UNIVERSIDAD NACIONAL DE SALTA

FACULTAD DE CIENCIAS NATURALES

SEDE REGIONAL TARTAGAL

“PROGRAMACIÓN Y SIMULACIÓN UTILIZANDO SCILAB”

(CAPITULO 1)

Dr. Nahuel Salvo
TUP. Gonzalo Barroso

Capítulo 1:

1.- Introducción

En general existen dos categorías de software científico: sistemas de álgebra computacional que realizan cálculos simbólicos y sistemas que realizan cálculos numéricos diseñados específicamente para aplicaciones científicas. En la primera categoría están incluidos programas como Maple, Mathematica y Máxima. El segundo grupo está representado principalmente por Matlab y se incluye aquí a Scilab.

Antes de comenzar con el estudio de Scilab es necesario realizar algunos comentarios con respecto a sus características y orígenes. En principio puede decirse que es uno de los programas y lenguajes más usados a nivel académico y esto se debe principalmente por dos motivos: uno es su potencia de cálculo y otro la claridad o facilidad de las sintaxis cuando se realiza (por dar un nombre) *"programación científica"*. También es necesario e importante hacer notar que es la contra parte libre del paquete de cálculo científico *"Matlab"*.

Una característica de SCILAB es que constituye un programa desarrollado de forma tal que en un solo ambiente de trabajo se disponen de herramientas de manejo de archivos, cálculo numérico, programación, gráficos, etc.

Este programa ha sido desarrollado por el INRIA (Institut Nationale de Recherche en Informatique et en Automatique) y el ENPC (Ecole Nationale des Ponts et Chaussées) de Francia y puede ser empleado en diferentes sistemas operativos tales como UNIX, Windows, Linux, etc. A partir de Mayo de 2003, el programa pasó a ser mantenido por un conjunto de instituciones y empresas francesas denominado *"Consorcio SCILAB".* Este Consorcio actualiza con frecuencia el software, creando nuevas versiones, corrigiendo errores e implementando nuevas funciones, teniendo como principales objetivos lo siguiente:

• 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, la educación e investigación, etc.

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

- Disponibilidad de la última versión del software vía Internet en forma gratuita
- El programa puede ser utilizado, copiado y distribuido en forma legal
- Los resultados obtenidos pueden ser divulgados sin restricción alguna
- Se tiene acceso al código fuente de diferentes subrutinas lo que permite modificarlas o ampliarlas
- La certeza de estar participando de una comunidad cuyo principal objetivo es la difusión irrestricta del conocimiento

Scilab está disponible para su descarga en http://www.scilab.org/. Desde este sitio pueden descargarse diferentes versiones del programa y consultar información general, manuales y documentación relacionada. También existe un grupo de noticias (News) donde pueden intercambiarse experiencias, ideas y programas desarrollados en *Scilab* o

consultar numerosos links donde puede encontrarse información adicional sobre el software.

2.- Características principales de Scilab

Como se mencionó, Scilab es un paquete libre de código abierto para la computación científica que incluye muchas funciones de uso general y funciones especializadas para cálculo numérico.

Scilab tiene también disponibles varias formas de visualización de resultados que incluyen gráficas en 2D y 3D, contornos, dibujos paramétricos y animaciones. Los gráficos pueden ser exportados en diferentes formatos, como gif, postcript-latex, etc. Además de las funciones de interfaz de usuario propias, Scilab permite el empleo de la interfaz Tcl/Tk.

Como se mencionó, SCILAB constituye es un ambiente de programación cuyas principales características y prestaciones son:

- Programación con lenguaje simple y de fácil entendimiento
- Posee una capacidad de generación de gráficos en dos y tres dimensiones
- Permite diversas operaciones matriciales
- Es posible operar con con polinomios y funciones de transferencia en variable compleja
- Para problemas descriptos a partir de un sistema de ecuaciones, es posible resolverlo inclusive para el caso de sistemas de ecuaciones diferenciales
- Posibilita para el usuario la creación y definición de funciones para algún cálculo específico
- Además, soporta la creación y utilización de un conjunto de funciones, organizadas en librerías, destinadas a aplicaciones específicas denominadas “Toolboxes” que cubren áreas tales como la simulación, optimización, control, procesamiento de señales, redes neuronales, etc.

Es importante resaltar que el uso de Scilab con todas sus herramientas permite realizar una gran cantidad de operaciones con una o varias colecciones de valores numéricos, como por ejemplo: búsqueda de uno o varios datos en un determinado archivo, comparación de una serie de valores, resolución numérica de sistemas de ecuaciones, simulaciones de fenómenos relacionados con la matemática, física, ingeniería o cualquier rama donde se cuente con datos numéricos.

Finalmente en los diferentes capítulos que constituyen este documento se presentan algunos comandos de Scilab que permitirán programar entre otras cosas la manipulación y análisis de datos experimentales, obtener valores numéricos de expresiones algebraicas, ejecutar una serie operaciones matemáticas repetidas veces, graficar y ajustar valores numéricos obtenidos de cálculos, etc.

3.- Cómo comenzar

Para iniciar el trabajo con Scilab, habrá que instalarlo en la versión que corresponda con el sistema operativo en uso, luego de descargarlo de https://www.scilab.org/. En esta página también es posible obtener algunos documentos y tutoriales (en francies e inglés). Además, existen numerosos sitios en Internet donde puede encontrarse información adicional sobre diferentes temas relacionados con cálculo numérico utilizando Scilab. También es importante hacer notar que el consorcio Scilab actualiza con frecuencia el

software, creando nuevas versiones, corrigiendo errores e implementando nuevas funciones.

4.- La ventana de trabajo - Consola de Scilab

Una vez instalado el programa, cuando se lo ejecuta, aparece la ventana principal de Scilab (Console), que se muestra en la Figura 1.1. Esta ventana puede ser algo diferente de acuerdo a la versión instalada. Este documento está basado en la versión versión 5.5.2. que es la versión que trae por defecto el sistema operativo.

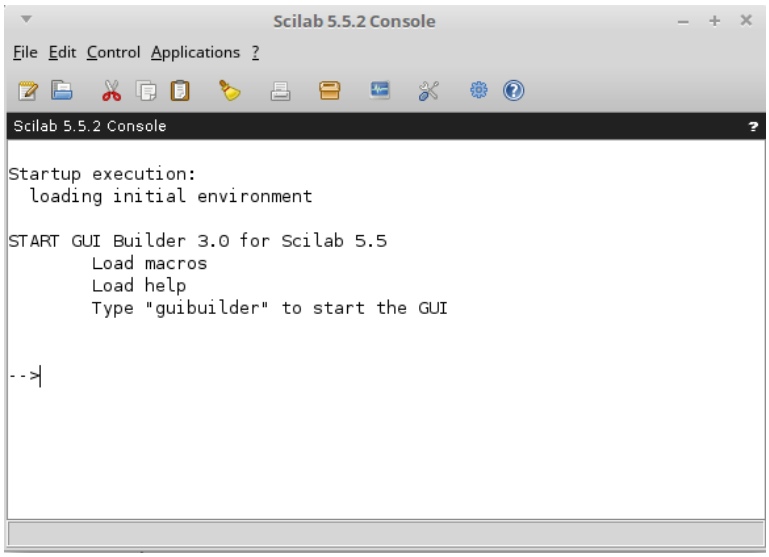

Figura 1.1: *ventana principal de Scilab*

4.1.- Barra de títulos

Presionando con el botón derecho del mouse sobre la barra “Scilab 5.5.2 Console” arriba de la ventana se despliega un menú donde se encuentran las opciones para manejar la apariencia de la ventana: Minimizar, Maximizar, Mover, Redimensionar, etc. (Figura 1.2)

4.2.- Barra de menú principal

En esta zona de la ventana se encuentran los menús necesarios para manejar el programa (File, Edit, Control, Applications, ?)

- *File – (archivo)*: Cuando se ejecuta con el mouse este menú, aparece una lista de comandos con diferentes opciones que permiten operar con los archivos de trabajo, como se muestra en la Figura 1.3.

- *Edit – (editar)*: En este caso la lista contiene opciones para: cortar, copiar, pegar, seleccionar texto, etc. mediante el uso del “clipboard” (porta papeles). Este menú se muestra en la Figura 1.4.

- *Control – (control):* La lista contiene opciones para controlar la ejecución de funciones. Las diferentes opciones se muestran en la figura 1.5.

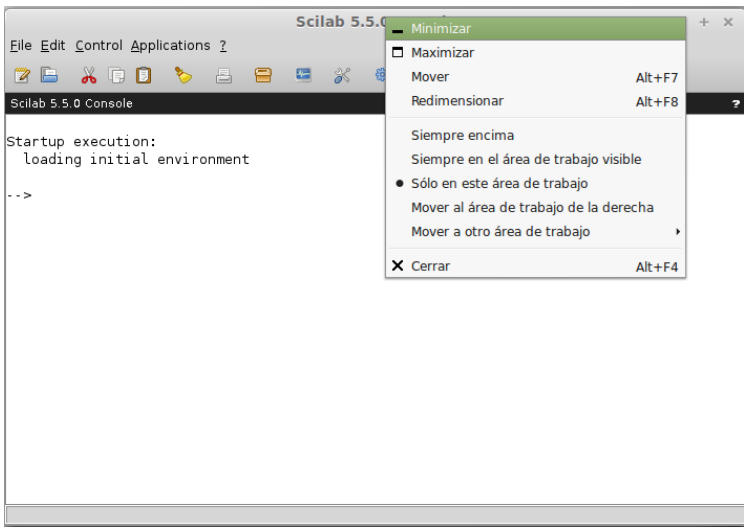

Figura 1.2: *menú de la barra Scilab 5.5.2*

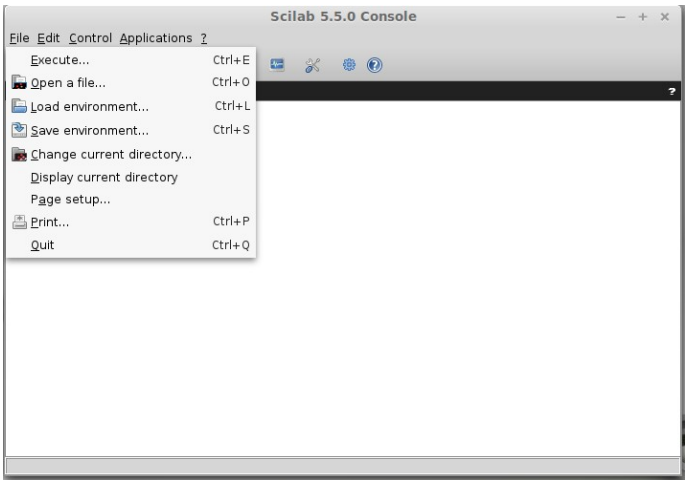

Figura 1.3: *Contenido del menú "File"*

- *Applications - (aplicaciones)*: En la ventana de este menú se pueden ejecutar algunas aplicaciones que funcionan bajo Scilab, como el editor (SciNotes), Scicos (Xcos) para la simulación de sistemas dinámicos, etc, como se muestra en la figura 1.6.

- *? - (ayuda)*: Este menú contiene las opciones de ayuda y demostraciones de Scilab como se muestra en la Figura 1.7.

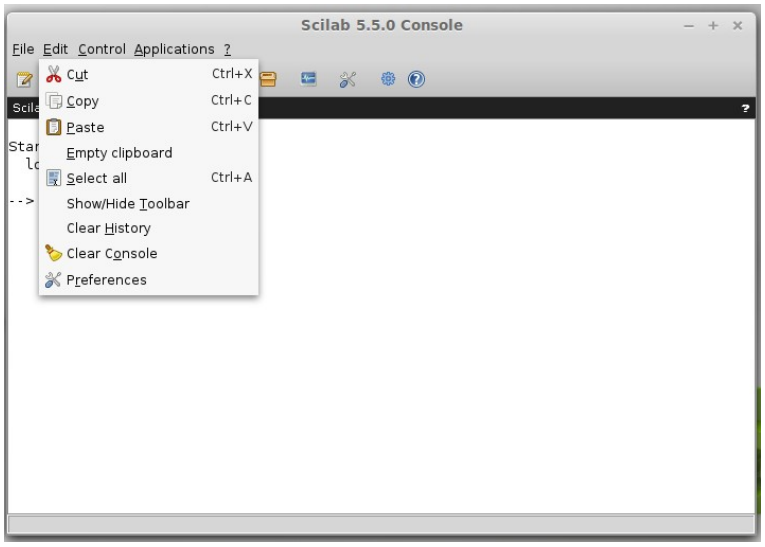

Figura 1.4: *Contenido del menú "Editar"*

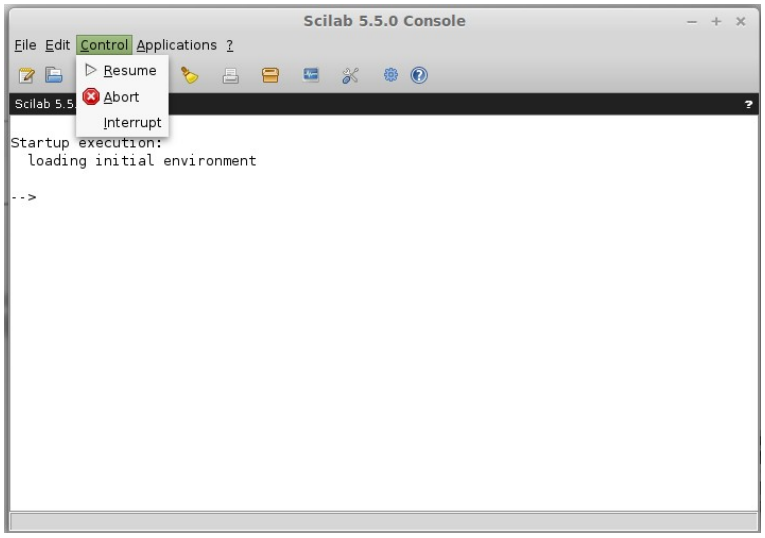

Figura 1.5*- Contenido del menú "Control"*

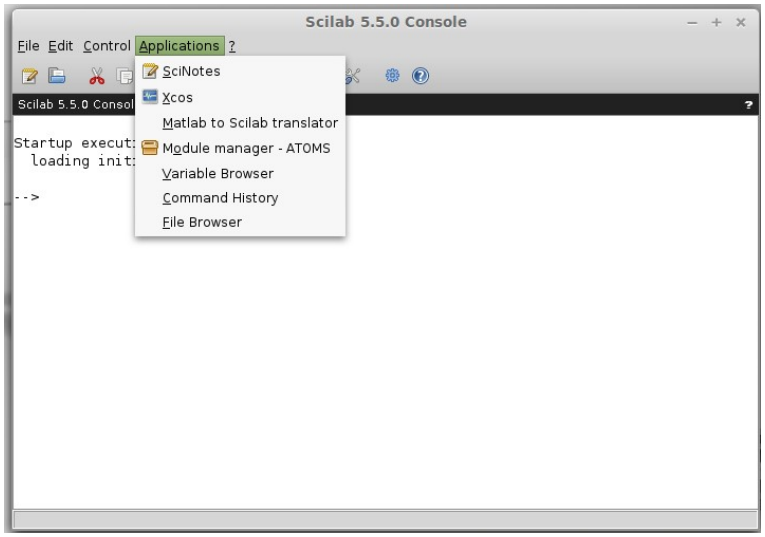

Figura 1.6: *Contenido del menú "Applications"*

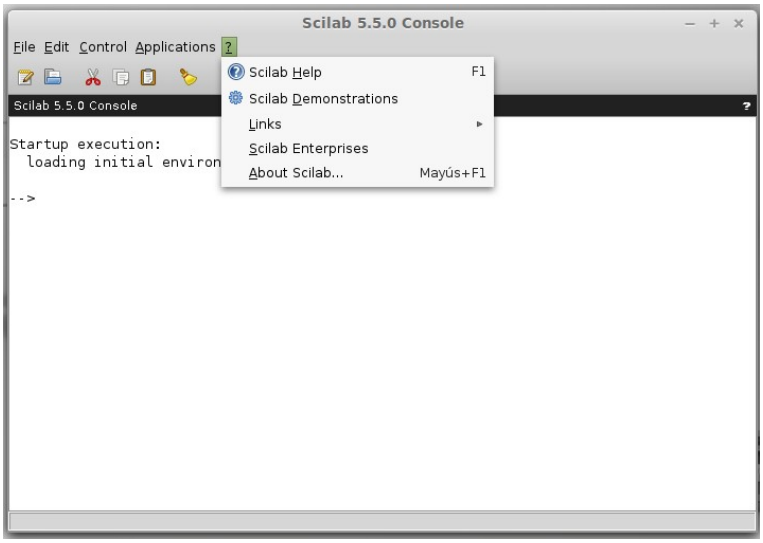

Figura 1.7: *Opciones del menú "?"*

4.3.- Barra de herramientas

Abajo del menú principal se encuentran una serie de iconos que constituyen la *"Barra de herramientas"*. Los iconos representan el acceso rápido a las opciones más comunes. (Figura 1.8)

Figura 1.8: *Iconos de acceso rápido*

4.4.- Zona de trabajo

Esta zona es la parte en blanco abajo de los diferentes menús y su principal característica es el prompt (-->). A continuación de él se introducen los comandos, como se explicará luego. Ver Figura 1.1.

La mejor forma de comenzar a explorar Scilab es ejecutar una demostración. Por ejemplo, en el menú "?", presionar sobre la opción "Demostraciones de Scilab" . Se abrirá una nueva ventana con un listado de diferentes opciones, seleccionar alguna de ellas y ejecutarla. La Figura 1.9 muestra algunas de las pantallas emergentes.

5.- Comandos Utilitarios

Como se mencionó en la zona de trabajo se escriben y ejecutan los diferentes comandos de Scilab y mientras dure la sesión los valores de las diferentes variables se mantienen activos o sea el programa las guarda en un espacio de memoria y se mantienen sin modificación salvo que el usuario las modifique por razones de programación. Cuando se cierra la sesión todas las operaciones y cálculos realizados se pierden. Puede suceder que cierta sesión sea necesaria para un trabajo futuro y para evitar que se pierda Scilab cuenta con una serie de comandos que permiten guardar una sesión y volver a ejecutarla posteriormente.

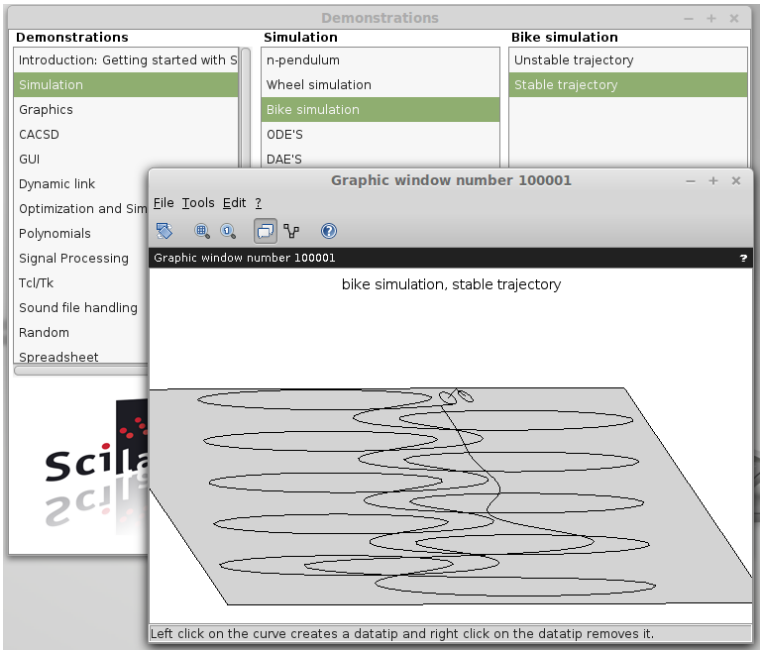

Figura 1.9: *Simulación Scilab, trayectoria de una bicicleta*

Para guardar los valores de las variables durante una sesión de trabajo y poder emplearlas posteriormente, en el menú *"File"* hay que seleccionar la opción *"Guardar entorno" (Save Enviroment)*. Al seleccionar esta opción se abre una ventana que permite elegir el directorio en el que se guardará el archivo y el nombre del mismo, que se guardará con extensión .sav . Cuando se abre nuevamente Scilab y en el menú *"File"* se selecciona la opción *"Cargar entorno" (Load Enviroment)*, Scilab recupera los valores de las variables de la sesión guardada.

Durante una sesión de trabajo y para facilitarlo Scilab mantiene en memoria todas las operaciones sucesivas que se están realizando. Si se quiere utilizar una de ellas a posteriori, basta presionar las teclas del cursor sucesivamente (flecha arriba o abajo) para recorrer la lista de comandos ejecutados. Todas las instrucciones ingresadas durante la sesión van apareciendo a continuación del prompt (-->). El usuario puede seleccionar cualquiera y ejecutarlo nuevamente. Otra forma de acceder al historial de comandos para ejecutar alguno nuevamente es a través del menú *"Applications"* en la opción *"Command History"* (Historial de Comandos). Presionando sobre esta opción se abre una ventana que contiene todos los comandos utilizados en la sesión y en anteriores que pueden volver a ejecutarse seleccionándolos con el botón derecho del mouse. Esto se muestra en la Figura 1.10.

Otra opción que ofrece Scilab para guardar una sesión de trabajo es el comando *"diary".*

Antes de explicar este comando es necesario saber en que directorio guarda Scilab por defecto los archivos generados, para esto se utiliza el comando *"pwd"* que permite conocer el directorio en el que se está trabajando. Para aclarar lo anterior ejecutar los siguientes comandos en la *"zona de trabajo"*, a continuación del prompt -->:

-->pwd <Enter>[1]

1 *<Enter> simboliza la acción de oprimir la tecla ↵ "Intro" o "Enter" dependiendo del teclado.*

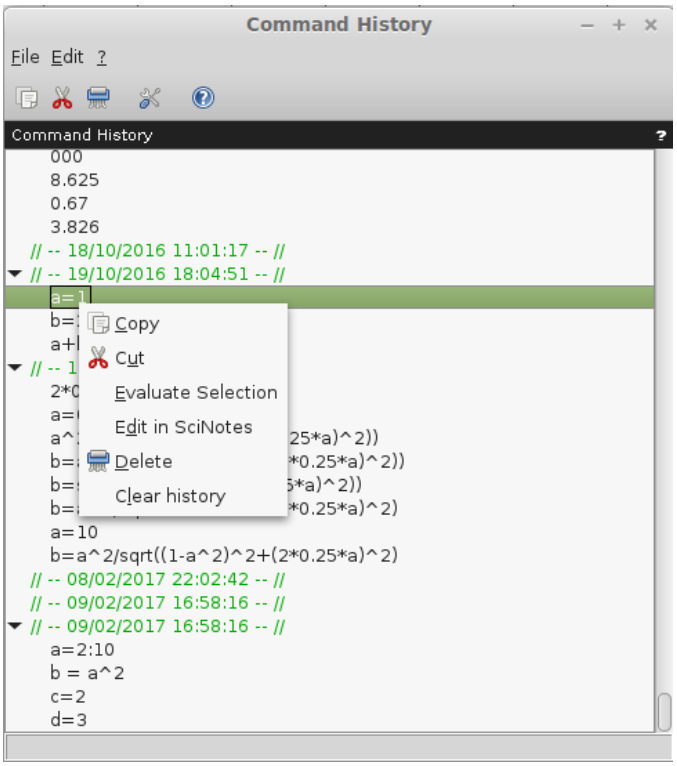

Figura 1.10: *Ventana de Command History (historial de comandos)*

el resultado muestra cual es el directorio sobre el que se esta trabajando o sea donde Scilab guardará los resultados de la sesión. Luego ejecutar:

-->diary('nombre de archivo.txt') <Enter>

Después de ejecutar el comando anterior en la ventana de trabajo, Scilab crea el archivo "*nombre de archivo.txt*" y escribe en él todas las instrucciones ejecutadas durante la sesión y los resultados obtenidos, lo que es diferente al comando anterior que solo almacena las instrucciones ejecutadas. Este archivo de texto generado se guarda en el directorio en el que se está trabajando. Si se quiere guardar en otro directorio hay que especificar el paso completo o ruta para llegar a él. Por ejemplo en Windows sería:

-->diary('C:\Mis Documentos\curso scilab\nombre de archivo.txt') <Enter>

Otra forma de trabajar, para no especificar la ruta al directorio de trabajo, es cambiar al inicio de la sesión de Scilab al directorio de trabajo y luego ejecutar la sentencia *"diary"* sin especificar la ruta. El cambio de directorio se realiza desplegando la ventana *"File"* y luego ejecutar Cambio de Directorio (*"Change Current Diretory"*). Ver Figura 1.11. Para cerrar el archivo de texto generado por *"diary"*, se debe ejecutar el comando:

-->diary(0) <Enter>

El archivo de texto guardado puede ser leído con cualquier editor de textos del estilo Bloc de Notas de windows o el editor de texto ASCII de linux, pero no con la opción del menú *"File"*, *"Load environment"* (Cargar entorno) de Scilab. [ver Figura 1.11]

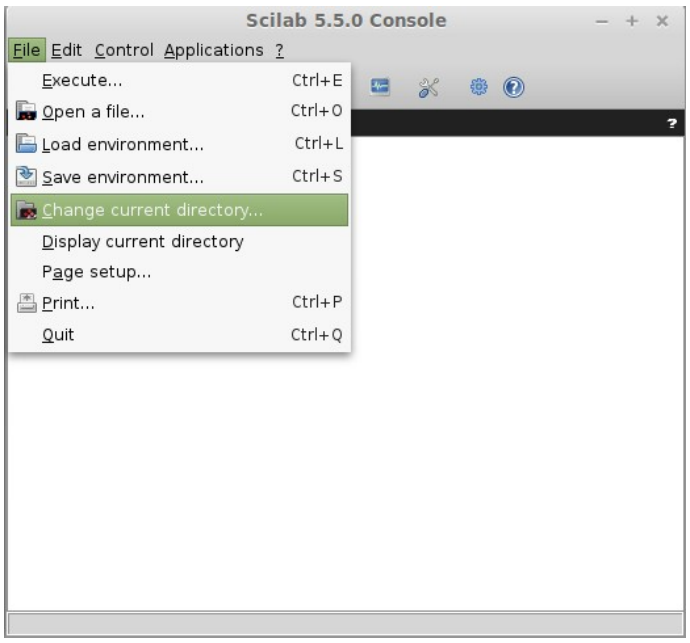

Figura 1.11: *Cambio de directorio de trabajo*

6.- Entrada, Ejecución y Salida

La consola de Scilab (ventana de comandos - Figura 1.1) es una ventana interactiva donde el usuario escribe un comando determinado a continuación del prompt "-->". El comando introducido será ejecutado por Scilab cuando se hace un *"retorno de carro"* ("Enter", *"Intro"* o *"Return"*). Una vez ejecutado, muestra el resultado y devuelve el control al usuario mostrando un nuevo prompt "-->". Se recomienda realizar, paralelamente con la lectura de este documento, la ejecución de los comandos que en cada caso se indica.

Es importante antes de ejecutar los ejemplos crear un directorio de trabajo, con el explorador de archivos, en el cual se almacenaran los archivos de trabajo, por ejemplo aquellos creados con *"diary"*.

Ejemplo 1-1: Ejecute las siguientes operaciones para sumar números:

-->1+1 <Enter>
-->124+37 <Enter>
-->124+37 // es una prueba <Enter>

Al ejecutar las operaciones del Ejemplo 1-1, debería obtenerse lo que se muestra en la Figura 1.12. Si después de una instrucción se agregan dos barras (//) y se escribe un comentario, Scilab interpreta que lo que sigue a las dos barras no debe ejecutarse. Esto es importante porque se pueden agregar comentarios sobre las instrucciones que se van a ejecutar sobre todo cuando se escriben programas.

Con respecto a la Figura 1.12 se debe notar que los resultados van seguidos de un punto, que para Scilab es la separación decimal.

Ejemplo 1-2: Ejecutar los siguientes comandos:

*-->2.1 * 3.27 <Enter>*
-->2.1/3.27 <Enter>

Hasta el momento se ha utilizado a Scilab como una calculadora, pero un aspecto importante que será empleado a lo largo del curso es que cualquier valor numérico puede ser asignado a una variable.

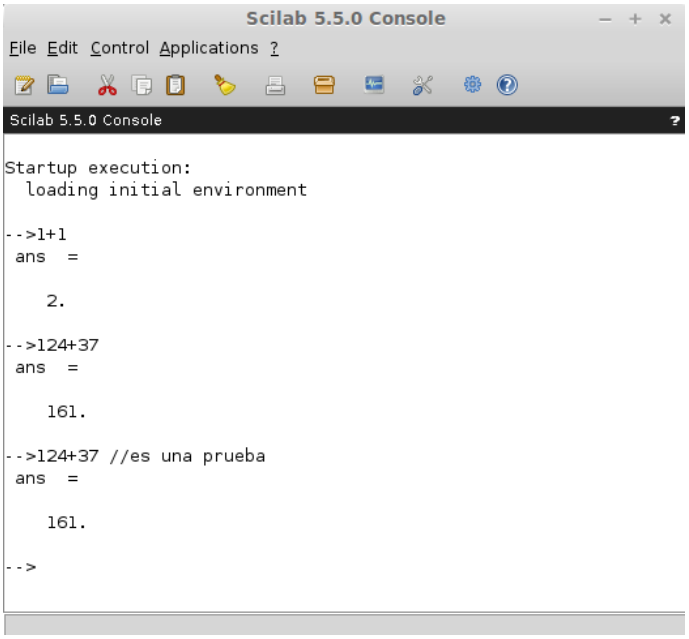

Figura 1.12: *Ejemplo 1, cálculos en Scilab*

En programación, las variables son espacios reservados en la memoria de la computadora y como su nombre indica, pueden cambiar de contenido (valor) a lo largo de la ejecución de un programa. Es saludable utilizar como nombres de variables algo nemotécnico para que de esta forma se pueda identificar fácilmente que tipo de dato almacena.

A continuación se explica como se definen variables en Scilab y tipos de datos que almacenan.

6.1.- Las variables y los tipos de datos

En Scilab pueden definirse *"variables"* en las que se almacenan valores numéricos o alfanuméricos. Estas variables quedan guardadas en la memoria RAM de la computadora y pueden utilizarse en diferentes cálculos mientras dure la ejecución del programa o no se redefinan. Una variable es creada simplemente asignándole un valor o sea ejecutando la orden *T = 1* se crea la variable *"T"* o *var = 2.3* se crea la variable *"var"*. El nombre de la variable puede ser cualquier letra o combinación de letras, también pueden tener por nombre una cadena alfanumérica, por ejemplo *"var12"*, *"B3"*, etc.

El nombre de las variables puede ser de cualquier longitud pero Scilab solo tiene en cuenta los primeros 24 caracteres. Por lo tanto para una mayor consistencia es necesario utilizar nombre de variables que no superen estos 24 caracteres. Todas las letras ASCII desde "a" hasta la "z", "A" hasta "Z", los dígitos desde *"0"* hasta *"9"* los caracteres adicionales *"%"*, "_", *"#"*, *"!"*, *"$"*, *"?"* son permitidos en el uso de nombre de variables. Aquellas variables cuyo primer símbolo sea "*%*" tienen un significado especial para Scilab y será analizado más adelante.

Es importante resaltar que esta versatilidad en la elección del nombre de las variables hace que sea posible emplear alguna regla nemotécnica en la definición, de forma tal que muestre su significado y esto puede apreciarse en el siguiente ejemplo:

Ejemplo 1-3: Asignar el resultado de sumar 15 + 24 a la variable "s" y el valor 2.4 a la variable "s1"

-->s=15+24 <Enter>
-->s1=2.4 <Enter>

Ejemplo 1-4: Determinar el espacio recorrido por un cuerpo que se mueve a una velocidad de 2.4 m/s en un tiempo de 15 s.

-->tiempo=15; <Enter>
-->velocidad=2.4; <Enter>
*-->espaciorecorrido=velocidad * tiempo; //movimiento uniforme <Enter>*
-->espaciorecorrido <Enter>

En el ejemplo 1-4, con las dos primeras instrucciones se almacenan dos números en las variables *"tiempo"* y *"velocidad"*, con la siguiente sentencia se calcula el espacio recorrido en el caso de un movimiento uniforme, almacenando el resultado en la variable *"espaciorecorrido"* y en la cuarta sentencia se muestra el valor almacenado en dicha variable. Hay que notar que las tres primeras instrucciones terminan con un punto y coma (;) antes de realizar <*Enter*>, esto hace que Scilab no muestre *"ans = "* como sucede en la Figura 1.12.

Ejemplo 1-5: Ejecutar los siguientes comandos:

-->s1=3.056; <Enter>
-->s=3.4; <Enter>
-->a=s+s1; <Enter>
*-->a2t=s*a+s1; <Enter>*
-->suma1=a+a2t <Enter>
-->who <Enter>

En el Ejemplo 1-5, Scilab para las cuatro primeras sentencias no muestra el resultado pero las variables "a" y "a2t" tienen un valor numérico, como lo demuestra la quinta sentencia.

El último comando ("who") devuelve la lista de las variables activas en la sesión de trabajo. Se ven las variables definidas por el usuario en los ejemplos desarrollados y variables propias del sistema, algunas de las cuales no pueden ser modificadas por el usuario y a esto se hacia referencia en párrafos anteriores para aquellas variables que inician con el signo "%". En Scilab algunas variables matemáticas están predefinidas y sus nombre comienzan con el signo "%" en el Cuadro 1.1 se muestran algunas.

%i	Número imaginario
%e	Constante de Euler
%pi	Número Pi

Cuadro 1.1: *variables predefinidas*

Ejemplo 1-6: Ejecutar los siguientes comandos:

-->%pi <Enter>
-->%i%i <Enter>*
-->%pi=8 <Enter>

Después del último comando, en el ejemplo 1-6, Scilab entrega un mensaje de error, debido a que la variable %pi no puede ser modificada. En particular %pi es una constante que toma el valor del número π = 3.1415927.....

En computación y en particular en Scilab, pueden usarse distintos tipos de datos: numéricos enteros, numéricos flotantes, textos alfanuméricos (denominados *"string"*), etc. Scilab determina el tipo de variable a partir del tipo de dato asignado a la misma, también se puede cambiar de tipo de variable simplemente asignando un nuevo tipo de dato.

Por ejemplo, se puede asignar a la variable, *"var1"*, un número y en este caso la variable será numérica, si luego se le asigna a la misma variable, *"var1"* un *"string"* o sea una palabra, la variable cambia de numérica a alfanumérica. Esto es una característica importante y que diferencia a Scilab de otros software donde las variables deben ser predefinidas al inicio de un programa. En otras palabras Scilab tiene un manejo muy dinámico del tipo de variable.

Los tipos de datos básicos que maneja Scilab son enteros, reales (en sus distintas notaciones), complejos, cadenas de caracteres.

Ejemplo 1-7: Asignar a diferentes variables los datos que se listan a continuación y realizar algunas operaciones básicas como suma, resta, multiplicación, etc. entre ellos.

4532
3.5
-4.1234
3.14e-10
3.14E-10
0.0035e20
-31.45e+11

Ejemplo 1-8: Asignar a dos variables diferentes las cadenas de caracteres alfanuméricos que se listan a continuación y luego mostrar la variable en la ventana de Scilab.

scilab
scilab 2.7

Ejemplo 1-9: Repetir el ejemplo 1-8 encerrando las cadenas de caracteres entre comillas y operando con ellas.

-->a= "sci"; <Enter>
-->b= "lab"; <Enter>
-->a+b <Enter>
-->a1= "scilab"; <Enter>
-->b1= " 2.7"; <Enter>
-->a1+b1 <Enter>

Como se aprecia a partir de los dos ejemplos anteriores, ejemplo 1-8 y 1-9, los textos alfanuméricos (string) deben introducirse entre comillas para que Scilab los reconozca como tales y no los confunda con otro tipo de variable, como sería el caso de introducir 2.7 sin comillas, que sería una variable numérica. En el ingreso de la variable *"b1"* se separó a 2.7 por un espacio y como esta encerrado entre comillas, Scilab toma al espacio como parte de la definición de la variable. La única operación definida para un string es la suma, para las otras operaciones Scilab entrega un mensaje de error. Verificar esto realizando el producto de las variables a y b del ejemplo 1-9.

Más adelante se verá que Scilab utiliza variables con características más complejas. Esto permite que se puedan almacenar y operar con ellas no solo números reales sino que también es posible manejar números complejos, matrices y polinomios, etc. En particular la posibilidad de almacenar matrices de números en una variable y realizar diversas operaciones con ellas hace que Scilab pueda llevar a cabo operaciones complicadas con suma facilidad, convirtiéndolo en un poderoso lenguaje de cálculo.

Las mayúsculas y minúsculas son distinguidas por Scilab. Esto significa que, por ejemplo, *"sa"* y *"Sa"* son dos variables diferentes, es decir se guardan en espacios separados en la memoria.

Ejemplo 1-10: Realizar con Scilab las operaciones que se indican a continuación:

```
-->sa = 2.56; <Enter>
-->Sa = 3.1e3; <Enter>
-->sa <Enter>
-->Sa <Enter>
```

Como se puede apreciar a partir del ejemplo 1-10 Scilab diferencia las variables. Siempre hay que tener presente, como se mencionó, que los nombres (identificadores) de las variables pueden tener hasta 24 caracteres de longitud.

Scilab posee una variable predeterminada *"ans"* que almacena el último valor calculado y el uso de la misma permite realizar operaciones posteriores empleando este valor como se puede comprobar en el siguiente ejemplo.

Ejemplo 1-11: Realizar las siguientes operaciones:

```
-->1+4; <Enter>
-->ans+3 <Enter>
-->ans/4 <Enter>
-->(ans+3)^2 <Enter>
-->t=ans <Enter>
```

Como se aprecia desde el ejemplo, *"ans"* siempre almacena el último valor calculado. En el ejemplo 1-11 se realiza un segundo cálculo operando con el resultado anterior, debe quedar claro que luego de esta segunda instancia *"ans"* adopta el nuevo valor. Scilab también permite realizar varias operaciones en una misma línea y en este caso la asignación debe estar separada por una coma (",")

Ejemplo 1-12: Ejecutar el siguiente comando:

-->a1=5, a2=4.35, a3=0.0023 <Enter>
*-->a1*a3+a2, ans*2 <Enter>*

Se mencionó anteriormente, en la ejecución de una orden, si ésta se encuentra finalizada por un punto y coma, el efecto es que no se muestra el resultado de la ejecución en la ventana de Scilab, pero la operación se lleva adelante. (Esta forma de ejecutar órdenes se utiliza mucho en la confección de programas).

Ejemplo 1-13: Repetir el ejemplo anterior separando las instrucciones con punto y coma.

7.- Operadores y funciones

Scilab incluye numerosos operadores, entre ellos los aritméticos. Las cuatro operaciones básicas: suma, resta, multiplicación y división están representadas por: + - * / respectivamente y la potencia por: ^ ó **, algunos ya fueron utilizados en los ejemplos anteriores. También es posible utilizar paréntesis para agrupar distintas operaciones matemáticas. Las reglas con las que operan los símbolos precedentes son las habituales en matemática. A continuación, en el Cuadro 1.2, se listan los operadores elementales empleados en Scilab.

Las funciones que Scilab tiene incorporadas prácticamente cubren la mayor parte de las necesidades habituales de un usuario. También se tiene la posibilidad de definir funciones particulares para realizar algún cálculo específico. Esto se explica más adelante. Las funciones elementales disponibles se pueden observar en los cuadros 1.3 y 1.4.

La forma de utilizar una función ya definida en Scilab (como ingresar su sentencia), debe contener lo siguiente: el nombre específico de la función y el argumento o los argumentos.

Operadores matemáticos elementales en Scilab
+ suma
- resta
* multiplicación
/ división derecha, $x/y = xy^{-1}$
\ división, izquierda, $x\y = x^{-1} y$
^ potencia, $x^y=x^y$
** potencia (iqual que ^)
' transpuesta conjugada

Cuadro 1.2: *operaciones elementales en Scilab*

Por lo general también en la sentencia donde se usa una función debe estar definida la variable donde se almacenará el resultado. El argumento de la función debe estar entre paréntesis y a continuación del nombre de la función. Es posible que alguna función pueda tener más de un argumento y en este caso estarán separados por coma o punto y coma, dependiendo de la definición de la función.

acos	acosd	acosh	acoshm	acosm	acot	acotd	acoth
acsc	acscd	acsch	asec	asecd	asech	asin	asind
asinh	asinhm	asinm	atan	atand	atanh	atanhm	atanm
cos	cosd	cosh	coshm	cosm	cotd	cotg	coth
cothm	csc	cscd	csch	sec	secd	sech	sin
sinc	sind	sinh	sinhm	sinm	tan	tand	tanh
tanhm	tanm						

Cuadro 1.3: *funciones trigonométricas*

exp	expm	log	log10	log1p	log2	logm	max
maxi	min	mini	modulo	pmodulo	sign	signm	sqrt
sqrtm							

Cuadro 1.4: *otras funciones de Scilab*

La sintaxis general para el uso de cualquier función predefinida de Scilab puede consultarse mediante el comando *"help"*. Si se escribe en la consola de trabajo *help* seguido del nombre de la función, Scilab abre una ventana de ayuda donde podrá leerse una descripción de la función consultada. El uso de *"help"* también es válido para cualquier comando.

Si no se dispone del nombre específico de la función, se puede solicitar información de todas las funciones de un tema dado utilizando el comando *"apropos"*. Por ejemplo, se puede pedir: *-->apropos polynomial* y obtener todas las funciones relacionadas con los polinomios.

Ejemplo 1-14: Ejecutar los siguientes comandos en la consola de Scilab:

-->x=sin(3%pi/2) <Enter>*
-->help ceil <Enter>

Además de las funciones elementales existe una gran cantidad de funciones especiales que se estudiarán a lo largo del curso, muchas de las cuales tienen especial interés por sus aplicaciones en física e ingeniería.

7.1.- Constantes

Es habitual utilizar en matemática ó física constantes que tienen una importancia especial, como ser el número Pi, el número e (base del logaritmo natural), etc. Estas constantes (números) se encuentran predefinidos en Scilab y pueden ser utilizados directamente en cualquier cálculo. La denominación de *"constantes"* es debido a que tienen propiedades especiales. Estos números están protegidos de modo que no pueden ser modificados por un usuario de Scilab. Se distinguen con una nomenclatura especial, colocando el signo % delante de su nombre como se mencionó anteriormente.

Ejemplo 1-15: Ejecutar las siguientes operaciones:

-->%e <Enter>

-->%i <Enter>
-->%pi, %inf <Enter>

7.2.- Números Complejos

Scilab puede operar con números complejos en forma muy sencilla. Internamente el software tiene definida la variable, %i, que contiene o representa al número imaginario i ($\sqrt{-1}$). Un complejo cualquiera se expresa como un binomio, usando la parte real más la imaginaria multiplicada por %i. Por ejemplo: a = 5+7*%i . Las funciones de Scilab admiten números complejos como argumento y el resultado se expresa como un número complejo.

Ejemplo 1-16: Realizar las siguientes operaciones:

-->b = sqrt(-4) <Enter>
-->c=sqrt(5+%i) <Enter>
-->c^2 <Enter>

8.- Vectores y matrices

En matemáticas, una matriz es un arreglo bidimensional de números. Las matrices se usan generalmente para describir sistemas de ecuaciones lineales, sistemas de ecuaciones diferenciales o representar una aplicación lineal (dada una base). Las matrices se describen en el campo de la teoría de matrices. Se utilizan para múltiples aplicaciones y sirven, en particular, para representar los coeficientes de los sistemas de ecuaciones lineales o para representar las aplicaciones lineales; en este último caso las matrices desempeñan el mismo papel que los datos de un vector para las aplicaciones lineales. Pueden sumarse, multiplicarse y descomponerse de varias formas, lo que también las hace un concepto clave en el campo del álgebra lineal.

El origen de las matrices es muy antiguo. En el Cuadro 1.5 se resume un poco su historia.

Cronología del uso de matrices	
Año	***Acontecimiento***
200 AC	En China los matemáticos usan series de números
1848 DC	J. J. Sylvester introduce el término “matriz”
1858	Cayley publica “Memoria sobre la teoría de matrices”
1878	Frobenius demuestra resultados fundamentales en álgebra matricial
1925	Werner Heisemberg utiliza teoría matricial en Mecánica Cuántica

Cuadro 1.5: *algunas fechas importantes*

En Scilab las matrices juegan un rol central y en esta sección se presenta la forma de crear matrices, operar con ellas y cómo acceder a sus elementos. En la Figura 1.13 se muestra el esquema o representación de una matriz. Por ser un arreglo bidimensional los datos estarán distribuidos en un cierto número de filas o columnas. En la Figura 1.13 el

primer subíndice corresponde a la identificación de las filas y el segundo subíndice a las columnas.

Los vectores son un caso particular de matrices, donde el número de filas (o columnas) es igual a 1. Un vector fila será de 1 fila y n columnas y un vector columna contendrá 1 columna y m filas.

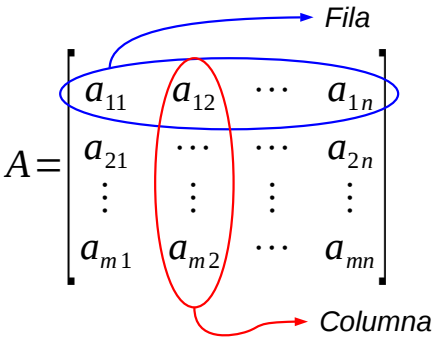

Figura 1.13: *Esquema de una matriz*

Las variables escalares no existen en Scilab y por lo tanto una variable escalar se representa por una matriz con 1 fila y 1 columna. Esta es la razón por la cual se analiza en esta sección el comportamiento de este nuevo tipo de variable, matrices de mxn, incluyendo también los vectores filas (1xn), vectores columnas (mx1) y escalares (1x1), siendo m el número de filas y n el número de columnas de la matriz.

8.1.- Definición de vectores y matrices en Scilab

Las matrices se definen en Scilab utilizando corchetes [;] para encerrar sus elementos que se separan por comas o por un espacio en blanco. Las diferentes filas de la matriz se separan por un punto y coma (;). También existen otras formas equivalentes de definir una matriz, lo que se ilustra mejor ejecutando y analizando los ejemplos que siguen.

Ejemplo 1-17: Ejecutar la siguiente instrucción:

```
-->C = [11,12,13;21,22,23] <Enter>
C =

   11.   12.   13.
   21.   22.   23.
```

Como se aprecia se ha generado una matriz de 2 filas y 3 columnas. Los elementos de cada fila fueron separados en columnas por coma y las filas separadas por punto y coma. Otra forma de generar la misma matriz es:

Ejemplo 1-18: Ejecute la siguiente instrucción:

```
-->C = [11 12 13;21 22 23] <Enter>
```

En este caso los elementos de una fila se separaron por espacios en blanco y las filas como antes por punto y coma. Las dos sintaxis de los ejemplos 1-17 y 1-18 son válidas.

Ejemplo 1-19: Ejecute la siguientes instrucciones:

```
-->C = [11 12 13  <Enter>
-->  21 22 23]  <Enter>
```

En este último ejemplo la matriz generada es la misma que en los ejemplos anteriores, y las columnas fueron separadas por un <*Enter*> sin cerrar el corchete. Según los ejemplos 1-17, 1-18 y 1-19 en todos los casos la expresión, para definir una matriz, debe comenzar y terminar con corchetes. La sintaxis del ejemplo 1-19, puede ser útil en situaciones en las cuales la matriz debe ingresarse a Scilab a partir de un archivo de datos porque simplifica la escritura de la matriz en el archivo de texto (blok de notas, editor texto linux, etc.).

Considerando la matriz C ya generada, un elemento que ocupa la fila i y la columna j se identifica como C(i,j) o sea C(1,3) es 13. Para esta matriz i puede solo tomar los valores 1 y 2 (C contiene dos filas) y j = 1, 2, 3 (C contiene tres columnas). Si se elije un determinado elemento de una matriz, luego este puede ser asignado a una variable o se utilizado para un determinado cálculo.

Otra forma de identificar un elemento puede ser empleando un solo índice y en este caso Scilab inicia la búsqueda en el primer elemento de la matriz (C(1,1) y recorre la matriz por columnas hasta alcanzar el valor indicado.

Ejemplo 1-20: Ejecutar las siguientes instrucciones y comprobar que se obtiene el mismo resultado.

```
-->C = [11,12,13;21,22,23;31,32,33]; // se redefine la matriz C <Enter>
-->b=C(1,1), d=C(2,3), suma= b+d  // se seleccionan dos elementos y se suman <Enter>
-->b1=C(1), d1=C(8), suma1=b1+d1 // los mismos elementos y se suman <Enter>
```

En el ejemplo 1-20 se sumaron dos elementos pertenecientes a la matriz C, el valor 11 y 23; luego se accedió a los mismos valores recorriendo la matriz por columnas. Este recorrido se muestra en la Figura 1.14.

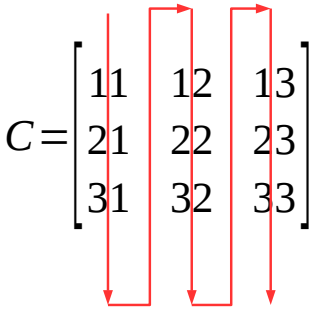

Figura 1.14: *Recorrido por columnas de una matriz*

Dada una matriz, su matriz transpuesta conjugada se obtiene utilizando el símbolo ' (una comilla) colocado después de la variable. La matriz transpuesta sin conjugar se obtiene con el símbolo .' (punto y comilla) colocados después de la variable.

Ejemplo 1-21: Ejecutar las siguientes instrucciones:

```
-->C1=[1+3*%i 12 13; 21 22  2-2*%i] <Enter>
-->C2=C' // transpuesta conjugada <Enter>
-->C3=C.' // transpuesta sin conjugar <Enter>
```

El producto de una constante por una matriz se expresa con el signo habitual para el producto (*). La constante multiplica cada elemento de la matriz.

Ejemplo 1-22: Verificar lo expresado en el párrafo anterior multiplicando la matriz C1 por una constante cualquiera.

El producto entre dos matrices se define mediante el operador para la multiplicación (*). La operación A*B indica el producto matricial entre las matrices A y B, de acuerdo a la definición del producto matricial (filas por columnas y sumando cada producto). También es posible realizar el producto elemento a elemento de dos matrices, para indicar esta operación en Scilab hay que anteponer un punto al signo asterisco, es decir: (.*). Es importante aclarar que las matrices deben tener las dimensiones correctas para efectuar estas multiplicaciones en caso contrario se puede obtener un resultado no deseado y en este caso Scilab muestra en la consola de salida un mensaje de error.

Ejemplo 1-23: Definir dos matrices de 3 filas y 3 columnas. Realizar el producto matricial y el producto elemento por elemento entre ambas matrices.

```
-->A1=[2,4,6;1,1,1;0,0,0] <Enter>
-->A2=[3,5,7;0,0,0;1,1,1] <Enter>
-->PA1=A1*A2 <Enter>
-->PA2=A1.*A2 <Enter>
```

Ejemplo 1-24: Ejecutar las instrucciones listadas a continuación y explicar los resultados obtenidos luego de ejecutar las diferentes sentencias.

```
-->B1=[2,4,6;1,1,1] // matriz dos filas y  tres columnas <Enter>
-->B2=[3,5,7;0,0,0]  <Enter>
-->PB1=B1*B2  <Enter>
-->PB2=B1.*B2  <Enter>
-->PB3=B1*B2'  <Enter>
```

Scilab cuenta con funciones que permiten generar matrices especiales en forma automática, por ejemplo: una matriz de m (filas) x n (columnas) cuyos elementos son números aleatorios entre 0 y 1, una matriz con una determinada diagonal, una matriz que solo contenga unos o ceros, etc. En el Cuadro 1.6 que figura a continuación se indican algunas de estas funciones.

Función	***Descripción***
diag	Genera una matriz de mxn con una dada diagonal o extrae la diagonal.
eye	Matriz de mxn con unos en la diagonal principal.
grand	Matriz de mxn con números aleatorios no gaussianos.
int	Matriz con la parte entera de los elementos de otra matriz.
linspace	Vector fila con elementos linealmente espaciados.
logspace	Vector fila con elementos logarítmicamente espaciados.

ones	Matriz de mxn con todos sus elementos iguales a 1.
rand	Matriz de mxn con elementos aleatorios gaussianos.
zeros	Matriz de mxn con todos sus elementos iguales a 0.

Cuadro 1.6: *matrices especiales en Scilab*

El argumento de cada función del Cuadro 1.6 debe indicar el número de filas y columnas que se desea. El siguiente ejemplo muestra su uso.

Ejemplo 1-25: Ejecutar las siguientes instrucciones y analizar los resultados. Describir lo que se obtiene para m3, m4, mt y d

```
-->m1=eye(2,1) // dos filas una columna con unos en la diagonal principal <Enter>
-->m2=3*ones(2,3) // dos filas una columna multiplicada por 3 <Enter>
-->m3=linspace(3,9,4), m4=zeros(1,4) <Enter>
-->mt=[m1,m2;m3;m4] <Enter>
-->d=diag(mt) <Enter>
```

8.2.- El generador de doble punto

Resulta muy útil en el entorno de Scilab la generación de vectores formados por una sucesión de valores igualmente separados. A esos efectos se dispone de una forma abreviada de generación de vectores que se escribe con el siguiente formato:

{Valor Inicial} : {incremento} : {Valor Final}

Esta instrucción genera un vector cuyo primer valor es *"Valor Inicial"* y los valores siguientes están igualmente espaciados en la cantidad *"incremento"* hasta llegar a *"Valor Final"*. Si se omite *"incremento"*, el programa considera por defecto que su valor es 1. Los dos puntos son parte de la sentencia.

Ejemplo 1-26: Generar un vector de 10 elementos separados por 0.5 y con un valor inicial igual a 5.

```
-->vec=5:0.5:9.5 <Enter>
```

Debido al uso del doble punto en la notación se le conoce con el nombre de generador doble punto. Este generador puede ser utilizado para formar vectores de las mas variadas características en sus componentes. En el ejemplo siguiente puede apreciarse esto:

Ejemplo 1-27: Generar dos vectores uno de ellos debe contener los 50 primeros números impares mayores que cero y el otro 20 primeros múltiplos de 7.

```
-->v1=1:2:100  // primeros 50 nros impares<Enter>
-->m=2 <Enter>
-->v2=(1:50)*m // solo números pares entre 1 y 100 <Enter>
-->v3=v2-1 //se obtienen los números impares
-->n=7; <Enter>
-->mult7=(1:20)*n // múltipos de siete<Enter>
```

En el ejemplo 1-27 se muestran 2 formas diferentes de obtener el mismo vector con números pares empleando el generador doble punto; las dos últimas sentencias se emplean para generar los múltiplos de 7. Esta notación doble punto puede ser utilizada para la extracción de submatrices como se verá más adelante. Recordar que una submatriz es una parte de una matriz ya definida, por ejemplo una fila o parte de una fila, una columna o parte de ella, un grupo de elementos.

Ejemplo 1-28: Extracción de la segunda y tercera columna de una matriz de 4x4 elementos y todos los elementos restantes al eliminar la primera fila y la primera columna de la matriz original.

```
-->A=[1,2,3,20;4,5,6,13;7,8,9,15;11,10,12,1] <Enter>
-->Asub1=A(:,2:3) <Enter>
-->Asub2=A(2:4,2:4) <Enter>
```

En la segunda instrucción del ejemplo, el operador doble punto, sin valores inicial ni final, indica todas las filas (o columnas) de la matriz. De esta forma se ha obtenido una submatriz de la matriz A previamente definida, cada submatriz esta asociada a las variables "*Asub1"* y "*Asub2"* respectivamente.

9.- Tiras de caracteres

Una tira de caracteres (string) es básicamente un texto que puede ser manipulado con diferentes comandos de Scilab. Para que el texto sea reconocido como tal, debe ser escrito entre comillas (simples o dobles) o sea un string puede ser encerrado tanto por comillas dobles (" ") como por comillas simples (' ') y Scilab lo interpreta de la misma forma. En esta sección se verán algunas funciones y operaciones que pueden ser de gran utilidad cuando se trabaja con archivos de texto.

Dos o más cadenas de caracteres pueden unirse en sucesión mediante la operación de concatenación, que se realiza directamente empleando el símbolo + (suma). Es posible definir matrices de caracteres y estas se construyen como matrices ordinarias, es decir utilizando corchetes. Un uso muy importante de las matrices de caracteres es la manipulación y creación de funciones. En el siguiente ejemplo se ilustra la creación de una matriz de caracteres.

Ejemplo 1-29: Ejecutar los siguientes comandos:

```
-->t1="El resultado es  " <Enter>
-->t2="positivo" <Enter>
-->t3=t1+t2 <Enter>
-->t4='= +8' <Enter>
-->T=[t1, t2, t4] <Enter>
```

Lo que se puede observar en el ejemplo anterior es que un número en Scilab puede ser considerado como variable numérica o como una cadena de caracteres (variable string) si se lo ingresa encerrado entre comillas como sucede para el caso de la variable t4 donde el signo igual, un espacio y el signo mas (+) que precede al 8 se consideran como string porque están encerrados entre comillas simples, por lo tanto t4 en el ejemplo 1-29 es un string. Además de acuerdo al mismo ejemplo, la operación entre variables de caracteres es la concatenación o sea (+).

Sería lógico preguntarse si se puede convertir un número ingresado como variable de texto o sea encerrado entre comillas a variable numérica. Para Scilab esto es posible y se utiliza la función *"evstr"*.

Para analizar como se utiliza esta función ejecutar y analizar el ejemplo 1-30.

Ejemplo 1-30: Ejecutar los siguientes comandos:

```
-->a='5'; b='2'; c='1-2*%i' ; //son todas variables string <Enter>
-->d=a+b <Enter>
-->a*b <Enter>
-->evstr(a)+evstr(b) <Enter>
-->evstr(a)*evstr(b) <Enter>
-->evstr('5')+4*evstr(c) <Enter>
```

El ejemplo 1-30 muestra que cuando se ejecuta a*b, Scilab no evalúa las operaciones matemáticas si la variable está definida como texto. Para obtener un resultado numérico hay que convertir la variable de texto (string) a variable numérica utilizando la función *"evstr"*. Otra función importante en el manejo de variables de texto es la función *"execstr" (ejecutar string)*, esta función ejecuta comandos ingresados como tira de caracteres como se aprecia en el ejemplo siguiente:

Ejemplo 1-31: Ejecutar los siguientes comandos:

```
-->execstr(['a=1','b=2', 's=a+b'])<Enter>
-->a, b, s <Enter>
```

En el ejemplo 1-31 se observa que después de la primera sentencia Scilab no muestra ninguna salida, pero internamente las operaciones han sido realizadas, como se ve al ejecutar la segunda sentencia.

Ejemplo 1-32: ejecutar la siguiente sentencia

```
-->execstr(['s=2'   'm=3'  'r=sqrt(s)'  'q=m+s'])
```

Scilab cuenta con varias funciones para el manejo de variables de string. El Cuadro 1.7 que se muestra da ejemplos de algunas de ellas.

Función	***Descripción***
ascii	Convierte un carácter a su valor ascii y viceversa.
execstr	Envía un string al intérprete de Scilab.
grep	Busca la ocurrencia de un string en una matriz de string.
part	Extrae un substring de un string.
strindex	Encuentra ocurrencias de strings en un string.
string	Convierte datos a string.
stripblanks	Remueve los espacios en blanco al principio y al final de un string.
strsubst	Sustituye un string en una matriz de string.
strcat	Concatena matrices de string.
length	Da la longitud de una tira de caracteres.

Cuadro 1.7: *funciones para el manejo de variables de string*

Ejemplo 1-33: Ejecutar los siguientes comandos. Explicar que realiza cada uno.

-->A=rand(2,8,'n') // matriz de 2x8 de números aleatorios con distribución normal <Enter>
-->A=sign(A) <Enter>
-->A=string(A) <Enter>
-->A=strsubst(A,'1','+') <Enter>
-->A=strsubst(A,'-+','-') <Enter>

Ejemplo 1-34: Idem al ejemplo anterior para los comandos que se muestran

-->name='x', n=3, val=[45 67 34] <Enter>
-->str=name+string(1:n)+'=val('+string(1:n)+') <Enter>
-->execstr(str) <Enter>
-->[x1,x2,x3] <Enter>

10.- Polinomios en Scilab

Un polinomio es una expresión algebraica de la forma:

$$p(x) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \cdots + a_1 x + a_0$$

donde n es el grado del polinomio que por lo general es un número natural, los diferentes factores a_n son constantes numéricas.

Scilab reconoce polinomios como un dato particular, inclusive pueden generarse matrices de polinomios, por lo tanto todas las operaciones posibles para matrices de números son permitidas para matrices polinómicas. Un polinomio se define con la función “poly” y puede ser definido teniendo en cuenta sus raíces o sus coeficientes.

La sentencia para generar polinomios es:

p = poly(a, vname, ["flag"])

donde:

- *“a”*: es una matriz o un número real; en el caso de ser una matriz, el polinomio que se genera es el *“polinomio característico”* de la matriz o sea det (A – λI), siendo I la matriz identidad.

- *“vname”* : es un string y determina el nombre simbólico de la variable del polinomio, puede tener como máximo cuatro caracteres.

- [*"flag"*]: es un string que puede ser *“roots”* o *“coeff”* o sea por raices o coeficientes, se toma por defecto *“roots”* o sea si no se pone de forma explícita Scilab realiza la operación como roots. También se permite escribir abreviadamente *“r”* o *“c”* para *“roots”* o *“coeff”* respectivamente

- *poly(v,"x",["roots"])*: es un polinomio cuyas raíces son los elementos de v y *“x”* es la variable. (En este caso roots y poly son funciones inversas como se vera más adelante). Notar que raíces infinitas generan coeficientes cero en las variables de alto grado.

- *poly(v,"x","coeff")*: genera un polinomio con variable *"x"* y los coeficientes son los elementos del vector v (v(1) es el término constante del polinomio). (Aqui poly y coeff son funciones inversas).

En la definición de poly aparece un término entre corchetes [ ] y como a lo largo del curso se mostraran otras definiciones de funciones es importante aclarar algo al respecto. Los corchetes en todas las definiciones simbolizan que se puede o no ingresar algún valor (valores optativos). Si no se ingresa ningún valor, Scilab toma el valor por defecto que tendrá la definición de la función.

El uso de poly se aprecia en los siguientes ejemplos.

Ejemplo 1-35: ejecutar los siguientes comandos y analizar los resultados

```
-->p=poly([1,3,4],'s') //polinomio definido por sus raíces  <Enter>
-->q=poly([1,3,4],'s','c') // polinomio definido por sus coeficientes <Enter>
```

Como se mencionó, para definir un polinomio empleando los coeficientes, hay que tener en cuenta que el primer elemento del vector de coeficientes se asigna al término independiente, el segundo al término lineal, el tercero al cuadrático y así sucesivamente. Si el polinomio no contiene alguna potencia en particular su coeficiente debe ser 0.

En el ejemplo *'s'* representa el argumento de poly y especifica el carácter que será usado como variable del polinomio.

La función *“varn”* identifica la variable utilizada en un polinomio y la función *“degree”* devuelve el grado del polinomio. Para obtener un vector con los coeficientes de un polinomio se puede utilizar la función *“coeff ”* y son listados en orden creciente de la potencia de la variable del polinomio.

Para mayor información sobre la función *“poly”* y todas las funciones relacionadas se aconseja consultar la ayuda de Scilab.

Ejemplo 1-36: ejecutar los siguientes comandos analizando el resultado

```
-->x=poly(0,'x') //se define la variable del polinomio <Enter>
-->p=3.5*x^2+2*x-5 // se define un polinomio <Enter>
-->c=coeff(p) <Enter>
-->r=roots(p)  <Enter>
-->A=rand(2,2); // genera una matriz de 2 x 2 de números aleatorios <Enter>
-->poly(A,"x")
```

Ejemplo 1-37: Ejecutar y analizar los resultados

```
-->s=poly(0,'s'); <Enter>
-->A=[s+1 s-2;(s+1)^2 s^2/2] // se define una matriz de polinomios <Enter>
-->det(A) <Enter>
-->horner(A,3) // evalúa los polinomios de la matriz A en s=3 <Enter>
-->h=(1+2*%s)/poly(1:4,'s','c') <Enter>
```

En la última sentencia el polinomio del denominador se genera con coeficientes que van de 1 a 4, siendo el 1 el término independiente. En el siguiente ejemplo se vuelve a utilizar la función *"roots"* para obtener las raíces de una expresión polinómica.

Ejemplo 1-38: uso de la función roots

```
-->v=[4,-2,1,7]; <Enter>
-->k=poly(v,'y','c')  // se define el polinomio k con los coeficientes propuestos en v <Enter>
-->q=poly([1,1,-1],'z','r') <Enter>
-->w=[2,3,-1] <Enter>
-->r=poly(w,'n') // El polinomio r se genera con las raíces w <Enter>
-->roots(k) // calcula las raíces del polinomio k <Enter>
-->roots(q) // calcula las raíces del polinomio q <Enter>
-->roots(r) // calculas las raíces del polinomio r <Enter>
```

Cuando se calcula roots(q) el resultado debe ser [1,1,-1] y cuando se calcula roots(r) el resultado debe ser el vector w, porque el polinomio fue construido a partir de las raíces definidas en w.

10.1.- Resolución de sistemas de ecuaciones lineales

Un sistema de n ecuaciones lineales y m variables puede ser escrito explícitamente de la siguiente forma:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1m}x_m &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2m}x_m &= b_2 \\ &\vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nm}x_m &= b_n \end{aligned}$$

En forma matricial estas ecuaciones pueden ser escritas como:

$$\boldsymbol{A}\vec{\boldsymbol{x}} = \vec{\boldsymbol{b}} \qquad \rightarrow \qquad \boldsymbol{A}_{n,m}\boldsymbol{x}_{m,1} = \boldsymbol{b}_{n,1}$$

donde el primer sub-índice (n) representa el número de filas y el segundo (m) el número de columnas.

En forma explícita la ecuación anterior se expresa como:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}$$

El vector x es el vector de incógnitas, la matriz A es la matriz de coeficientes de valores conocidos y b es el vector columna de términos independientes, también de valores conocidos.

El Cuadro 1.8 resume las principales características de un sistema de ecuaciones

COMPATIBLES *(con solución)*	DETERMINADOS *(una solución)*
	INDETERMINADOS *(infinitas soluciones)*
INCOMPATIBLES *(sin solución)*	
Homogéneos	*Todos los términos independientes son nulos*
No Homogéneos	*No todos los términos independientes son nulos*

Cuadro 1.8: *Sistema de ecuaciones*

Se denomina matriz ampliada de dimensión n filas y m+1 columnas a la matriz que se obtiene al añadir a la matriz de coeficientes la columna de los términos independientes, por lo general se denota como (**A**|**b**). Para saber si un sistema de n ecuaciones y m incógnitas tiene o no solución (si es compatible) y cuántas soluciones tiene (si es determinado o indeterminado) se aplica el teorema de Fouché – Frobenius. Este teorema establece la condición necesaria y suficiente para que el sistema tenga solución a partir de considerar el rango de la matriz de coeficientes y el de la matriz ampliada.

El rango de una matriz se entiende como la mayor submatriz (el menor) con determinante diferente de cero o sea no nulo y se denota como Rg (**A**) = rango de la matriz **A**. Entonces un sistema cualquiera de matriz A y matriz ampliada (**A**|**b**) tiene solución (es compatible) si y solo si Rg (**A**) = Rg (**A**|**b**). Por tanto si los dos rangos son distintos el sistema no tiene solucióon (S.I.). Además, si dicho rango coincide con el número de incógnitas del sistema, la solución es única (S.C.D.) y si dicho rango es menor que el número de incógnitas, hay infinitas soluciones (S.C.I.).

Es importante darse cuenta de que Rg (**A**) ≤ Rg (**A**|**b**), puesto que la matriz de coeficientes forma parte de la ampliada, es decir, la matriz A no puede tener rango mayor que la ampliada.

Para resolver un sistema de estas características con Scilab, se puede proceder de diferentes maneras, una utilizando la función *linsolve* y la otra es utilizando la división a la izquierda. A continuación se explica cada uno de los casos.

10.1.1.- Utilizando la función “linsolve”

Esta función más que solucionar la ecuación matricial expresada anteriormente (A x = b) soluciona la ecuación A x + c = 0 donde c = - b.

La sintaxis general para esta función es:

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

donde:

- *“A”*: es la matriz de coeficientes que caracteriza el sistema que se quiere resolver, de n filas por m columnas.

- *“B”*: es el vector columna de términos independientes del sistema.

- *“X0”*: es una solución particular (si existe) y *“kerA”* es el espacio nulo de A (nullspace A). Cualquier x = x0 + kerA*w con w arbitrario satisface A*x+b = 0.

“linsolve” calcula todas las soluciones de A*x+b = 0. Dependiendo del número de incógnitas y ecuaciones *“linsolve”* puede producir diferentes resultados que se resumen a continuación y se denotan como diferentes casos:

- *Caso 1: Sistema con igual número de ecuaciones que de incógnitas. Existe una única solución.*

Si el número de incógnitas es igual al número de ecuaciones la solución puede ser única en cuyo caso el espacio nulo (kerA) que retorna linsolve es vacío. Considerando el siguiente ejemplo se muestra lo anterior

Ejemplo 1-39: El sistema considera tres ecuaciones y tres incógnitas

```
-->A=[2,3,-5;1,-3,8;2,-2,4], c=[-13;13;6] <Enter>
-->[x0, nulo]=linsolve(A,c) <Enter>
```

En este caso Scilab retorna nulo=[ ] o sea un conjunto vacío (que es kerA), indicando que la solución es única. Para verificar esto ejecutar el siguiente ejemplo.

Ejemplo 1-40:

```
-->A*x0+c <Enter>
ans  =

 10^(-14) *

  - 0.1776357
    0.
    0.
```

Como se observa la solución no es idénticamente igual a cero pero es un número muy pequeño (del orden de 10^{-14}) lo que indica que x0 es una solución muy aproximada del sistema. Otro aspecto a tener en cuenta es que la variable que indica el espacio nulo, puede tener cualquier nombre.

- *Caso 2: Sistema con igual número de ecuaciones que de incógnitas y no existe una solución única.*

Ejemplo 1-41:

```
-->A=[2,3,-5;2,3,-5;2,-2,4], b=[-13;13;6] <Enter>
-->[x0, nulo]=linsolve(A,b) <Enter>
```

Scilab en este caso muestra lo siguiente:

WARNING: Conflicting linear constraints!

nulo =

[]

x0 =

[]

indicando que no hay solución, esto es lógico porque en el ejemplo las dos primeras ecuaciones representan planos paralelos en el sistema cartesiano. Esto se puede observar mejor si se escribe el sistema de ecuaciones:

$$\begin{aligned} 2x+3y-5z &= -13 \\ 2x+3y-5z &= 13 \\ 2x-2y+4z &= 6 \end{aligned}$$

las dos primeras ecuaciones solo difieren en el término independiente, por lo tanto los planos son paralelos y ese es el motivo que hace que no exista una única solución.

- *<u>Caso 3:</u> Sistema con mayor número de incógnitas que de ecuaciones*

Un sistema de ecuaciones de este tipo no está unívocamente determinado y generalmente se lo denomina sobre-determinado. Sería el caso de la intersección de dos planos (no paralelos) en el espacio cartesiano. La intersección de dos planos determina una línea recta, entonces existe más de un punto que es solución del sistema y por lo tanto no esta unívocamente determinado. (Figura 1.15)

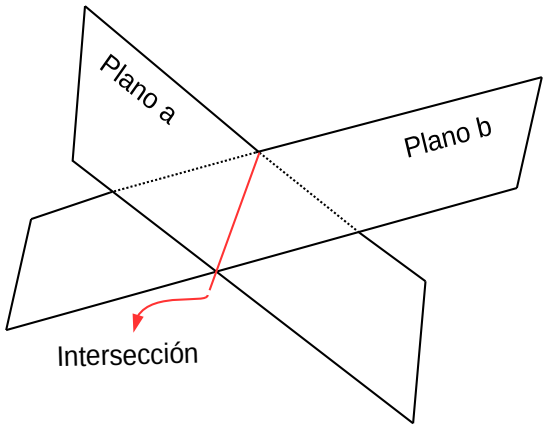

Figura 1.15: *linea generada por la intersección de dos planos cualquiera*

Supongamos el sistema lineal de ecuaciones:

$$\begin{aligned} 2x+3y-5z+10 &= 0 \\ x-3y+8z-85 &= 0 \end{aligned} \quad \rightarrow \quad A=\begin{bmatrix} 2 & 3 & -5 \\ 1 & -3 & 8 \end{bmatrix} \quad x=\begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad c=\begin{bmatrix} 10 \\ -85 \end{bmatrix}$$

Queda claro del análisis de las ecuaciones anteriores que existen más incógnitas que ecuaciones y que cada ecuación representa un plano en el espacio cartesiano, por lo tanto la intersección de dos planos produce una línea recta, lo que significa que cada punto de la recta es solución del sistema y es por esto que el sistema no esta

unívocamente determinado. El ejemplo que se da continuación muestra lo que sucede cuando se aplica *"linsolve"* a un sistema de esta característica.

Ejemplo 1-42: Ejecutar
-->A=[2,3,-5;1,-3,8] <Enter>
-->c=[10;-85] <Enter>
-->[x0,nsa]=linsolve(A,c) <Enter>

En este caso Scilab devuelve valores no nulos para *"x0"* y *"nsa"* que es el espacio nulo de A. Por lo tanto x0 es una solución particular del sistema. Como se mencionó anteriormente no existe una solución única para el sistema y cualquier valor de x obtenido mediante: x1=x0+ nsa*t; con t cualquier valor real es solución del sistema.

Ejemplo 1-43: De acuerdo a lo calculado en el ejemplo anterior, ejecutar:

*-->x1=x0+nsa*10 //se supone t=10 <Enter>*
*-->A*x1+c // para verificar que es solución del sistema <Enter>*

El resultado del último cálculo es del orden de 10^{-12}, que es prácticamente 0, por lo tanto es solución del sistema de ecuaciones. Si se realiza la operación anterior con otro valor para t se comprobará lo mismo.

- *Caso 4: Sistema con más ecuaciones que incógnitas*

Un sistema de este tipo no tiene una solución única. Suponiendo el siguiente sistema de ecuaciones:

$$x_1+3x_2-15=0$$
$$2x_1-5x_2-5=0$$
$$-x_1+x_2-22=0$$

cada una de las ecuaciones anteriores representa una línea recta en el sistema de coordenadas x_1 ,x_2 . Salvo que dos de las tres ecuaciones sean iguales (la misma recta) las tres rectas tendrán tres puntos de intersección diferentes y es por esta razón que la solución no es única. Cuando se pretende resolver un sistema de este tipo con Scilab, se obtiene un mensaje de error.

Ejemplo 1-44: Ingresar la matriz A de coeficientes correspondiente al sistema de ecuaciones que mostrado anteriormente y el vector c de términos independientes. La matriz A es una matriz de 3 filas y 2 columnas y c es un vector columna de 3 elementos. Cuando se ejecuta:

-->[x0,nsa]=linsolve(A,c) <Enter>

Scilab entrega el siguiente mensaje:

WARNING: Conflicting linear constraints!
nsa =

[]
x0 =

[]

Indicando que el sistema no tiene solución.

10.1.2.- Utilizando “división a la izquierda (\)”

Si se tiene el sistema de ecuaciones A*x=b, donde A es una matriz de coeficientes, cuadrada (igual número de filas que de columnas), se puede solucionar el sistema empleando directamente el operador (\) denominado división a la izquierda (left-division operator) para diferenciarlo del operador (/) que sería división a la derecha y que tiene otra aplicación.

En este caso para Scilab se cumple que: x = A\b. Este operador “división a la izquierda” representa la división de vectores y matrices de la misma forma que el operador “división a la derecha” representa la división de números.

Considerando el sistema lineal de ecuaciones:

$$\begin{aligned} 2x_1 - 5x_2 + x_3 &= 18 \\ 2x_1 + 2x_2 - 3x_3 &= 23 \\ 5x_2 - 2x_3 &= -51 \end{aligned}$$

En Scilab se soluciona ejecutando lo que indica el ejemplo siguiente

<u>Ejemplo 1-45:</u> Ejecutar

```
-->A=[2,-5,1;2,2,-3;0,5,-2]; b=[18;23;-51]; <Enter>
-->x=A\b <Enter>
```

Para comprobar que la solución es correcta se debe realizar la operación A*x y se debería obtener el vector b ingresado.

10.1.3.- Utilizando la matriz inversa

La forma tradicional de escribir la solución de un sistema de ecuaciones lineales es de la forma A.x = b entonces:

$$x = A^{-1}.b$$

la que se obtiene al multiplicar ambos miembros de la primera ecuación por la inversa de la matriz A, considerando además que A^{-1} .A = I, siendo I la matriz identidad, la que se caracteriza por tener unos en su diagonal principal y cero en los demás términos extradiagonales.

Para Scilab el calculo de la matriz inversa se realiza rápidamente ejecutando la función que se muestra en el ejemplo siguiente

<u>Ejemplo 1-46:</u> Considerando la matriz A y el vector b ingresados en el ejemplo anterior, calcular:

```
-->invA=inv(A) <Enter>
-->invA*A  //se comprueba que se obtiene la matriz identidad
-->x=inv(A)*b  // se obtiene el mismo resultado que en el Ejemplo 45 <Enter>
```

Ejemplo 1-47: Comprobar que otra forma posible de obtener la solución sería:

*-->x=A^(-1)*b <Enter>*

11.- Bibliografía

Campbell S. L., Chancelier J. P., Nikoukhah R. – (2006) – “Modeling and Simulation in Scilab/Scicos” – ISBN-10: 0-387-27802-8 , ISBN-13: 978-0387278025 – Springer – USA.

Sandeep Nagar – (2016) - “Introduction to Scilab for Scientists and Engineers” - Present book is presented under Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International license - www.bookmuft.com

Scilab Group INRIA Meta2 Project/ENPC Cergrene - “Scilab Reference Manual”

Urroz E. G. - (2002) - “Numerical and Statistical Methods with SCILAB for Science and Engineering”

12.- Links

https://www.scilab.org/

http://www.openeering.com/sites/default/files/Scilab_calculator.pdf

http://www.openeering.com/sites/default/files/First_Steps_0.pdf