PRÁCTICA 1. INTRODUCCIÓN A SCILAB I

OBJETIVO. El objetivo de esta práctica es aclarar la dinámica de trabajo y de evaluación de esta materia así como una introducción al ambiente de trabajo de Scilab, el manejo de variables y constantes y los comandos básicos para trabajar con matrices y vectores.

DINÁMICA DE TRABAJO DE ESTA MATERIA:

1) El alumno deberá haber leído el instructivo de cada práctica antes de cada sesión.
2) Para asegurarse de lo anterior, el profesor comenzará la sesión haciendo preguntas orales al azar sobre los temas que trata el instructivo.
3) Al final de cada sesión, el profesor aplicará una evaluación individual (de 15 a 20 minutos) por escrito, de la cual obtendrá una calificación de la sesión.
4) Se dejará de tarea elaborar un reporte o resolver algún problema o problemas, lo cual se deberá entregar en la siguiente sesión.
5) Se aplicarán tres exámenes parciales, los cuales podrán consistir en proyectos a realizar o en exámenes por escrito.
6) La calificación final de la materia considerará: asistencia a las sesiones, participaciones orales en clase, evaluaciones escritas de cada sesión, exámenes parciales y reportes.

INTRODUCCIÓN.

Scilab (Scientific Laboratory) es un paquete de herramientas de software libre, inspirado en **Matlab** y desarrollado por INRIA (Institut National de Recherche en Informatique et en Automatique) y la ENPC (École Nationale des Ponts et Chaussées) desde 1990. En 2003 fue creado el Scilab Consortium dentro de la fundación Digiteo y actualmente Scilab es desarrollado por Scilab Enterprises desde julio 2012.

Scilab fue diseñado para hacer cálculos numéricos, especialmente cálculos basados en matrices y álgebra lineal, análisis y visualización de datos y para facilitar la escritura de nuevos programas dentro de este tipo de objetivos. **Scilab** también ofrece la posibilidad de hacer algunos cálculos simbólicos como derivadas de funciones polinomiales y racionales. Posee cientos de funciones matemáticas y la posibilidad de integrar programas en los lenguajes más usados (Fortran, Java, C y C++).

La ventaja de **Scilab** es que combina funciones matemáticas y gráficas con un poderoso lenguaje interpretado de alto nivel. **Scilab** viene con numerosas herramientas: gráficos 2D y 3D, animación, álgebra lineal, matrices dispersas, polinomios y funciones racionales, Simulación: programas de resolución de sistemas de ecuaciones diferenciales (explícitas e implícitas).

Scilab contiene a **SciCos (o Xcos)**, el cual es un simulador similar a **Simulink**, basado en diagramas de bloques de sistemas dinámicos, el cual permite la simulación de sistemas lineales, no lineales e híbridos.

Scilab es una herramienta poderosa en diversos campos tales como: control clásico, control robusto, optimización diferenciable y no diferenciable, Tratamiento de señales, Grafos y redes, computación paralela empleando PVM (parallel virtual machine), Estadística, Creación de GUIs (Graphic User Interfaces) , Interfaz con

el cálculo simbólico (Maple, MuPAD), Interfaz con TCL/TK. Además el uusario puede agregar numerosas herramientas o toolboxes.

En el pasado **Scilab** podía ser utilizado en el análisis de sistemas, pero no podía interactuar con el exterior. Hoy en día se pueden construir interfaces para que desde Scilab se pueda manejar dispositivos externos, con la posibilidad de conectar una tarjeta de adquisición de datos a Scilab y de esta forma realizar el control de una planta en tiempo real.

☞ Como **Scilab** es un lenguaje interpretado, algunas secuencias son más lentas que en lenguajes compilados, especialmente las que involucran ciclos escritos por el usuario. Por esta razón una primera recomendación es evitar al máximo esta situación y escribir los ciclos siempre que sea posible en forma vectorizada o recurrir a funciones incorporadas en Scilab que realicen lo que se quiere hacer con el ciclo.

Desde su creación hasta la fecha se han desarrollado una gran cantidad de versiones de Scilab. Este manual de prácticas está basado en la versión 5.5.0 para Windows.

INICIANDO SCILAB

Scilab para Windows puede iniciarse seleccionándolo desde el icono de Windows. Al ejecutarse Scilab crea una ventana como la mostrada en la figura 1.1. La ventana principal es la denominada **Consola** o Ventana de Comandos que muestra un mensaje inicial y termina con un apuntador consistente en los signos "-->" seguidos del cursor parpadeante indicando que Scilab está listo para recibir comandos en esa línea (Línea de comandos).

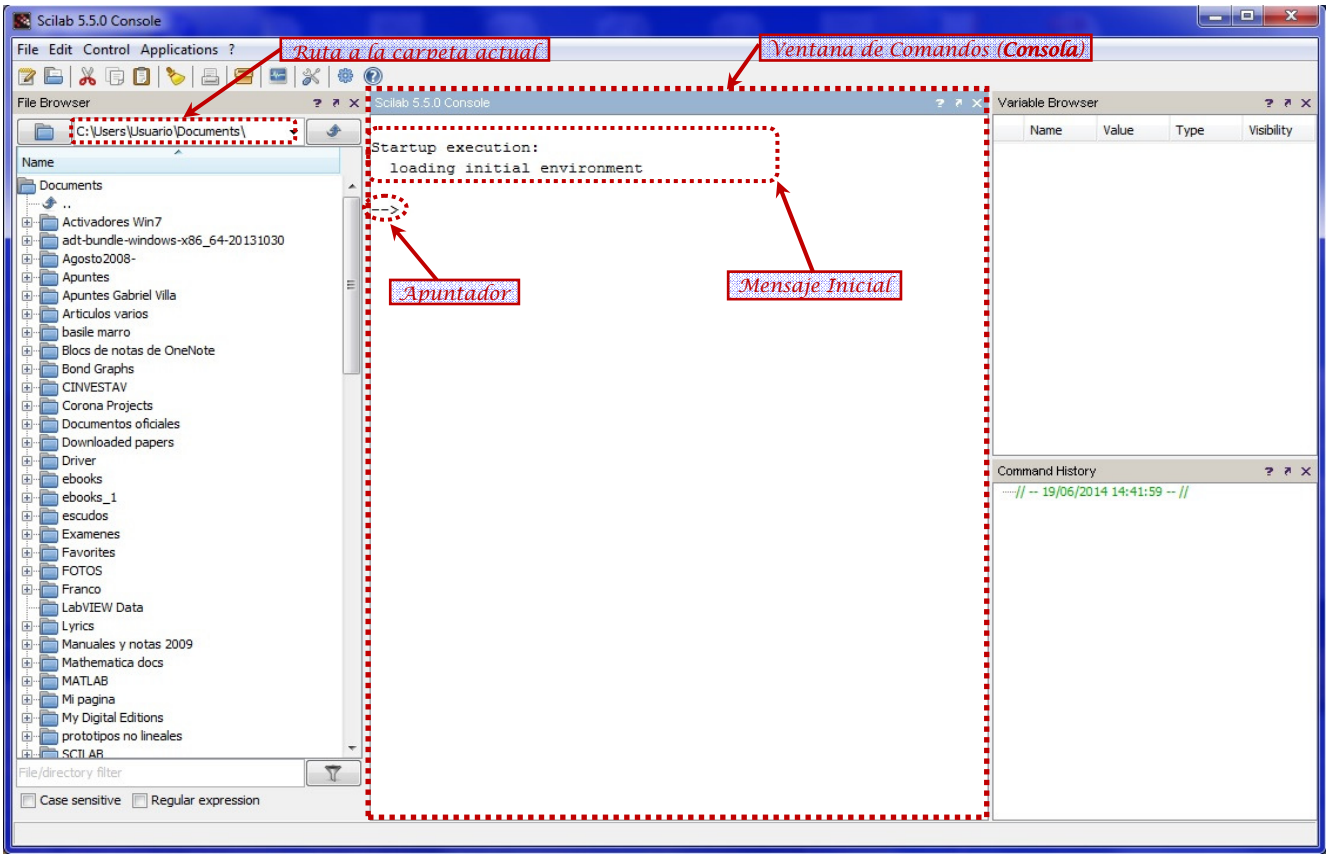

Figura 1.1.- Ventana principal y consola de Scilab

Casi todas las actividades de las prácticas se realizarán en la ventana de comandos, sin embargo, en la figura 1.1 se muestran también otras ventanas que aparecen por default la primera vez que se ejecuta Scilab:

- **File Browser.-** Explorador de archivos, visualiza los archivos guardados en la carpeta actual y permite navegar por las diferentes carptetas del sistema.
- **Variable Browser.-** Explorador de las variables que se van creando o las que se van importando.
- **Command History.-** Visualiza y permite re-ejecutar las instrucciones introducidas en la línea de comandos.

USO DE SCILAB COMO UNA CALCULADORA.

La manera más sencilla de utilizar Scilab es introduciendo en la línea de comandos las operaciones que se desea realizar y el resultado se obtiene inmediatamente al terminar el comando con la tecla [enter] en la variable `ans`.

Ejemplo:

```
(4+8)/2
ans =
     6.
```

Ejemplo: lo mismo, pero usando una variable

```
x=4+8;
x/2
ans =
     6.
```

OPERACIONES ARITMÉTICAS BÁSICAS:

De la misma forma que una calculadora, Scilab ofrece las siguientes operaciones aritméticas básicas:

Operación	Símbolo	Ejemplo	Precedencia
Suma	+	5 + 3	3
Resta	-	23 - 12	3
multiplicación	*	20.5 *1.5	2
división	/ o \	56/8 = 8\56	2
Potencia	^	5^2	1

PRECEDENCIA DE LAS OPERACIONES BÁSICAS:

Las expresiones se evalúan de izquierda a derecha, con la operación de potencia teniendo la orden de precedencia más alta, seguida por la multiplicación y división que tienen ambas igual precedencia y seguidas finalmente por suma y resta que tienen ambas igual precedencia. Se pueden emplear paréntesis para alterar esta usual ordenación.

Ejemplo: Para evaluar la expresión $\left(\frac{8}{7}\right)\times\left(\frac{5}{4}\right)=\frac{8\times5}{7\times4}$ sin usar paréntesis se puede realizar como sigue:

```
8*5/7/4
ans =
    1.4285714
```

o también:

```
8/7*5/4
ans =
```

```
    1.4285714
```

FORMATO DE VISUALIZACIÓN DE NÚMEROS Y SU REPRESENTACIÓN INTERNA.

La visualización de las cantidades numéricas se pueden obtener en varios formatos. Los formatos de visualización de números no cambian la representación interna de un número, solamente su visualización. El formato por default es `format('v',10)`. Los formatos de visualización disponibles son:

Comando	Visualización de 1/6	Comentario
`format('v',n)`	`0.166...67`	Despliega n-2 dígitos (dos caracteres son para el punto y el signo). Por default n=10.
`Format('e',n)`	`1.66...67D-01`	Despliega n-6 dígitos (6 caracteres son para el punto, el signo, la letra D, el signo del exponente y dos dígitos de exponente). Por default n=10.

CONSTANTES PREDEFINIDAS EN SCILAB.

Scilab proporciona valores predefinidos de algunas constantes que son usuales en los cálculos matemáticos. En la siguiente tabla se muestran las principales.

Variable	Valor predefinido	Comentario
`%i`	$\sqrt{-1}$	Unidad imaginaria (permite definir números complejos)
`%pi`	$\pi = 4\tan^{-1}(1)$	180^o en radianes
`%e`	`exp(1)`	Base de los logaritmos naturales
`%f, %F`	Valor lógico **Falso**	
`%t, %T`	Valor lógico **Verdadero**	
`%eps`	2^{-52}	Número positivo más pequeño sumable con un entero en Scilab
`%inf`	∞	Resultado Infinito, por ejemplo: 1/0
`%nan`	"No es un Número"	Operación no definida, por ejemplo: 0/0, inf-inf

Las constantes de la tabla anterior no pueden ser redefinidas.

Ejemplo: La siguiente expresión calcula el área de un círculo de radio 1.5:

```
%pi*1.5^2
ans=
   7.0685835
```

Ejemplo: se pueden hacer operaciones aritméticas con números complejos, por ejemplo

```
(1+%i)*(2-%i)
ans =
  3. + i
```

Ejemplo: Los operadores de comparación producen resultados lógicos

```
2 > 5
ans =
  F
```

Ejemplo: Los números más pequeños que %eps se desprecian al sumarse con un entero:

```
format('v',20)
1+%eps
```

```
ans =
  1.0000000000000022
1+%eps/2
ans =
  1.
```

AYUDA EN LÍNEA.

Scilab tiene un comando llamado `help` muy bien documentado. Escribiendo `help` en la ventana de comandos se despliega en una nueva ventana el **Navegador de Ayuda** como se muestra en la figura 1.2. El cual ayuda al usuario a navegar por la enorme lista de tópicos sobre Scilab.

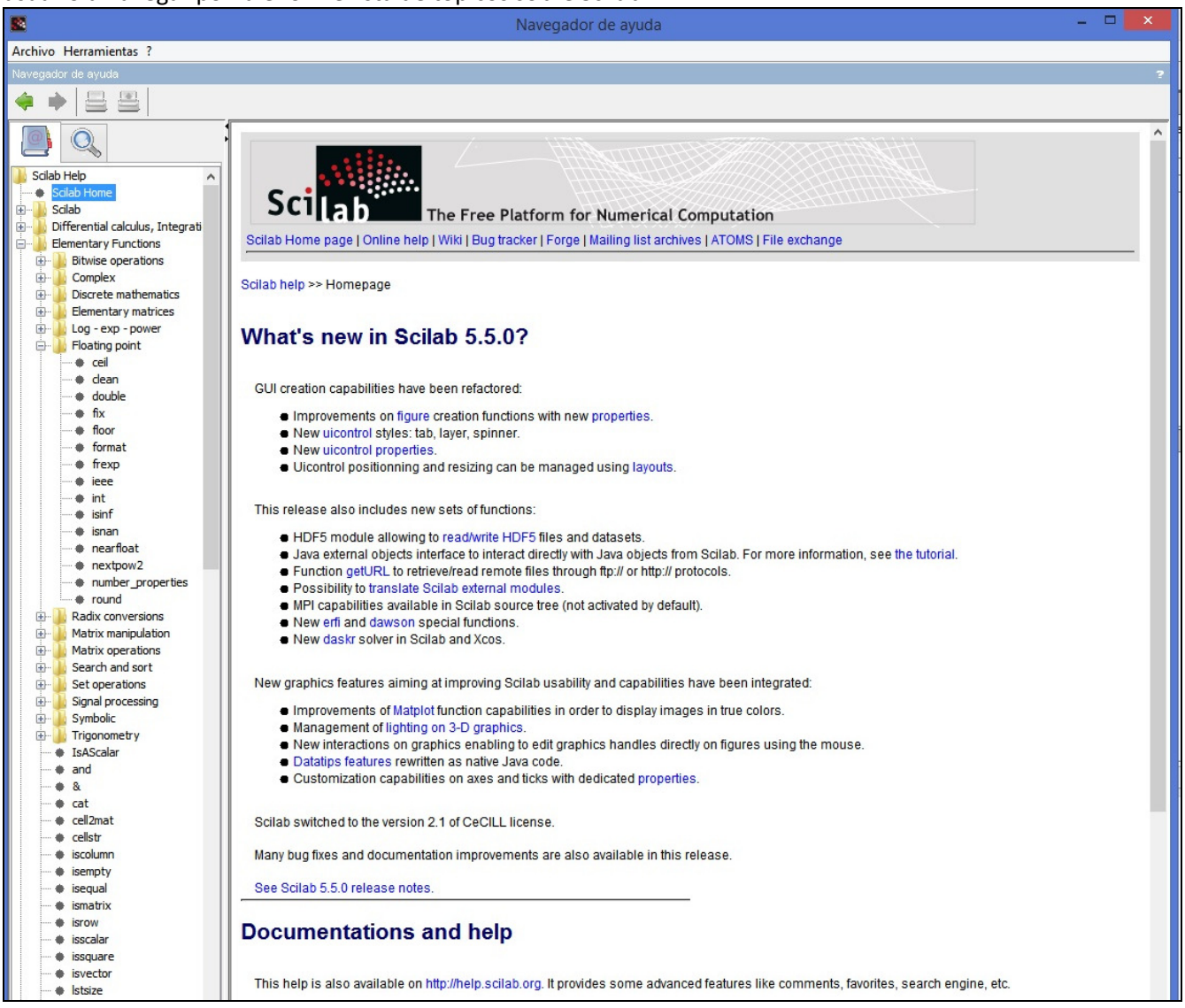

Figura 1.2.- Ventana inicial del Navegador de ayuda de Scilab

También se puede solicitar ayuda sobre un comando o función específica de Scilab, por ejemplo si deseamos ayuda sobre la instrucción `format`, se puede escribir en la línea de comandos help `format`. Con esto se abre la ventana del navegador de comandos mostrando la página específica que describe la instrucción format y sus diferentes variantes, como se muestra en el cuadro 1.1.

Scilab Help >> Elementary Functions > Floating point > format

Ruta de esta página de ayuda

FORMAT

number printing and display format

CALLING SEQUENCE

```
format([type],[long])
v = format()
format(m)
```

ARGUMENTS

type
: character string

long
: integer (max number of digits (default 10))

v
: a vector for the current format.
v(1) is a type of format : 0 for 'e' and 1 for 'v'
v(2) is a number of digits

m
: a vector to set new format
m(1) is a number of digits
m(2) is a type of format : 0 for 'e' and 1 for 'v'

DESCRIPTION

Sets the current printing format with the parameter type; it is one of the following :

"v"
: for a variable format (default)

"e"
: for the e-format.

long defines the max number of digits (default 10). format() returns a vector for the current format: first component is the type of format (1 if 'v' ; 0 if 'e'); second component is the number of digits.
In the old Scilab versions, in "variable format" mode, vector entries which are less than %eps times the maximum absolute value of the entries were displayed as "0". It is no more the case, the clean function can be used to set negligible entries to zeros.

EXAMPLES

```
x=rand(1,5);
format('v',10);x
format(20);x
format('e',10);x
format(20);x

x=[100 %eps];
format('e',10);x
format('v',10);x

format("v")
```

SEE ALSO

- write — write in a formatted file
- disp — displays variables
- print — prints variables in a file
- clean — cleans matrices (round to zero small entries)

Cuadro 1.1.- Página del navegador de ayuda para la instrucción `format`.

MANEJO DE MATRICES Y VECTORES.

Las Matrices son el principal tipo de datos que maneja Scilab. La manera más sencilla de escribir una matriz es como una lista de elementos. Solamente hay que seguir unas convenciones básicas:

- Toda matriz es una arreglo rectangular de elementos y está organizada por:
  - renglones o filas (en forma horizontal)
  - columnas (en forma vertical).
- Los elementos de una fila se escriben separados con espacios o comas.
- Se Usa ; (punto y coma) para indicar el fin de cada renglón (fila).
- Se encierra la lista entera de elementos con paréntesis cuadrados, [ ].

Ejemplo: Para introducir la siguiente matriz de dos filas y dos columnas: $A=\begin{bmatrix}1 & 2\\ 3 & 4\end{bmatrix}$:

```
 A=[ 1 2; 3 4]
A =
   1. 2.
   3. 4.
```

Una matriz de una sola fila o de una sola columna se denomina ***vector***:

Ejemplo: Introducir el vector fila $b=\begin{bmatrix}5 & 6\end{bmatrix}$ y el vector columna $c=\begin{bmatrix}7\\ 8\end{bmatrix}$

```
 b=[ 5 6]
b =
   5. 6.
 c=[ 7;8]
c =
   7.
   8.
```

Para evitar el despliegue de los resultados en la pantalla cada vez que se asigna una variable use el operador ";" al final de la expresión, para el ejemplo anterior:

```
 b=[ 5 6];
 c=[ 7;8];
```

CONCATENANDO MATRICES:

Se pueden juntar dos matrices con el mismo número de columnas o con el mismo número de renglones para formar una matriz más grande:

Ejemplo:

```
 A=[1 2 3; 4 5 6;7 8 9];
 R=[10 11 12];
 B=[A ; R]          //Junta dos matrices de tres columnas
B =
     1.     2.     3.
     4.     5.     6.
     7.     8.     9.
    10.    11.    12.
 C=[10; 11; 12];
 B=[A C]            //Junta dos matrices de tres filas
B =
     1.     2.     3.    10.
     4.     5.     6.    11.
```

```
     7.     8.     9.    12.
```

SUBMATRICES:

Si `A` es una matriz, la notación `A(i:j,m:n)` se refiere a la submatriz formada por los elementos desde la fila `i` hasta la fila `j` y desde la columna `m` hasta la columna `n`.

☞ Obsérvese que i, j, m, n sólo pueden ser números enteros mayores a cero y menores al tamaño correspondiente de la matriz.

Ejemplo: Si B es la matriz del ejemplo anterior

```
 C=B(2:3,2:3)
C =
     5.     6.
     8.     9.
```

MANIPULACIÓN DE MATRICES ELEMENTO A ELEMENTO:

Si `A` es una matriz, la notación `A(i,j)` hace referencia al elemento individual de la fila `i` y la columna `j`. Esto permite leer o modificar cada elemento de matriz en base a su posición renglón columna especificados por los índices `i, j`

El símbolo dos puntos ":" permite referirse a un rango de valores enteros, lo cual puede ser usado para referirse a un rango de filas o de columnas.

Ejemplo: Generar todos los números enteros del 1 al 20, con incrementos de 2 en 2

```
x=1:2:10
x =
     1.     3.     5.     7.     9.
```

Si el incremento no se especifica, se toma por default 1:

```
 x=1:10
x =
     1.     2.     3.     4.     5.     6.     7.     8.     9.    10.
```

Usado como índice, la notación de dos puntos permite omitir no solamente el incremento, sino el inicio y el final:

Ejemplo: Partiendo del ejemplo anterior

```
 x(1:4) //los elementos del 1 al 4
ans =
     1.     2.     3.     4.
 x(:)  //los elementos desde el primero hasta el último en forma de columna
ans =
     1.
     2.
     3.
     4.
     5.
     6.
     7.
     8.
     9.
    10.
```

Es decir, el uso de dos puntos para la designación de filas y columnas implica, respectivamente, todas las filas o columnas;

Ejemplo: `A(:,3)` representa todas las filas en la columna tres, `A(2,:)` representa todas las columnas en la fila dos

```
 A=[1 2 3;4 5 6;7 8 9]
A =
     1.     2.     3.
     4.     5.     6.
     7.     8.     9.
 A(:,2)
ans =
     2.
     5.
     8.
 A(2,:)
ans =
     4.     5.     6.
```

☞ **Observaciones**:

- Usar sólo dos puntos, por ejemplo `A(:)`, reagrupa todos los elementos de una matriz en un vector columna.
- Colocar datos fuera del rango actual de una matriz rellena con ceros las zonas no especificadas, manteniendo la forma rectangular de la matriz.
- Fijar las filas o columnas de una matriz igual a la matriz vacía `[ ]` elimina estas filas o columnas.
- Los valores lógicos `%F` y `%T` pueden utilizarse para seleccionar partes de un vector. Los elementos falsos (`F`) se eliminan, los elementos verdaderos (`T`) se retienen.
- Los valores lógicos pueden ser generados como resultado de desigualdades.

Ejemplo: Prosiguiendo con la matriz del ejemplo anterior

```
 A(:)  // Convierte a columna
ans =
     1.
     4.
     7.
     2.
     5.
     8.
     3.
     6.
     9.
 A(2,5)=10 // Agrega un elemento fuera de rango
A =
     1.     2.     3.     0.     0.
     4.     5.     6.     0.    10.
     7.     8.     9.     0.     0.
 A(:,4:5)=[] // Elimina las columnas 4 y 5
A =
     1.     2.     3.
     4.     5.     6.
     7.     8.     9.
 B=A(:)'   // Convierte a renglón
B =
     1.     4.     7.     2.     5.     8.     3.     6.     9.
 C = (B>5)   // Genera un vector de valores lógicos
C =
     F  F  T  F  F  T  F  T  T
 B(C)  // Selecciona los elementos de B con valor mayor a 5
ans =
     7.     8.     6.     9.
```

EJEMPLOS VARIOS:

```
A=[1 2 3; 4 5 6; 7 8 9]   //Crea la matriz A
A(3,3)=0                  //Cambia un elemento de la matriz
A(2,6)=1;                 //Agrega un elemento fuera de rango
A=[1 2 3; 4 5 6; 7 8 9]   //Vuelve a crear la matriz A
B=A(3:-1:1,:)             //Crea la matriz B con las filas A en orden inverso
C=[A B(:,[1 3])]          //Añade la 1ª y 3ª columna de B a la derecha de A
B=A(1:2,2:3)              //Extrae submatriz de A
B=A(:)                    //Convierte A en vector columna
B=B'                      //transpone la columna para convertirla a fila
B(:,2)=[]                 //Elimina la segunda columna de B
B=B'                      //Transpone una matriz
A=B                       //Copia la matriz B en A
B(2,:)=[]                 //Elimina la segunda fila de B
x=-3:0.5:3                //Crea un vector desde -3 a 3 con incrementos de 0.5
y=abs(x)>0                //Marca con 1's los elementos positivos de x
y=x(y)                    //Selecciona los elementos positivos de x
y=x([1 1 1 1])            //Crea y tomando el primer elemento de X tres veces
x(abs(x)<2)=[]            //Elimina valores de x tales que -2<x<2
```

MATRICES CON ELEMENTOS COMPLEJOS.

Scilab permite el manejo de números complejos como ya se dijo, mediante el uso de la variable predefinida: `%i`$=\sqrt{-1}$.

Ejemplo: Para introducir la siguiente matriz con elementos complejos: $A=\begin{bmatrix} 1+i & 2-3i \\ 3-3i & 4+2i \end{bmatrix}$, se puede proceder como sigue:

```
A=[1+%i 2-%i*3; 3-%i*3 4+%i*2];
```

ó bien, separando en parte real y parte imaginaria:

```
A=[1 2; 3 4]+%i*[1 -3; -3 2];
```

EL CONJUGADO Y EL TRASPUESTO DE UNA MATRIZ.

El operador comilla (') obtiene el traspuesto conjugado de una matriz, si se desea solamente trasponer la matriz se debe usar punto comilla (.') o bien, la función `conj`.

Ejemplo: Partiendo del ejemplo anterior

```
 A'    // Obtiene la matriz traspuesta y conjugada
ans =
   1. - i      3. + 3.i
   2. + 3.i    4. - 2.i
 A.'    // Obtiene la matriz traspuesta sin conjugar
ans =
   1. + i      3. - 3.i
   2. - 3.i    4. + 2.i
 conj(A')    // Misma operación que A.'
ans =
   1. + i      3. - 3.i
   2. - 3.i    4. + 2.i
```

OPERACIONES ARITMÉTICAS CON MATRICES.

Las operaciones aritméticas básicas: suma, resta, multiplicación, e inclusive división se pueden realizar entre matrices de dimensiones compatibles mediante los operadores +, -, *, /.

☞ En el caso de la división matricial, ésta debe interpretarse como sigue, Si A y B son matrices de dimensiones compatibles (número de filas de B igual a número de columnas de A), y si B tiene inversa, entonces

$$A / B = A * B^{-1}$$

Además de las operaciones aritméticas básicas entre matrices anteriores, Scilab dispone de operaciones especiales elemento a elemento, las cuales se enumeran en la siguiente tabla, en la cual se supone que `c` es un escalar arbitrario y las matrices `A` y `B` siguientes:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ & & \dots & \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{bmatrix}, \; B = \begin{bmatrix} b_{11} & b_{12} & \dots & b_{1m} \\ b_{21} & b_{22} & \dots & b_{2m} \\ & & \dots & \\ b_{n1} & b_{n2} & \dots & b_{nm} \end{bmatrix}$$

Operaciones elemento a elemento con matrices	
Suma con escalar `A+c = c+A`	$\mathtt{A+c} = \begin{bmatrix} a_{11}+c & a_{12}+c & \dots & a_{1m}+c \\ a_{21}+c & a_{22}+c & \dots & a_{2m}+c \\ & & \dots & \\ a_{n1}+c & a_{n2}+c & \dots & a_{nm}+c \end{bmatrix}$

Multiplicación por escalar
`A*c = c*A =`
`A.*c = c.*A`
	$\mathtt{A*c} = \begin{bmatrix} a_{11}*c & a_{12}*c & \dots & a_{1m}*c \\ a_{21}*c & a_{22}*c & \dots & a_{2m}*c \\ & & \dots & \\ a_{n1}*c & a_{n2}*c & \dots & a_{nm}*c \end{bmatrix}$

Multiplicación elemento a elemento `A.*B = B.*A`	$\mathtt{A.*B} = \begin{bmatrix} a_{11}*b_{11} & a_{12}*b_{12} & \dots & a_{1m}*b_{1m} \\ a_{21}*b_{21} & a_{22}*b_{22} & \dots & a_{2m}*b_{2m} \\ & & \dots & \\ a_{n1}*b_{n1} & a_{n2}*b_{n2} & \dots & a_{nm}*b_{nm} \end{bmatrix}$
División elemento a elemento `A./B`	$\mathtt{A./B} = \begin{bmatrix} a_{11}/b_{11} & a_{12}/b_{12} & \dots & a_{1m}/b_{1m} \\ a_{21}/b_{21} & a_{22}/b_{22} & \dots & a_{2m}/b_{2m} \\ & & \dots & \\ a_{n1}/b_{n1} & a_{n2}/b_{n2} & \dots & a_{nm}/b_{nm} \end{bmatrix}$

Elevación a Potencia elemento a elemento
`A.^c`
`c.^A`
`A.^B`
	$\mathtt{A.\hat{}c} = \begin{bmatrix} a_{11}^{\,c} & a_{12}^{\,c} & \dots & a_{1m}^{\,c} \\ a_{21}^{\,c} & a_{22}^{\,c} & \dots & a_{2m}^{\,c} \\ & & \dots & \\ a_{n1}^{\,c} & a_{n2}^{\,c} & \dots & a_{nm}^{\,c} \end{bmatrix}$ $\mathtt{c.\hat{}A} = \begin{bmatrix} c^{a_{11}} & c^{a_{12}} & \dots & c^{a_{1m}} \\ c^{a_{21}} & c^{a_{22}} & \dots & c^{a_{2m}} \\ & & \dots & \\ c^{a_{n1}} & c^{a_{n2}} & \dots & c^{a_{nm}} \end{bmatrix}$
	$\mathtt{A.\hat{}B} = \begin{bmatrix} a_{11}^{\,b_{11}} & a_{12}^{\,b_{12}} & \dots & a_{1m}^{\,b_{1m}} \\ a_{21}^{\,b_{21}} & a_{22}^{\,b_{22}} & \dots & a_{2m}^{\,b_{2m}} \\ & & \dots & \\ a_{n1}^{\,b_{n1}} & a_{n2}^{\,b_{n2}} & \dots & a_{nm}^{\,b_{nm}} \end{bmatrix}$

Ejemplo: Encontrar la solución del sistema de tres ecuaciones con tres incógnitas siguiente

$$x_1 + x_2 + x_3 = 1$$
$$x_2 + x_3 = 2$$
$$x_3 = 3$$

```
 A=[1 1 1;0 1 1;0 0 1];    //se define la matriz del sistema
 b=[1 2 3]';         //se define el vector columna de términos independientes
 x=A^-1 *b                 //se obtiene la solución
x =
    -1.
    -1.
     3.
 x=inv(A)*b                //inv(A) es lo mismo que A^-1
x =
    -1.
    -1.
     3.
```

Ejemplos varios: Continuando con el ejemplo anterior

```
 A^-1          // Obsérvese la diferencia del operador ^
 A.^-1         // y el operador .^
 A^3           // Calcula el producto matricial A*A*A
 A.^3          // Eleva al cubo cada elemento de A
 B=A*A'        // Obtiene una nueva matriz simétrica
 A*B           // Obsérvese la diferencia entre la multiplicación matricial *
 A.*B          // y la multiplicación elemento a elemento .*
```

Ejemplos de Errores comunes: Continuando con el ejemplo anterior se pueden intentar las siguientes operaciones que fallarán por no respetar las dimensiones de las matrices y su compatibilidad con las operaciones:

```
 b^2           // Intento de multiplicar un vector columna por si misma
Warning: Syntax "vector ^ scalar" is obsolete. It will be removed in Scilab 6.0.
  Use "vector .^ scalar" instead.
b*A            /// Intento de multiplicar matrices de dimensiones incompatibles
 !--error 10
Multiplicación inconsistente.Inner matrix dimensions must agree.

 A(3,4)        /// Intento de acceder a un elemento fuera de las dimensiones de A
     !--error 21
Índice inválido.
  A/0          /// Obsérvese que a diferencia de Matlab, dividir entre cero SI es ERROR
   !--error 27
División por cero...
```

FUNCIONES INTRÍNSECAS DE SCILAB

Scilab proporciona un número gigantesco de funciones. Algunas funciones son intrínsecas o construidas en el propio núcleo ejecutable de Scilab. Otras están disponibles en librerías externas (archivos sci) distribuidos con Scilab (Toolboxes). Y otras son adicionadas por los usuarios, o grupos de usuarios, para alguna aplicación específica en archivos sci.

Usando el comando `help <nombre de Función>` Scilab despliega una explicación concisa (en inglés) sobre la función así como del parámetro o parámetros necesarios para su correcta ejecución y de los resultados que produce, incluyendo un breve código para algunos ejemplos representatativos. Por ejemplo, en el cuadro 1.2 se muestra la ayuda para la función `sqrt`.

Scilab Help >> Elementary Functions > Log - exp - power > sqrt

SQRT

square root

CALLING SEQUENCE

y=sqrt(x)

ARGUMENTS

x

real or complex scalar or vector/matrix

DESCRIPTION

sqrt(x) is the vector/matrix of the square root of the x elements. Result is complex if element of x is negative.

EXAMPLES

```
A = matrix(4:4:16,2,2)
sqrt(A)
sqrt(-1)
```

SEE ALSO

- hat — (^) exponentiation
- sqrtm — matrix square root

Report an issue

<< polar | Log - exp - power | sqrtm >>

Cuadro1.2.- Ayuda de Scilab para la función `sqrt`.

FUNCIONES BÁSICAS CON MATRICES:

Se puede obtener un listado de las funciones matemáticas elementales de Scilab con ayuda del navegador de ayuda en la página: **Scilab Help >> Elementary Functions**

La mayoría de las funciones de Scilab pueden operar tanto sobre escalares como sobre matrices, cuando operan sobre matrices la función la aplican elemento a elemento, a menos que por definición se especifique lo contrario.

Ejemplo:

```
 A=[1 1 1;0 1 1;0 0 1]*%pi/2     //Define una matriz
A =
    1.5707963    1.5707963    1.5707963
    0.           1.5707963    1.5707963
    0.           0.           1.5707963

sin(A)        //Calcula el seno de cada elemento
ans =
     1.     1.     1.
     0.     1.     1.
     0.     0.     1.
cos(A)        //Calcula el coseno de cada elemento
ans =
    6.123D-17    6.123D-17    6.123D-17
```

```
    1.            6.123D-17    6.123D-17
    1.            1.           6.123D-17
exp(A)        //Calcula el exponencial de cada elemento e.^A
ans =
    4.8104774     4.8104774    4.8104774
    1.            4.8104774    4.8104774
    1.            1.           4.8104774
expm(A)       //Calcula la matriz exponencial e^A
ans =
    4.8104774     7.5562802    13.490969
    0.            4.8104774    7.5562802
    0.            0.           4.8104774
```

La siguiente es una lista de funciones básicas útiles que operan sobre matrices:

Función	**Descripción**
`det(A)`	Determinante de la matriz cuadrada A
`inv(A)`	Inversa de la matriz cuadrada A
`spec(A)`	Valores y vectores propios de la matriz cuadrada A
`[n,m]=size(A)`	Dimensiones de la matriz A (n renglones, m columnas)
`length(A)`	Producto de las dimensiones de A
`zeros(n,m)`	Genera una matriz de puros ceros de dimensiones nxm
`ones(n,m)`	Genera una matriz de puros unos de dimensiones nxm
`eye(n,m)`	Genera una matriz identidad de dimensiones kxk, donde k=min(n,m) y rellena con ceros para tener dimensiones n x m.
`rand(n,m)`	Genera matriz n x m con valores aleatorios entre 0 y 1.
`linspace(a,b,n)`	Genera vector fila de n valores desde a hasta b en incrementos constantes de (b-a)/(n-1). Equivale a `a:(b-a)/(n-1):b`
`logspace(a,b,n)`	Genera vector fila de n valores desde 10^a hasta 10^b en incrementos exponenciales. Equivale a `10.^linspace(a,b,n)`

EL ESPACIO DE TRABAJO DE SCILAB:

Se denomina espacio de trabajo al conjunto de variables creadas por el usuario en cada sesión. El espacio de trabajo está vacío al inicio de cada sesión, o después de un comando `clear`. Se puede obtener información de las variables en el espacio de trabajo visualizando el navegador de variables (Variable Browser) como se muestra en la figura 1.3.

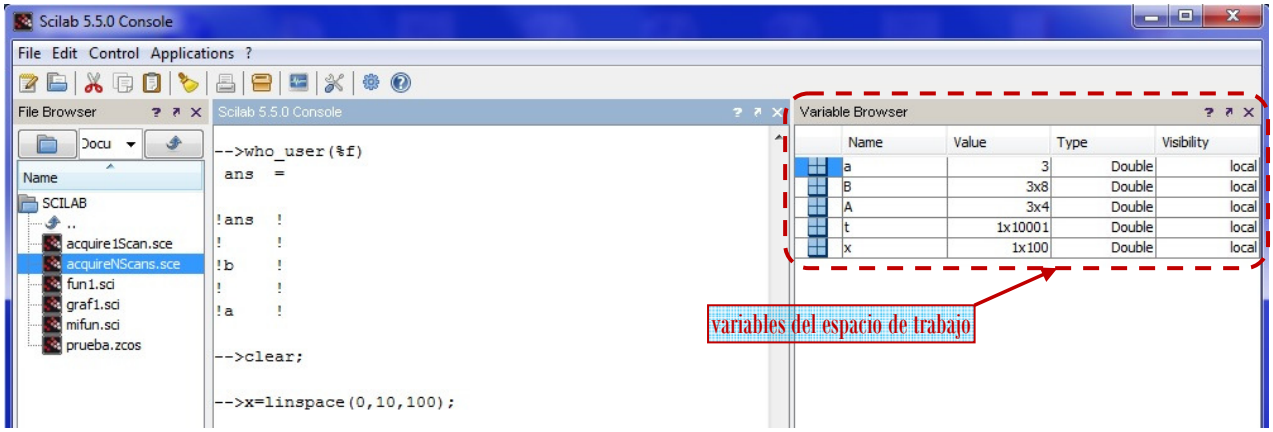

Figura 1.3.- navegador de variables mostrando el espacio de trabajo de Scilab.

O si se prefiere, se puede también utilizar el comando `who_user` o `who`; por ejemplo:

Ejemplo:

```
clear         // Limpia el espacio de trabajo
who_user      // obsérvese que el espacio de trabajo está vacío
ans  =
     []
A=rand(3,3)
A =
    0.1454259    0.2165825    0.5196690
    0.4306040    0.9683006    0.8674864
    0.6140478    0.6637877    0.2877042
[n,m]=size(A)
n =
     3.
m =
     3.
who_user
User variables are:
 n         m         A         ans

 Using 20 elements out of 9990166
 ans  =
!n    !
!     !
!m    !
!     !
!A    !
!     !
!ans  !
```

GUARDAR EL ESPACIO DE TRABAJO DE UNA SESIÓN: COMANDOS *SAVE* Y *LOAD*

Cuando uno ejecuta el comando `clear` o cierra la sesión de Scilab, el espacio de trabajo se pierde y todas las variables creadas así como sus valores desaparecen de la memoria. En ocasiones puede ser necesario interrumpir el trabajo con Scilab y poder recuperarlo más tarde en el mismo punto en el que se dejó (con el mismo espacio de trabajo.

Para guardar el estado de una sesión existe el comando `save`. Si se teclea `save(nombre_de_archivo)`. El espacio de trabajo se guarda en el archivo especificado y puede recuperarse la siguiente vez que se arranque el programa con el comando `load(nombre_de_archivo)`.

si no se desea guardar el espacio de trabajo completo se pueden guardar también matrices y vectores de forma selectiva y en archivos con nombre especificado por el usuario. Por ejemplo, el comando siguiente

```
save(nombre_de_archivo,'A','x','y')
```

guarda las variables A, x e y en el archivo especificado. Para recuperarlas en otra sesión es necesario teclear

```
load(nombre_de_archivo,'A','x','y')
```

Ejemplo

```
who_user            // checamos como está el espacio de trabajo
User variables are:
 n         m         A         %_sodload

save('variables.dat','A','m','n') // salvamos algunas variables
clear         // borramos el espacio de trabajo
who_user      // checamos espacio de trabajo vacío
```

```
load('variables.dat','A','m','n') /// rescatamos las variables salvadas
who_user
User variables are:

 A         m         n         %_sodload
```

GUARDAR SESIÓN Y COPIAR SALIDAS: COMANDO *DIARY*

Mientras que el comando `save` crea archivos binarios o ASCII que guardan el estado de la sesión para su posterior recuperación, también se puede almacenar en un archivo de texto toda la actividad desplegada en la consola (comandos introducidos desde el teclado y respuestas obtenidas) durante una sesión o fragmento de sesión de Scilab. Esto se hace puede hacer con el comando `diary` en la forma siguiente:

```
diary('filename.txt')      // Abre el archivo del diario de la sesión
…
diary('filename.txt','pause'|'off')     // suspende la ejecución de diary
…
diary('filename.txt','resume'|'on')     //reanuda la ejecución de diary
…
diary(0)                   //cierra el archivo abierto de la sesión.
```

Para poder acceder al fichero `filename.txt` con *Notepad* o *Word* es necesario que el archivo halla sido cerrado previamente.

DESARROLLO DE LA PRÁCTICA.

Durante el desarrollo de esta práctica se presenta el funcionamiento del ambiente de trabajo de Scilab, en cada uno de las explicaciones se dan ejemplos del funcionamiento del ambiente de Scilab, y se proponen algunos ejercicios.

1. Probar todos los ejemplos propuestos por el profesor conforme los va explicando.
2. Realizar todos los ejercicios propuestos.
3. Contestar el cuestionario de evaluación de la práctica.

REPORTAR:

1. Localizar el listado de funciones elementales en el navegador de ayuda, en la página la página: **Scilab Help >> Elementary Functions** e imprimirlo.
2. Elegir de la lista, dos de las funciones más raras que te llamen la atención (no vistas en esta clase) y explicar con tus propias palabras para que sirven. Dar ejemplos explicados de aplicación de cada una éstas dos.