LABORATORIO DE COMPUTACIÓN
DE ALTO DESEMPEÑO (LCAD)

FACULTAD DE INGENIERÍA
UNIVERSIDAD NACIONAL DE ASUNCIÓN

INTRODUCCIÓN

A

SCILAB

(Programa de cálculo numérico)

Febrero/Marzo 2005

Objetivo

Este folleto es preparado como material de apoyo y complemento a las clases a ser desarrolladas dentro del curso ¨**Introducción a SCILAB**¨.

El objetivo es que el alumno cuente con un material impreso que le pueda servir de repaso y guía, que pueda ser utilizado como consulta rápida a la hora que se encuentre desarrollando sus programas en SCILAB.

Con este curso se pretende promover el uso de un programa computacional de cálculo numérico, herramienta básica en cualquier rama de la Ingeniería, de manera a que la misma sea paulatinamente utilizada como medio de profundización y facilitador de la enseñanza en las diferentes materias de las distintas carreras de la FIUNA.

Este material fue preparado en el LCAD. Comentarios, sugerencias o correcciones son bienvenidos.

Objetivo 2

1 Introducción 4
1.1 Qué es SCILAB? 4
1.2 Dónde encontrar SCILAB? 4
1.3 Qué se puede hacer con SCILAB? 5

2 Ambiente SCILAB 5

3 Resumen de comandos y operadores 6
3.1 Operaciones Básicas 6
3.2 Gráficos 6
3.3 Programación 7
3.4 Vectores 10
3.5 Matrices and Álgebra Lineal 10
3.6 Números complejos 11
3.7 Polinomios 12
3.8 Resolución de ecuaciones no lineales 13
3.9 Integración numérica 13
3.10 Resolución de Ecuaciones Diferenciales Ordinarias 14

4 Otras Herramientas 14

5 Referencias Bibliográficas 15

1 Introducción

1.1 Qué es SCILAB?

SCILAB es un programa desarrollado de forma a disponer en un sólo ambiente herramientas de cálculo numérico, programación y gráficos. El mismo fue desarrollado por el INRIA (Institut Nationale de Recherche en Informatique et en Automatique) y el ENPC (Ecole Nationale des Ponts et Chaussées) de Francia. Es similar a MATLAB y otros programas de cálculo numérico. Puede ser utilizado en una variedad de sistemas operativos tales como UNIX, Windows, Linux, etc.

A partir de Mayo de 2003, el programa pasa a ser mantenido por un conjunto de instituciones y empresas francesas denominado Consorcio SCILAB. Los objetivos principales del mismo son:

- Organizar la cooperación e intercambio entre los desarrolladores de SCILAB, con vistas a incorporar dentro del programa los últimos avances científicos en el área de computación numérica;
- Organizar la cooperación e intercambio entre usuarios de SCILAB de forma a que el programa pueda ser utilizado en forma efectiva en la industria, educación e investigación.

Desde el punto de vista del usuario, SCILAB presenta algunas ventajas tales como:

- Disponibilidad de la última versión vía Internet;
- El programa puede ser utilizado, copiado y distribuido en forma legal;
- Los resultados obtenidos pueden ser divulgados sin restricción;
- Se tiene acceso al código fuente;
- La certeza de estar participando de una comunidad cuyo principal objetivo es la difusión irrestricta del conocimiento.

1.2 Dónde encontrar SCILAB?

SCILAB es disponible en forma gratuita en sitio web oficial de SCILAB: http://scilabsoft.inria.fr.

Documentación referente a SCILAB puede ser encontrada y bajada desde el mismo sitio web de SCILAB, así como muchas otras obras y colaboraciones disponibles en Internet. Normalmente, todos los materiales pueden ser utilizados sin costo, solamente se deben mantener los créditos y referencias correspondientes para los autores.

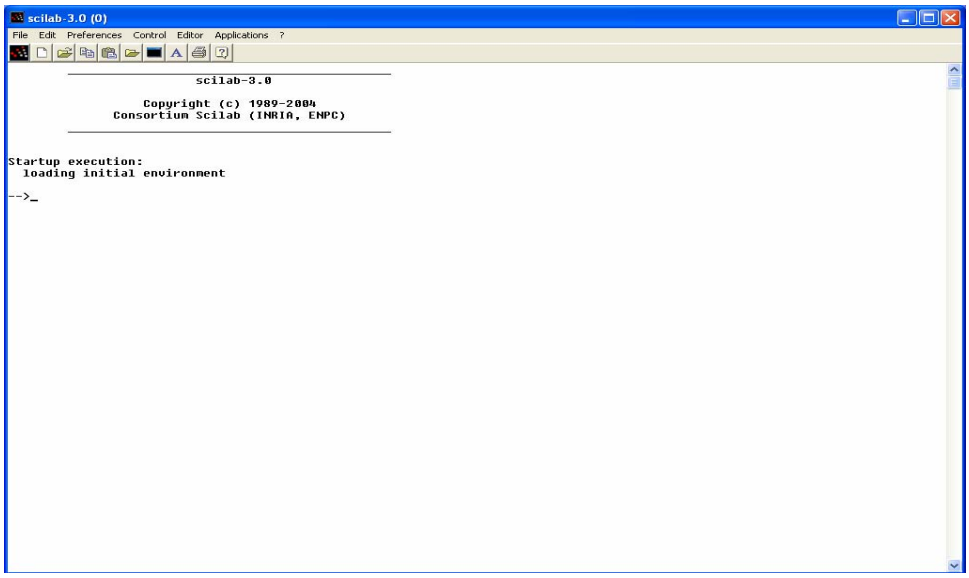
1.3 Qué se puede hacer con SCILAB?

Como ya fuera dicho, SCILAB es un ambiente de programación flexible cuyas principales características y prestaciones son:

- Programación con lenguaje simple y fácilmente asimilable;
- Posee capacidades de generación de gráficos en dos y tres dimensiones;
- Permite operaciones diversas operaciones matriciales;
- Permite operaciones con polinomios y funciones de transferencia;
- Permite la resolución de sistemas de ecuaciones lineales y ecuaciones diferenciales;
- Posibilita al usuario la creación y definición de funciones propias;
- Soporta la creación y utilización de conjuntos de funciones destinadas a aplicaciones específicas denominados "*Toolboxes*", por ejemplo: Control, Optimización, Redes Neurales, etc.

2 Ambiente SCILAB

En la siguiente figura se ve la ventana de trabajo de SCILAB.



En la barra de herramientas se tienen diferentes opciones, entre las cuales se puede mencionar como más importantes:

- **File**: para manejo y ejecución de archivos
- **Editor**: que inicializa el editor de archivos de comandos y funciones
- **Control**: con las funciones *resume*, *abort* e *interrupt*, que permiten moverse dentro de diferentes *workspace* (ambiente de trabajo)

También es importante recordar, que se dispone de un menú de ayuda **Help**.

3 Resumen de comandos y operadores

3.1 Operaciones Básicas

- SCILAB muestra el siguiente símbolo indicando que el programa está listo para ejecutar la siguiente instrucción. Esto se conoce como *prompt*: -->
- Las variables van siendo cargadas al *workspace* mediante asignaciones:
  -->a = 2.3
- Para ver las variable las variables activas se utiliza:
  --> who
- Existen variables pre-definidas, por ejemplo: %e, %i, %pi, %eps, %inf, %nan, etc.
- El operador : (dos puntos) sirve para crear un vector fila, por ejemplo:
  -->nombre del vector = valor inicial : incremento : valor final
- El operador ; (punto y coma) evita la impresión en pantalla de la salida del comando
- Para obtener la transpuesta de una matriz se debe usar un apóstrofe ’
- Para introducir un comentario y no ejecutar la línea se usa: //
- Para abrir el menú de ayuda se usa: --> help
- Para empezar a guardar una sesión de SCILAB en un archivo se usa:
  --> diary(nombre_del_archivo)
- Para terminar de guardar la sesión de SCILAB se usa: --> diary(0)
- Para ver el directorio actual de trabajo: --> pwd
- Para cambiar el directorio de trabajo: --> chdir(‘nombre de nuevo directorio’)
- Para listar archivos existentes en el directorio de trabajo: --> ls
- Para correr un archivo de comandos (*script*) se usa: --> exec(‘nombre de archivo’)

3.2 Gráficos

- Para gráficos simples en 2 dimensiones (2-D):
  --> plot(x,y, "título_eje_x","título_eje_y","título_del_gráfico")
- Agregando un grilla para gráficos simples en 2 dimensiones (2-D):
  --> grid(n)

- 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()

- Para gráficos simples en 2 dimensiones (2-D):
  --> plot2d(x,y,[style,strf,leg,rect,nax])

  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@l 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.

- Gráficos en 3 dimensiones:
  --> plot3d(x,y,z[,theta,alpha,leyenda,flag,ebox])
  Donde "theta" y "alpha" son los ángulos (en grados sexagesimales) representado las coordenadas esféricas del punto de vista, "leyenda" contiene las leyendas identificadoras de los ejes.

- Contorno en 3 dimensiones:
  --> contour(x,y,z[,theta,alpha,leyenda,flag,ebox])

- Contorno (curvas de nivel) en 2 dimensiones :
  --> contour2d(x,y,z[,theta,alpha,leyenda,flag,ebox])

3.3 Programación

- Operadores de comparación: = =, <, >, <=, >=, <> o ~=

- Operadores lógicos: & (and), | (or), ~ (not)

- Lazo FOR:
  for índice = valor_inicial : incremento : valor_final
  <comandos o instrucciones>
  end

- Lazo WHILE:
  while condición
  <comandos o instrucciones>
  end

- Condicional IF:
  if condición **then**
  <comandos o instrucciones>
  else
  <comandos o instrucciones>
  end

- Selección de casos con SELECT-CASE:
  select nombre_variable
  case valor_1
  <comandos o instrucciones>
  case valor_2
  <comandos o instrucciones>
  ….
  end

- Definición de funciones en una sola instrucción:
  deff(‘[variable_salida] = nombre_funcion(variable_entrada)’,[‘ variable_salida = definición de la función’])

  Observación: El corchete alrededor de la variable de salida en el primer argumento de **deff**, indica que la variable de salida es opcional cuando se evalúa la función. El corchete en el segundo argumento de **deff** es solamente requerido si se define más de una función.

  Ejemplos de definiciones:
  --> deff(‘[y] = f(t)’,’y = exp(-t)*sin(2*t)’)
  --> deff(‘[x,y] = h(r,theta)’,[’x = r*cos(theta)’,’y = r*sin(theta)’])

- Definición de funciones usando archivos (extensión del archivo **sci**) :
  Primera línea del archivo debe empezar con:
  Function [y1,…,yn] = nombre_funcion(x1,…,xm)

  y1,…,yn son las variables de salida, creadas/definidas en la función
  x1,…,xm son las variables de entrada

  Se recomienda que el nombre del archivo sea el mismo que el de la función.

La función puede ser creada usando el editor.

Para utilizar una función primeramente debe ser cargada usando el comando getf:
--> getf(´nombre_de_archivo_de_función')

La última línea del archivo debe ser:
EndFunction

- Continuación de una línea: en caso que requiera dividir una línea de comando en más de una línea, se tienen que colocar los caracteres (…) al final de la línea:
  Ejemplo: A = [1 2 3 …
  4 5 6]
  Se definió un vector de 1 fila y 6 columnas.

- Variable Global: son aquellas variables definidas en el ambiente principal SCILAB.

- Variable Local: son aquellas variables definidas solamente dentro de una función.

- Guardando las variables en un archivo:
  --> save(´nombre_de_archivo', lista_de_variables)

- Cargando las variables de un archivo:
  --> load(´nombre_de_archivo')

- Imprimiendo en archivo de salida sin formato:
  --> print(´nombre_de_archivo', lista_de_variables)

- Creando un nuevo archivo de salida:
  --> nombre_unidad = file(´open', ´ nombre_de_archivo', 'new')

- Escribiendo en un archivo de salida:
  --> write(´nombre_unidad', lista_de_variables, '(formato)')

- Formato para escribir a un archivo: los formatos usados son estilo FORTRAN, ejemplo:
  Iw : i para entero, w = número de caracteres permitido (ancho del campo)
  fw.d : f para coma-flotante, w = ancho de campo, d = número de decimales
  ew.d : e para coma-flotante con notación científica, w = ancho del campo, d = número de decimales
  aw : a para alfa-numérico (string - cadena), w = ancho del campo, a : ancho de la variable

- Entrada de datos desde teclado:
  --> [x]=input("Mensaje",["string"])
  "String" debe ser incluído en caso que la entrada sean caracteres

- Abriendo un archivo existente:
  --> nombre_unidad = file(´open', ´ nombre_de_archivo', 'old')

- Leyendo una matriz de un archivo:

--> matriz_mxn = read(nombre_unidad, m,n)

- Cerrando un archivo:
  --> file('close', nombre_unidad)

- Concatenando cadenas: cadena1 + cadena2
  --> cadena1 + cadena2

- Extrayendo partes de una cadena
  --> part(cadena, [n,m])

- Longitud de una cadena
  --> length(cadena)

- Localiza caracteres dentro de una cadena
  --> strindex(cadena, sub-cadena)

- Convierte un número a caracter
  --> string(número)

- Concatenación de caracteres
  --> strcat(cadena_1,cadena_2)

- Convierte una cadena en número
  --> evstr(cadena)

- Mostrar en pantalla una cadena de caracteres
  --> disp(cadena)

3.4 Vectores

- Magnitud de un vector: norm(vector)

- Transpuesta de un vector: '

- Mínimo y máximo de los valores de un vector: min(vector) max(vector)

- Producto escalar de dos vectores fila: u*v'

- Operación término a término de matrices: usar el punto (.) antes del operador, ejemplo: .*, ./ ,.^2

3.5 Matrices and Álgebra Lineal

- Transpuesta de una matriz/vector: usar el apóstrofe: ejemplo: A'

- Inversa de una matriz: inv(matriz)

- Matriz identidad: eye(n,n)

- Traza: trace(matrix)
- Dimensiones de una matriz/vector: size(matriz)
- Matriz con elementos aleatorios: rand(n,m)
- Extrayendo filas: A(2,:), A(1:3,:)
- Extrayendo columnas: A(:,1), A(:,2:5)
- Para acceder a la última fila/columna de una matriz usar el símbolo "$", por ejemplo: A(:,$) extrae la última columna de la matriz A
- Concatenando vectores fila: rv = [], rv = [rv r1], rv = [rv r2]
- Concatenando vectores columna: B = [], B = [B;r1], B = [B;r2]
- Suma y producto de elementos de una matriz por columnas: sum(B,'r'), prod(B,'r')
- Suma y producto de elementos de una matriz por columnas: sum(B,'c'), prod(B,'c')
- Solución de un sistema lineal A*x = b : --> xsol = A\b
- Factorización LU: [L,U] = lu(A) or [L,U,P] = lu(A)
- Descomposición en valores singulares: [U,S,V] = svd(A)
- Rango de una matriz: rank(matriz)
- Normas de una matriz:
  norm(A) o norm(A,2) : norma Euclidiana
  norm(A,1) : norma columna
  norm(A,'inf'): norma infinita
  norm(A,'fro') : norma Frobenius
- Número de condición de uma matriz: cond(matriz)
- Determinante de una matriz: det(matriz)
- Autovalores y autovectores de una matriz: spec(matrix)

3.6 Números complejos

Si z es un número complejo z = x + iy, donde i = $\sqrt{-1}$ (%i en SCILAB)

- Para hallar la parte real de z: real(z)

- Para hallar la parte imaginaria de z: imag(z)
- Para hallar la representación polar de z: [r,theta] = polar(z)
- Para hallar la magnitud : abs(z)
- Para hallar el argumento: arctan(Im(z)/Re(z))
- Para hallar el complejo conjugado: conj(z)
- Para hallar el negativo: -z
- En general las matrices definidas en SCILAB pueden tener como argumentos a números complejos.

3.7 Polinomios

- Los polinomios son creados en SCILAB usando el comando poly
- Definiendo una variable polinomial: --> x = poly(0,'x')
- Definiendo un polinomio en base a sus coeficientes:
  --> Polinomio = poly(c,'x','coeff')

  Donde c es un vector que contiene los coeficientes en orden creciente
  c = [ao a1 a2 a3 …an]
- Definiendo un polinomio en base a sus raíces:
  --> Polinomio = poly(r,'x')

  Donde r es un vector que contiene las raíces del polinomio r = [r1 r2 …rn]
- Para hallar las raíces de un polinomio:
  --> raices = roots(Polinomio)
- Una vez definidos dos polinomios p y q de la misma variable, se pueden efectuar operaciones de polinomios: ejemplo: p*q, p/q, p+q, etc.
- Para evaluar un polinomio:

  --> valor_del_polinomio = horner(Polinomio,x)

  Donde x es el punto en el cual se está evaluando el polinomio.
- También pueden definirse matrices cuyos elementos sean polinomios.

3.8 Resolución de ecuaciones no lineales

- Para resolver una ecuación o un sistema de ecuaciones no lineales en SCILAB, se puede usar el comando:
  --> [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,
  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. ($1x10^{-10}$ es el valor por defecto),
  x : vector real que contiene el valor final de la solución aproximada,
  v : vector real que contiene el valor de la función en x,
  info : indicadores de culminación
  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.

3.9 Integración numérica

- SCILAB permite realizar integración numérica utilizando la regla del trapecio usando el comando: , se puede usar el comando:
  --> [v] = inttrap([x,] y),

  Donde:
  x es un vector de valores crecientes de abscisa
  y es el vector conteniendo las ordenadas de los puntos de la función
  v es el valor de la integral

- Para obtener integrales por cuadratura se usa
  --> [x] = integrate(f,v,a,b [,ea [,er]])

  Donde:
  f es la función a ser integrada
  v es la variable de integración introducida como cadena
  a,b son números reales que determinan los límites de integración
  ea ,er son números reales indicando el valor de error absoluto y relativo deseado

- También puede ser utilizado el comando:
  --> [I,err]=intg(a,b,f [,ea [,er])

  Donde:

a,b son los límites de integración
f es la función a ser integrada
ea es el error absoluto requerido
I es el valor de la integral
Err es el valor estimado del error absoluto

3.10 Resolución de 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 (x0,y0).

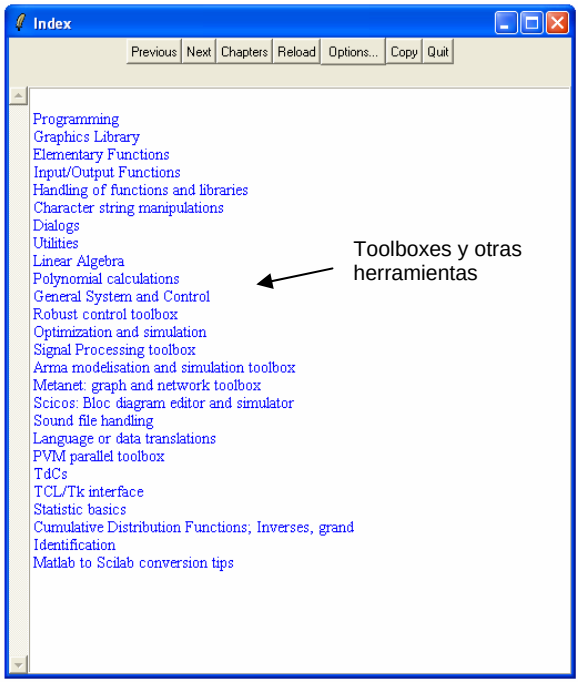
  El comando a ser utilizado es:
  --> [y] = ode([type],y0,x0,x,f)

  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’
  y0 : vector conteniendo las condiciones iniciales
  x0 : 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

4 Otras Herramientas

A más de las funciones y comandos aquí descriptos y mencionados, SCILAB trae muchas otras herramientas como ser:

- Toolbox de análisis de sistemas y control,
- Toolbox de control robusto,
- Toolbox de optimización,
- Toolbox de procesamiento de señales,
- Toolbox para procesamiento paralelo,
- Y otros.



No olvidar, que el primer material de consulta de las posibilidades que ofrece el SCILAB debe ser el menú de ayuda incluida en el programa (HELP).

5 Referencias Bibliográficas

Este material fue preparado en base a los siguientes materiales obtenidos de Internet:

- SCILAB Group, “Introduction to SCILAB - User’s Guide”. Referencia preparada por el Grupo de Soporte de SCILAB y disponible en el siguiente sitio web: http://scilabsoft.inria.fr/doc/intro/index..html
- Gilberto E. Urroz, “SCILAB”, materiales disponibles en: http://www.engineering.usu.edu/cee/faculty/gurro/Scilab.html
- Paulo Sérgio da Motta Pires, “Introdução ao SCILAB”, Dpto. de Engenharia de Computação e Automação, Universidade Federal do Rio Grande do Norte, Natal, Brasil.