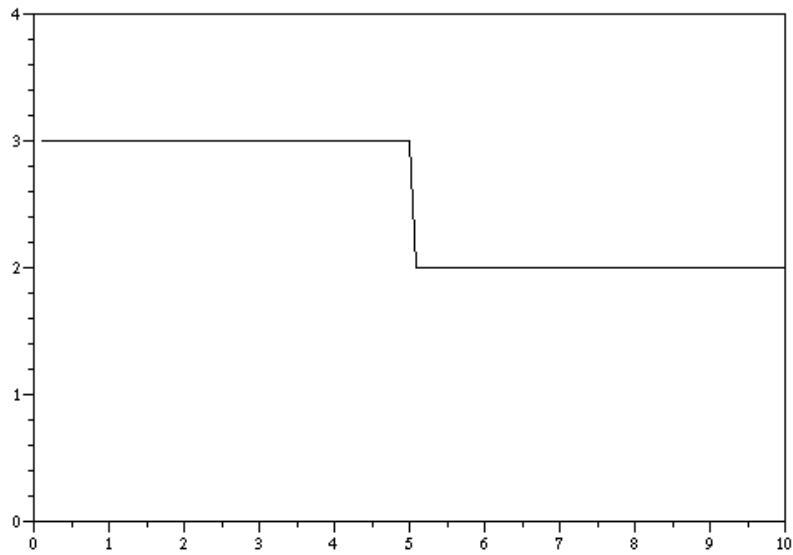


Grafico No.2



Grafica No.3

Bibliografía:

1. Introducción a los Sistemas Lineales, Giovanni Ghelfi, Universidad de Carabobo.
2. Análisis Introductorio de Circuitos, Boylestad, R.L.
3. Paginas en Internet:
www.scilab.org

Estaré muy agradecida por los comentarios, sugerencias o correcciones enviadas a: dsevilla@cantv.net



REPÚBLICA BOLIVARIANA DE VENEZUELA
MINISTERIO DEL PODER POPULAR PARA LA
EDUCACIÓN SUPERIOR
INSTITUTO UNIVERSITARIO EXPERIMENTAL
DE TECNOLOGIA DE LA VICTORIA
LA VICTORIA- ESTADO ARAGUA
Comisión Académica del Programa
Nacional de Formación de Electricidad



CÁLCULO NUMÉRICO

PARA

SISTEMAS ELÉCTRICOS

PRÁCTICA I: INTRODUCCIÓN A LA COMPUTACIÓN

Realizado por: Prof. DORIS E. SEVILLA
Revisado por: Prof. JESUS A. PEREZ

La Victoria Octubre de 2007

TEMARIO:

- **Modelar y simular sistemas eléctricos bajo la herramienta Scicos de Scilab**

OBJETIVOS:

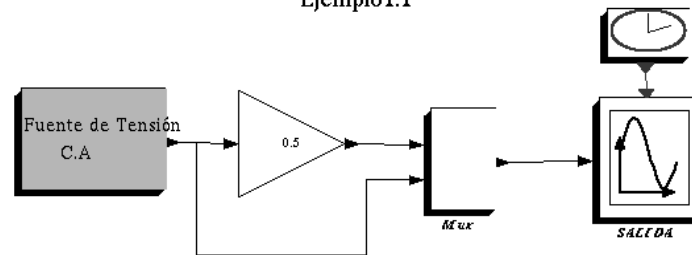
- **Diseñar, Modelar y simular sistemas eléctricos y mecánicos.**
- **Reconocer la fuente de información que ofrece la ayuda de Scilab dentro de la librería de Scicos.**
- **Visualizar los Demos de Scicos en Scilab.**

Scicos de Scilab es un paquete para modelado y simulación de sistemas dinámicos que incluye subsistemas tanto continuos como discretos.

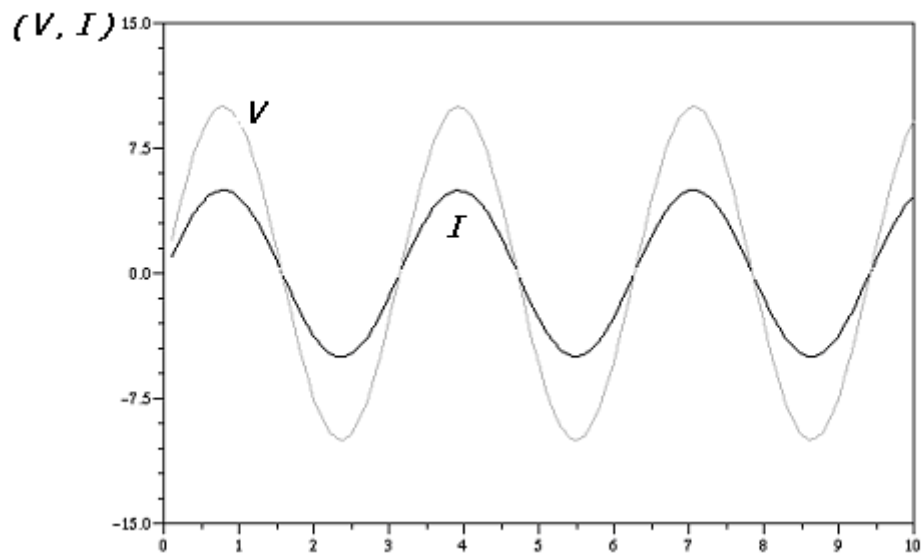
Los sistemas eléctricos pueden ser simulados en Scicos para observar su comportamiento y realizar las variaciones necesarias al sistema según sean los requerimientos del diseñador.

Ejemplo No.1:

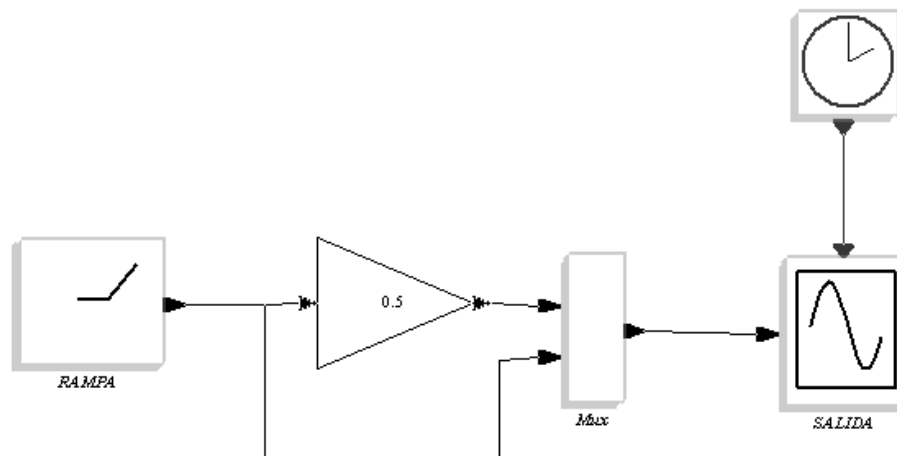
Ejemplo1.1



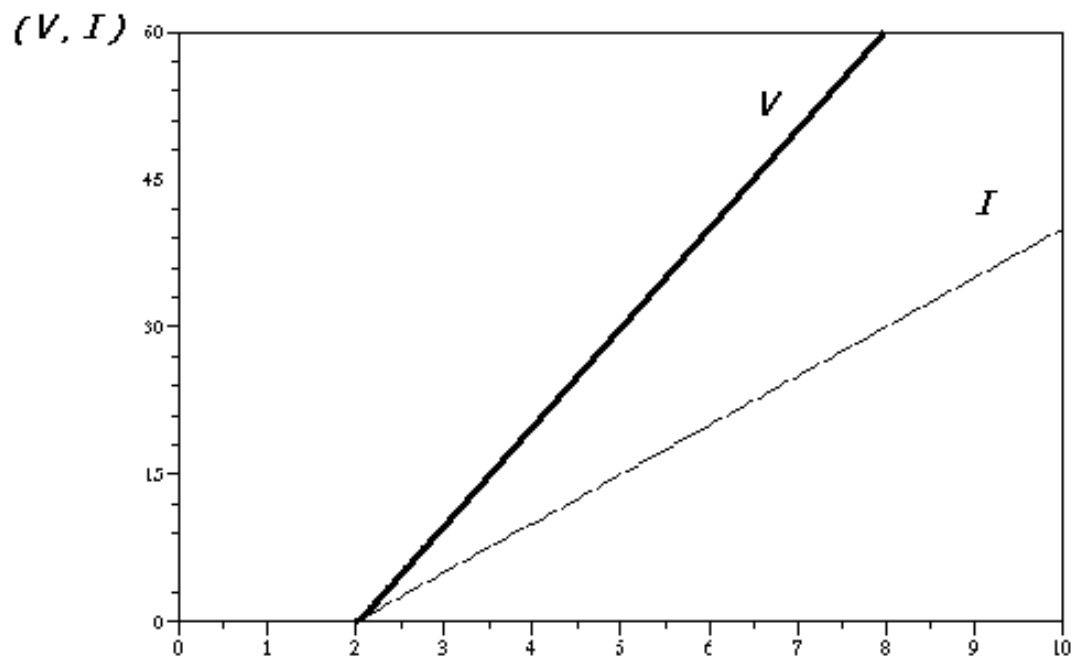
Simulación:



Ejemplo 1.2:

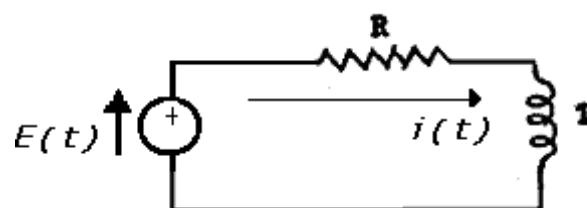


Simulación:



2. Sea un circuito simple que contiene una resistencia $R=50\Omega$ y una bobina de inducción $L=0.05H$, en serie con una fuerza electromotriz $E(t)$. Realice la simulación del circuito en Scicos para conocer el comportamiento de la corriente cuando la fuente de tensión es: (resuelva el circuito aplicando Ecuaciones diferenciales)

1. Una rampa. 2. Un escalón



Solución:

- La caída de potencial debido a la resistencia R, es proporcional a la intensidad i de la corriente que fluye por el circuito, $E_R = Ri$
- La caída de potencial debido a la bobina de inducción, es proporcional a la rapidez de cambio de la intensidad i de la corriente, o sea, $E_L = L \cdot \frac{di}{dt}$

Las leyes fundamentales para circuitos eléctricos y redes, se conocen como leyes de kirchoff:

1. La suma algebraica de las caídas de potencial alrededor de un circuito cerrado es cero.
2. La suma algebraica de todas las corrientes que fluyen a un nodo es cero.

Para el caso de un circuito simple solo se aplica la ley 1, entonces se tiene la siguiente relación:

$$E(t) - Ri - L \frac{di}{dt} = 0$$

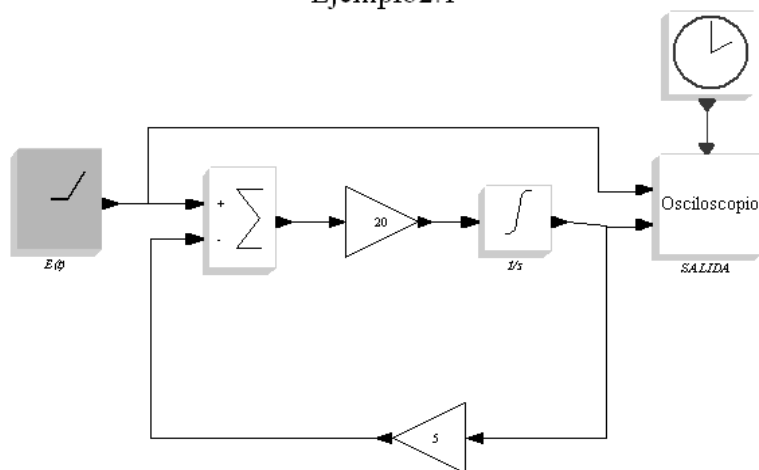
$$\frac{di}{dt} = \left[E(t) - iR \right] \cdot \frac{1}{L}$$

$$i(t) = \int \frac{1}{L} \cdot [E(t) - iR] dt$$

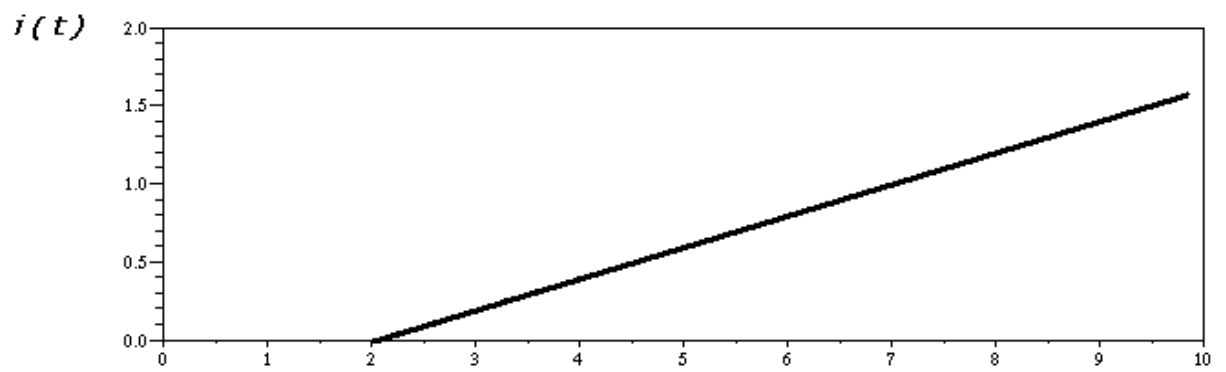
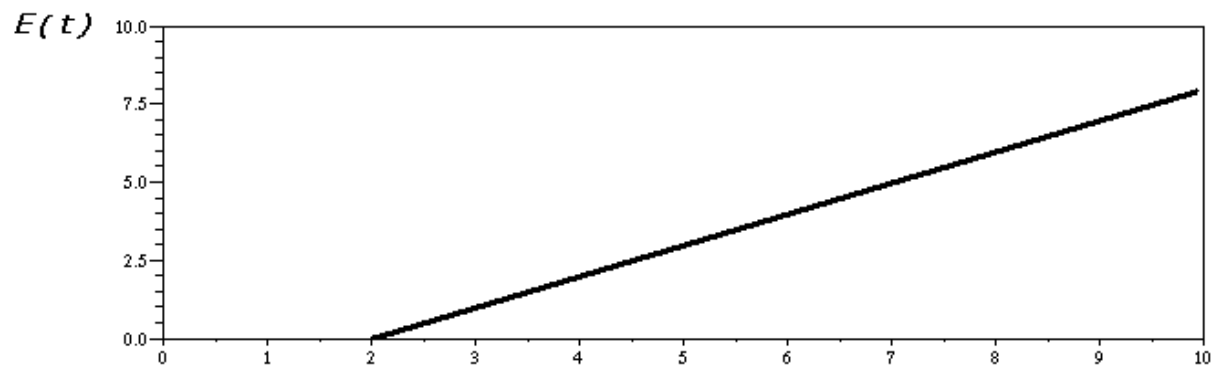
2.1. Entrada rampa, $R=5\Omega$; $L=0.05H$

Scicos:

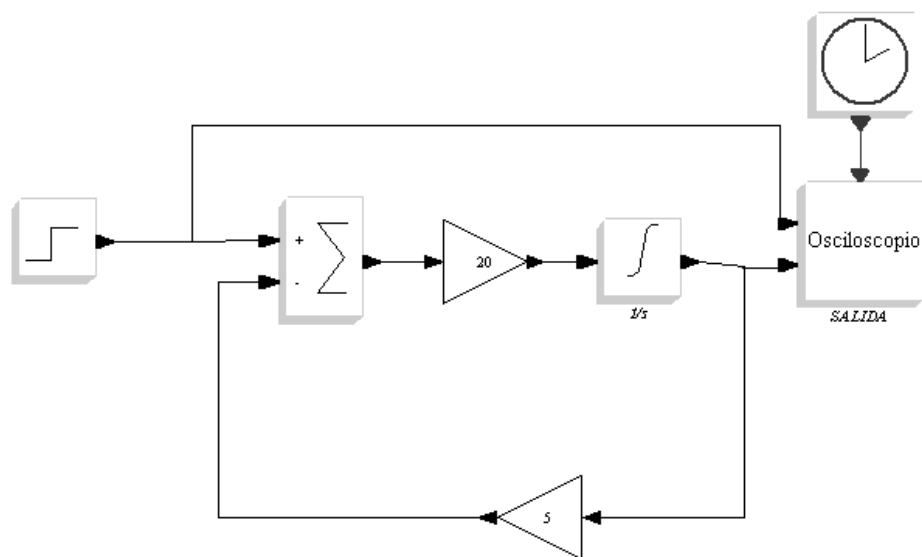
Ejemplo2.1



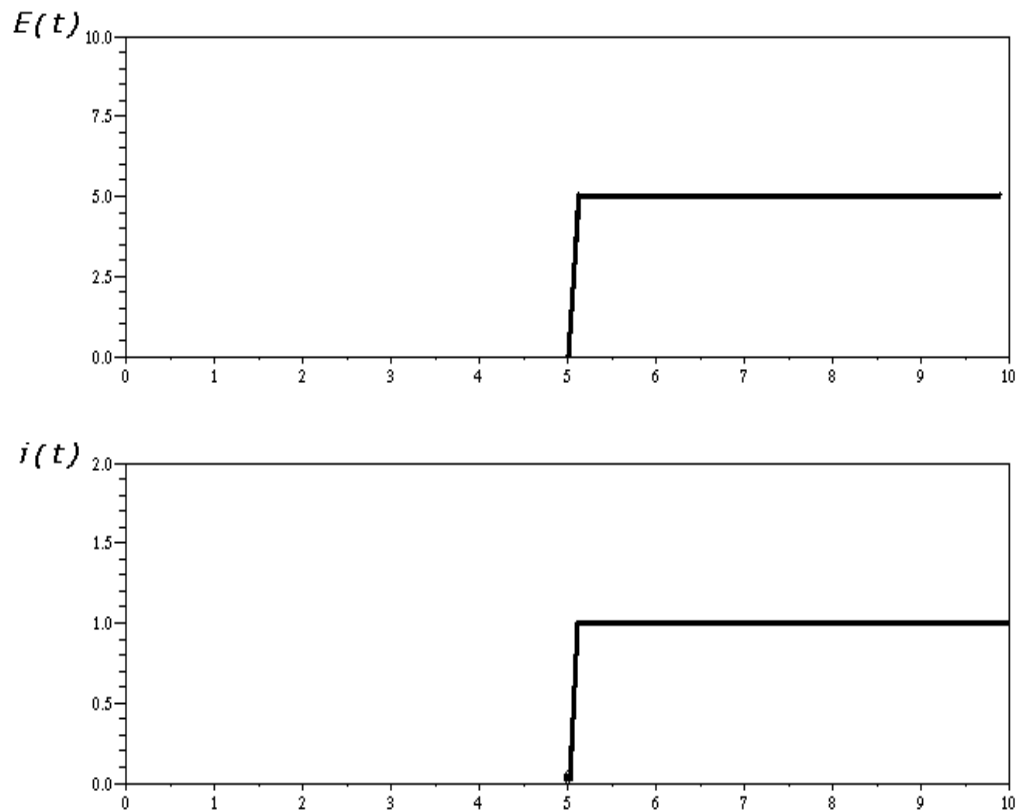
Simulación:



2.2. Entrada Escalón: $R=5\Omega$; $L=0.05H$



Simulación:



Ejemplo 3: Modela los siguientes circuitos R, RL, RC y observa a través del Scope el comportamiento de la corriente que circula por el circuito (aplicando transformada de laplace). Si el voltaje aplicado es una fuente escalón unitario:

$0 \dots t < 1$

$1 \dots t \geq 1$

$$V(t) = \begin{cases} 0 & t < 1 \\ 1 & t \geq 1 \end{cases}$$

Sabiendo que una función de transferencia en un sistema es:

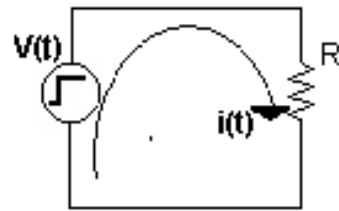
$$G(s) = \frac{Y(s)}{R(s)}$$

Donde:

$R(s)$: Variable de entrada.

$Y(s)$: variable de Salida

CIRCUITO RESISTIVO PURO:



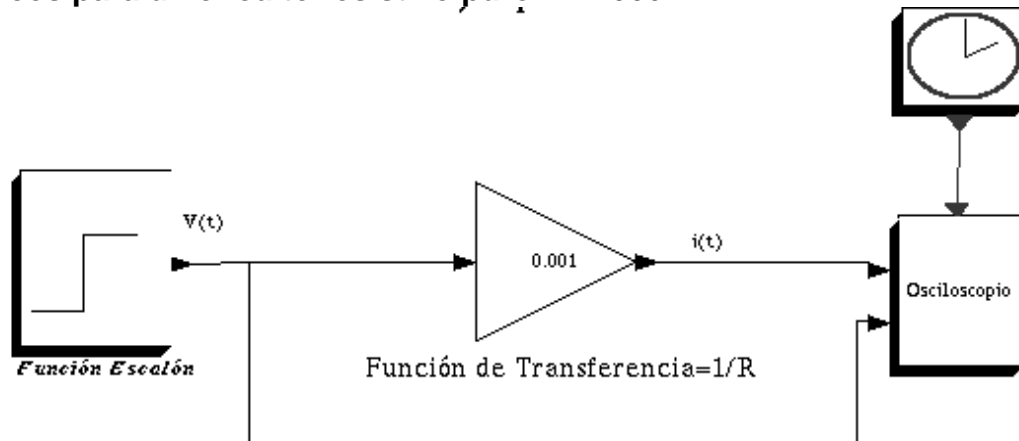
$$V(t) = V_r(t) = i(t) \times R$$

$$V_r(s) = I(s) \times R$$

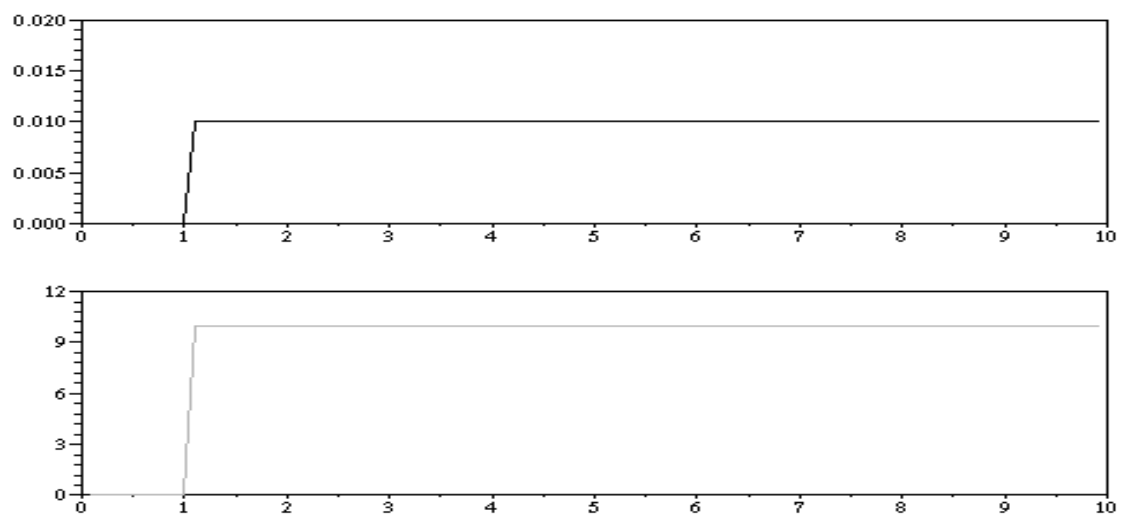
$$G(s) = \frac{I(s)}{V(s)} = \frac{1}{R}$$

$$G(s) = \frac{1}{R}$$

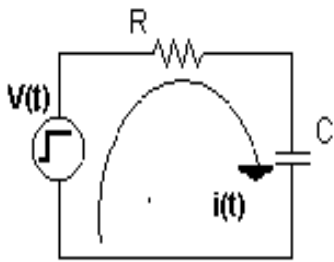
Scicos para un circuito resistivo puro: $R=1000\Omega$



Simulación:



CIRCUITO RC EN SERIE:



$$V(t) = V_R(t) + V_C(t) = i(t) \cdot R + \frac{1}{C} \int i(t) dt$$

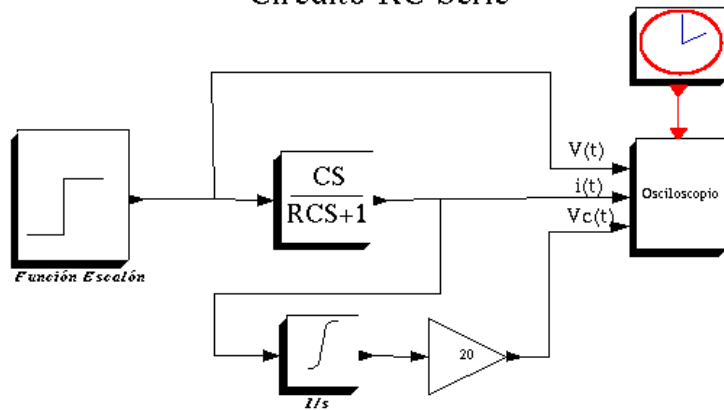
$$V(s) = I(s) \cdot R + \frac{1}{C} \cdot \frac{1}{s} I(s) = I(s) \left[R + \frac{1}{Cs} \right]$$

$$G(s) = \frac{I(s)}{V(s)} = \frac{Cs}{RCs + 1}$$

$$G(s) = \frac{Cs}{RCs + 1}$$

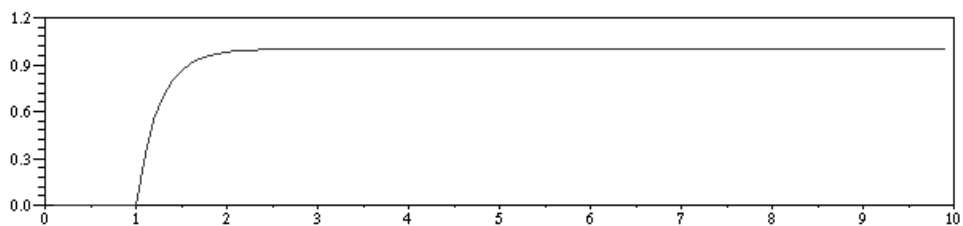
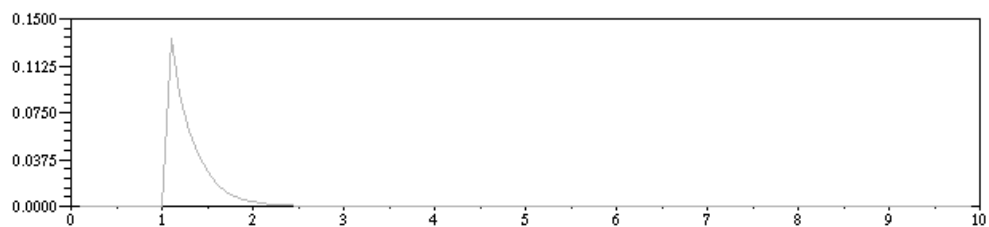
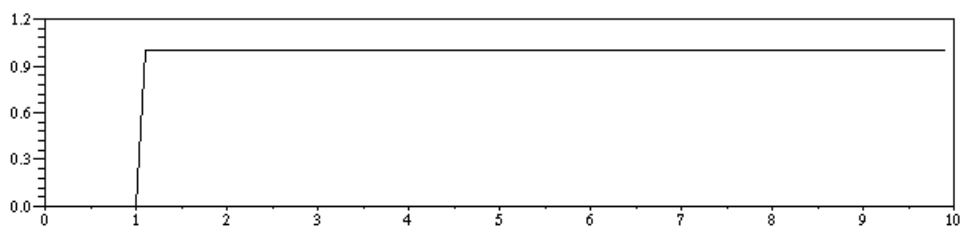
Scicos del circuito RC en serie: R=5л; C=0.05f

Circuito RC Serie

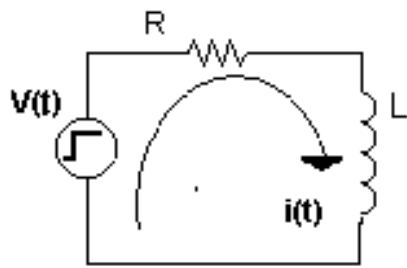


Simulación:

Graficas: 1: V(t),2: i(t),3: Vc(t)



CIRCUITO RL EN SERIE:



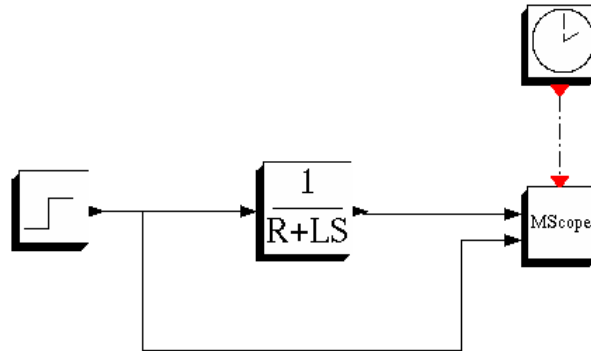
$$V(t) = V_R(t) + V_L(t) = i(t)R + L \frac{di(t)}{dt}$$

$$V(s) = I(s)R + LsI(s) = I(s)[R + Ls]$$

$$G(s) = \frac{I(s)}{V(s)} = \frac{1}{R + Ls}$$

$$G(s) = \frac{1}{R + Ls}$$

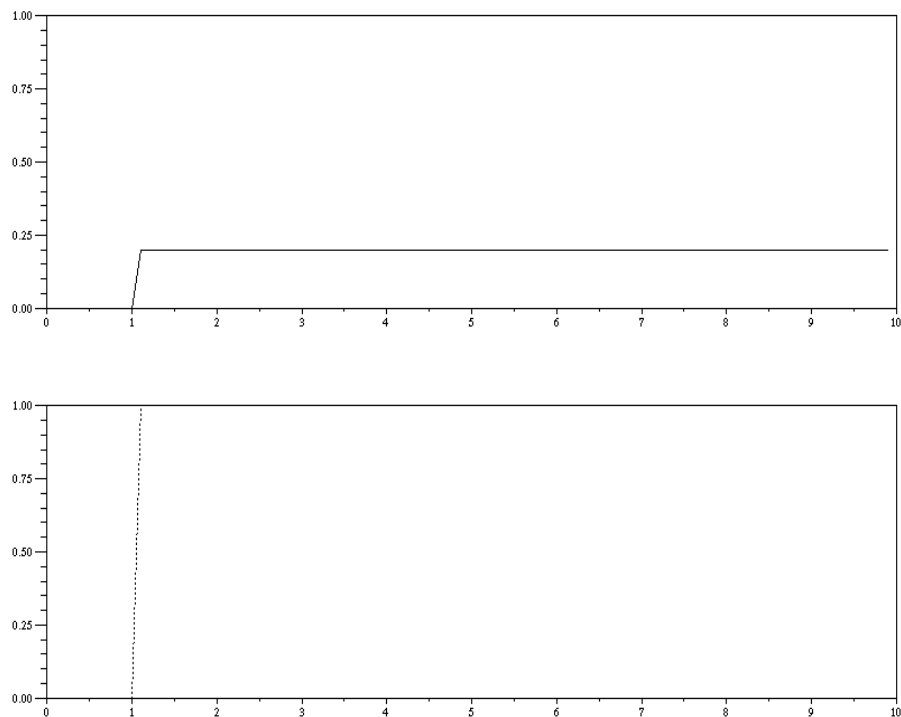
Valores del circuito RL en serie: $R=5\Omega$; $L=0.05H$



Simulación:

Graficas:

1: $i(t)$, 2: $V_L(t)$



Ejemplo 4:

Dado un circuito con R , L , C en serie con una fuente de tensión escalón unitario, observar el comportamiento de la carga almacenada por el condensador.

ve el comportamiento de la carga almacenada por el condensador.



$$V(t) + V_R(t) + V_L(t) + V_C(t) = 0$$

$$V(t) = L \frac{dq^2(t)}{dt^2} + R \frac{dq(t)}{dt} + \frac{1}{C} q(t)$$

Aplicando la Transformada de Laplace, nos queda:

$$V(s) = \left[LS^2 + RS + \frac{1}{C} \right] Q(s)$$

$$\frac{Q(s)}{V(s)} = \frac{C}{LS^2 + RCS + 1}$$

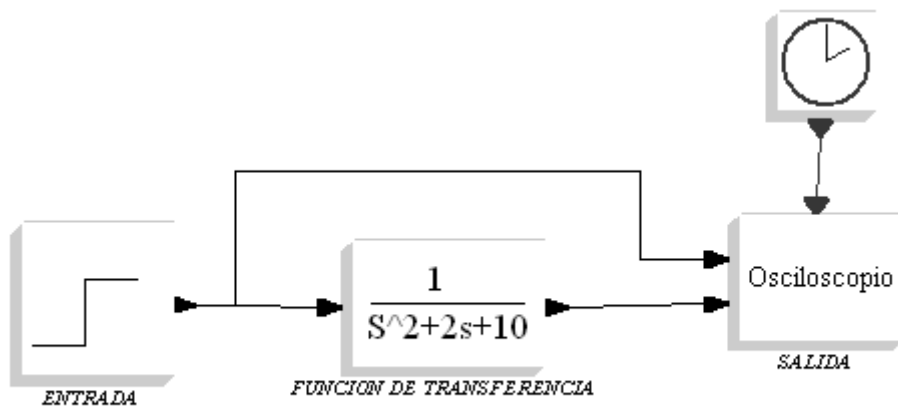
$$F(s) = \frac{Q(s)}{V(s)}$$

Siendo F(s): la función de transferencia del sistema.

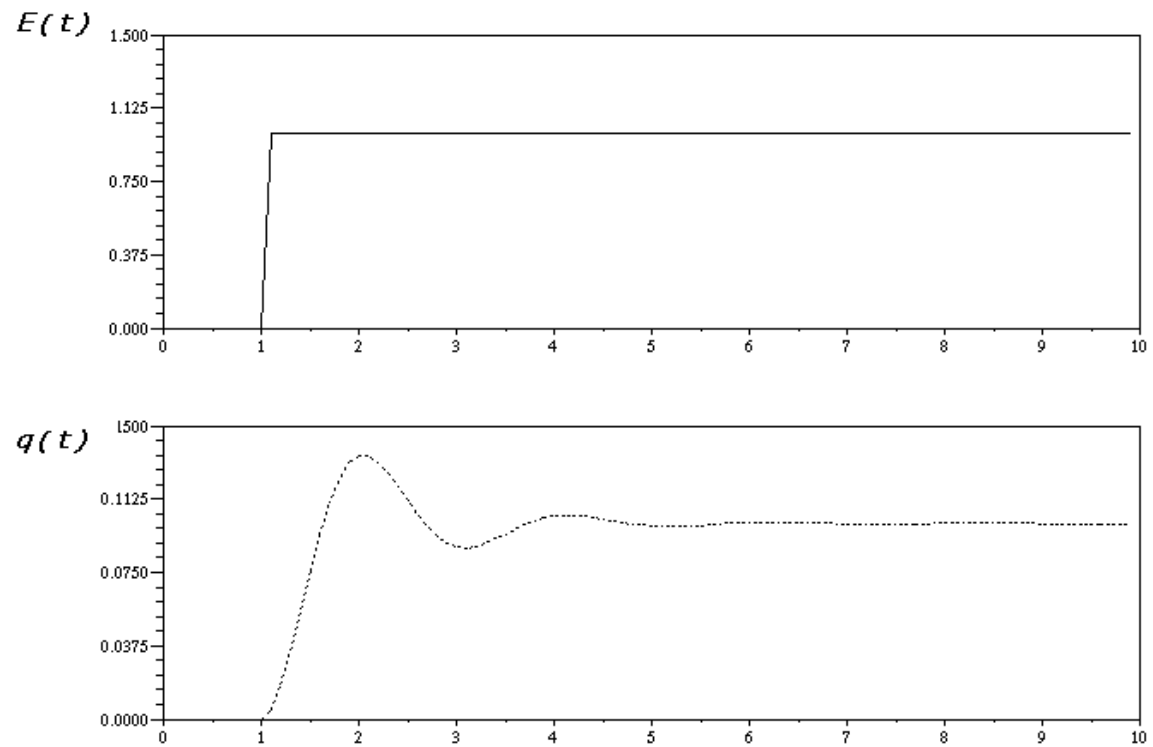
Siendo: R = 1Ω, L = 1H, C = 1F

$$s_{1,2} = -\frac{1}{2} \pm j \frac{\sqrt{3}}{2}$$

$$F(s) = \frac{1}{s^2 + s + 1}$$



Simulación:



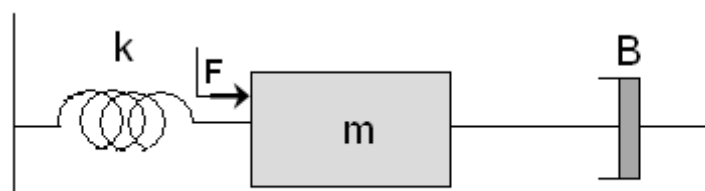
Ejemplo 5: Dado el siguiente sistema mecánico, con $m=1\text{kg}$; $k=3$; $B=15$

$0 \dots t < 1$

$10 \dots t \geq 1$

Con :

$$F(t) = \begin{cases} 0 & 0 \leq t < 1 \\ 1 & 1 \leq t \leq 10 \end{cases}$$

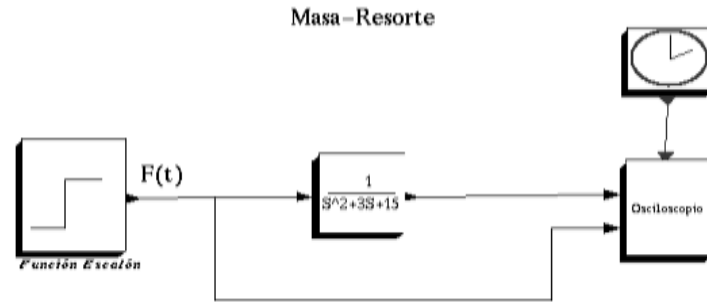


sistema masa resorte amortiguador

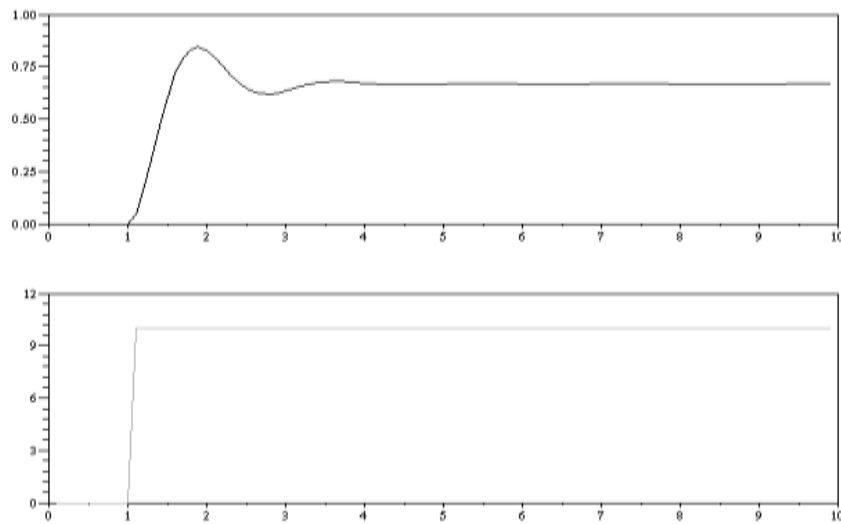
Del diagrama de cuerpo libre de la masa obtengo:

$$F(s) = [ms^2 + ks + B]X(s) \quad \frac{X(s)}{F(s)} = \frac{1}{ms^2 + ks + B}$$

Scicos:

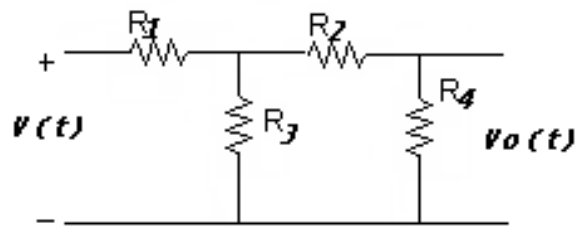


Simulación:

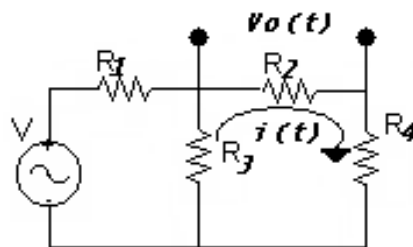


Ejercicios: Modele los siguientes circuitos, observe a través de un Scope el comportamiento de las variables que se le indique:

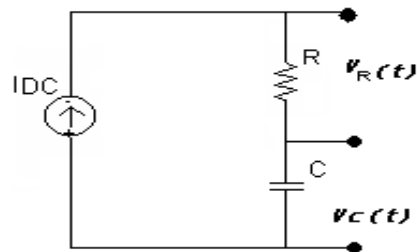
a) $R_1=5K\Omega$; $R_2=1K\Omega$; $R_3= 2.2K\Omega$; $R_4=1K\Omega$. Observe y comente $V(t)$ vs. $V_o(t)$



b) $V(t)$: Amplitud = 10V, Frecuencia = 60Hz, $R_1=1K\Omega$; $R_2=680\Omega$; $R_3= 2K\Omega$; $R_4=1K\Omega$. Observe y comente $V(t)$, $V_o(t)$, $i(t)$.

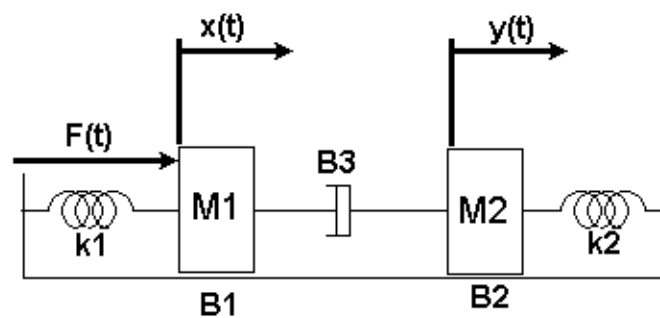


c) $I(t)= 8A$, $R=1K\Omega$; $C= 0.025f$. Observe y comente $I(t)$, $V_r(t)$, $V_c(t)$.



d) Dado el siguiente sistema mecánico, con $M_1=1\text{kg}$; $M_2=$; $B_1=$; $B_2=$; $B_3=$
 $k_1=3$; $k_2=$;
 $0 \dots t < 1$
 $20 \dots t \geq 1$

$F(t) = \begin{cases} \dots \end{cases}$



Ejercicios Propuestos por el Profesor.

Paginas en internet::

www.scilab.org/

www.dim.uchile.cl/~labma33a/Scilab/Intro_Spanish.ps

www.poli.usp.br/d/pmr5215/Introscilab.pdf -

www.geocities.com/lioraghershman/scicos.htm

<http://atmori.info/engineer/scilab/scicos/index.htm>