Introducción a las Prácticas con MATLAB

Ventanas más usadas del escritorio o pantalla, "desktop"

command window
editor
workspace
command history
current folder

Para cambiar el estilo del Desktop por ejemplo respecto al tipo de letra y/o a su tamaño, se ejecuta la siguiente secuencia de opciones desde la Barra de Herramientas (barras superiores del Desktop) :

`File` → `Preferences` → `Fonts` → **cambio** → `Apply`

La ventana de comandos

Para seleccionar la configuración del escritorio en la que sólo aparece visible la ventana de comandos debemos seleccionar en la Barra de Herramientas la secuencia de opciones:

`Desktop` → `Desktop layout` → `Command Window Only`

Para recuperar la configuración original del escritorio la secuencia de opciones es:

`Desktop` → `Desktop layout` → `Default`

El prompt >> en la ventana de comandos indica que MATLAB está listo para recibir los comandos (también llamados instrucciones)

La tecla <`enter`> pulsada después de escribir una instrucción produce la ejecución de dicha instrucción.

```
>> 2+2
ans =
     4
```

... al final de una línea indica que la instrucción continúa en la línea siguiente.

```
>> 2 + ...
3
ans =
     5
```

`ans` (del inglés answer) es el último resultado calculado no asignado a una variable

Los comentarios van precedidos del símbolo `%` y pueden ocupar toda la línea o ir escritos después de una instrucción

```
>> 2+2                      % ejemplo de suma
ans =
     4


>> % esto es solo un comentario
```

Current Folder (es la carpeta o directorio de trabajo, también llamado Working Directory)

Desde la Barra de Herramientas se puede determinar cual es la carpeta de trabajo y "subir y bajar" de unos directorios a otros. Obviamente, también puede hacerse desde la ventana de comandos (Command Window), con las instrucciones que se dan a continuación:

```
>> pwd   % Devuelve el nombre del directorio de trabajo ...
            pwd es abreviatura de ‘‘print working directory’’
      ans =
            C:\Users\carballor\Documents\MATLAB


>> cd  C:\Users\carballor\Documents
      ans =
            C:\Users\carballor\Documents

>>  % La instruccion anterior nos lleva a la carpeta ‘‘Documents’’
>>  % cd es abreviatura de ‘‘change directory’’
>>  pwd     % confirmamos que la nueva carpeta es la de nombre ‘‘Documents’’
    ans =
            C:\Users\carballor\Documents

>>  cd ..          % sube un nivel
    ans =
            C:\Users\carballor

>>  cd Documents   % baja un nivel
>>  pwd
    ans =
            C:\Users\carballor\Documents
```

Nótense las tres variedades de sintaxis utilizadas con "cd":

`cd sitio` nos sitúa en el dirección especificada en `sitio`
`cd ..` nos "sube" a la carpeta o directorio inmediatamente superior
`cd foldername` nos "baja" a la carpeta denominada `foldername` dentro del directorio de trabajo

Algunas funciones o comandos del sistema operativo

```
>> dir                    % lista de todos los ficheros en el Current Folder
                          % el comando  ls  hace lo mismo que el comando dir
>> type nombrefichero     % para visualizar por pantalla el contenido ascii de ese fichero
>> pwd
>> cd nombredireccion
>> cd ..
>> delete nombrefichero   % para borrar ese fichero del directorio
>> exit                   % comando para cerrar la sesion de MATLAB
>> quit                   % comando para cerrar la sesion de MATLAB
```

Más adelante veremos otras, como `diary, who, whos, clear, format, help, doc, lookfor`, entre otras.

La función `diary`

```
>> diary  filename   % para guardar en formato ascii todo lo que se introduce y sale ...
por pantalla
>> diary martes.txt
>> 2+2
ans =
       4
>> 2-3
ans =
      -1
>> diary off
```

En el fichero `martes.txt`, localizado en el Current Folder, quedan escritas las instrucciones anteriores y sus salidas. Al fichero se le añadió la extensión `.txt` para que desde los PCs se abra directamente con el programa asignado a esa extensión (generalmente será el Bloc de Notas).

Si en cada sesión de prácticas ejecutamos al inicio `>> diary name.txt` y al final `>> diary off`, podremos disponer de un fichero donde quede almacenado todo el trabajo realizado.

Efectuando `>> type martes.txt` tendríamos el siguiente resultado:

```
>> type martes.txt

>> 2+2
ans =
       4
>> 2-3
ans =
      -1
>> diary off
```

Si queremos borrar el fichero `martes.txt` desde la sesión de MATLAB efectuamos:

```
>> delete martes.txt
```

Asignación a variables: nombre = expresión

El nombre de la variable tiene que empezar por una letra seguida de letras, números o signos

```
>> a = 4+5
   a =
       9
```

Una vez definida una variable, se puede operar con ella

```
>> a + 3
   ans =
       12
```

```
>> a = a + 4       % notese que el signo igual representa una asignacion, ...
y no una igualdad
   ans =
       13
>> c = a - 3
   ans =
       10
```

Separación de instrucciones en una misma línea

En una misma línea podemos escribir más de una instrucción, separando éstas mediante "una coma" o mediante "un punto y coma". Cuando se usa punto y coma no se muestra el resultado. Ejemplo:

```
>> a=3 , b= 6, c=a+b
   a =
       3
   b =
       6
   c =
       9

>> a=5 ; b= -6; c=a+b
   c =
      -1
```

`a` y `b` toman valores `5` y `-6` respectivamente, aunque estas asignaciones no se muestran por pantalla.

Las instrucciones que finalizan con " ; " decimos que son instrucciones mudas.

Minúsculas y mayúsculas

Nótese que MATLAB distingue entre mayúsculas y minúsculas (en toda la sintaxis: variables, funciones, etc).

Listado de las variables en el Workspace: `who` y `whos`

El Workspace es el espacio de trabajo interno de MATLAB
`whos` da más información que `who`

Nótese que MATLAB interpreta los escalares como matrices 1×1, tanto si son valores numéricos como si son variables.

Borrar variables: `clear`

```
>> clear a       % borra la variable de nombre a
>> clear a b     % borra las variables a y b
>> clear         % borra todas las variables
```

Aritmética básica

Sumar, restar, multiplicar, dividir y potencia

```
+ , - , * ,  / , ^
```

utilizando paréntesis ( ) cuando sea necesario.

Los operadores anteriores se usan tanto para las operaciones con escalares como para las mismas operaciones con matrices.

Seguidamente recordamos la jerarquía en las operaciones, también llamada precedencia: primero potencia, luego multiplicación/división y finalmente suma/resta

```
>> 2 + 3 - 4^2/8
   ans =
        3

>> 2 + 3 - 4^(2/8)
   ans =
        3.5858

>> (2 + 3 - 4^2)/8
   ans =
        -1.3750
```

Nótese como MATLAB usa el punto decimal, y no la coma.

```
>> 4^.5           % raiz cuadrada
   ans =
         2

>> 4^(1/2)
   ans =
         2
```

Formatos para los resultados numéricos

El formato por defecto para la presentación de resultados es "format short", que proporciona un máximo de 7 cifras significativas. Por ejemplo:

```
>>   100/3
 ans =
       33.3333

>>  1000/3
 ans =
       333.3333

>> 10000/3
 ans =
       3.3333e+003
```

Los formatos más utilizados son los siguientes:

```
>> format short   % decimal corto
>> format long    % decimal largo
>> format rat     % racional
```

Una instrucción del tipo `>> format formattype1` activará el formato `formattype1`, permaneciendo éste como formato mientras no se active otro.

```
>> format rat
>> 7/3-2/3
   ans=
        5/3

>> format short
>> 7/3-2/3
   ans=
        1.6667

>> format long
>> 7/3-2/3
   ans=
        1.666666666666667
```

Constantes

```
>> format short
>> pi
   ans =
        3.1416

>> i      % es el numero complejo i
   ans =
        0 + 1.0000i

>> i^2
   ans =
         -1

>> format long
>> pi
   ans =
         3.141592653589793
```

Ctrl - c

Detiene una instrucción sin salir de MATLAB

Ayuda con función `help`

La ayuda más general se obtiene con `>> help`

De forma más específica, para obtener información sobre una función utilizamos:

```
>> help nombrefuncion
```

La función `>> doc` da información mucho más detallada sobre la función consultada

```
>> doc nombrefuncion
```

Al ejecutar cualquiera de las dos últimas instrucciones veremos que en la parte inferior de la salida por pantalla se muestran los nombres de otras funciones relacionadas, que pueden resultarnos de ayuda.

Adelantamos aquí que todas las funciones de MATLAB se escriben con todas las letras en minúscula.

Ejemplo de ejecución de la función `help`

```
>> help pwd

pwd - Identify current folder

    This MATLAB function displays the MATLAB current folder.

    pwd
    currentFolder = pwd

    Reference page for pwd

    See also cd, dir
```

Vemos cómo nos ofrece información sobre funciones relacionadas, por ejemplo `cd` y `dir`. "Reference page for pwd", "cd" y "dir" son en efecto enlaces url a documentación más detallada.
Mediante el enlace "Reference page for pwd" llegamos a la misma página que si efectuamos `>> doc pwd`

Cuando se quiere ejecutar la función `help` sin especificar nada más, puede hacerse desde la barra de herramientas o desde el "prompt" en la ventana de comandos. Si se hace desde el "prompt" la salida será una lista de enlaces, cada uno correspondiendo a una temática (`help topic`). Pinchando sobre cada enlace obtendremos nuevos enlaces a funciones específicas correspondientes a esa temática ("topic").

Si accedemos a la ayuda desde la barra de herramientas, seleccionando primero "help", a continuación "documentation" y seguidamente "Matlab" tenemos la ayuda ordenada por categorías en una presentación muy cómoda, que facilita encontrar las funciones que realizan las tareas que queramos llevar a cabo. Con esta ayuda sabremos el nombre de la función y su descripción.

Las funciones matemáticas van seguidas de paréntesis "( )". Dentro de los paréntesis se escribe el argumento o los argumentos, si son varios separados por comas.

Ejemplo de ayuda respecto de una función matemática desde el "prompt". En particular ayuda sobre la función "raíz cuadrada":

```
>> help sqrt
 SQRT   Square root.

     This MATLAB function returns the square root of each element of the array X.

    B = sqrt(X)

    Reference page for sqrt

    See also nthroot, realsqrt, sqrtm
```

```
>> sqrt(4)
   ans =
         2

>> sqrt(-1)
   ans =
        0 + 1.0000i
```

La raíz cuadrada de -1 es el número imaginario puro i

Ayuda con la función `lookfor`

Cuando no conocemos el nombre en MATLAB de la función matemática concreta que queremos usar, podemos buscarlo en la ventana de comandos con el comando `lookfor`, añadiendo a esta función una palabra clave relacionada con la función buscada. El único problema es que en general habrá muchas salidas, es decir, muchas funciones posibles, porque en las librerías de MATLAB (las denominadas toolboxes) habrá en general numerosas funciones que puedan tener alguna relación con esa palabra clave. Si la salida es larga podemos detener el proceso (`Ctrl-c`) después de haber visto unas cuantas funciones, puesto que en un nivel básico, como el de este curso, la que buscamos estará seguramente al principio. Las palabras clave (keywords) son las que aparecen en la primera línea de la ayuda. Por ejemplo para la función `sqrt` vista más arriba "square" y "root" son las palabras clave.

Si por ejemplo queremos averiguar el nombre de la función trigonométrica que calcula la tangente de un ángulo:

```
>> lookfor tangent
acot                    - Inverse cotangent, result in radian.
acotd                   - Inverse cotangent, result in degrees.
acoth                   - Inverse hyperbolic cotangent.
atan                    - Inverse tangent, result in radians.
atan2                   - Four quadrant inverse tangent.
atan2d                  - Four quadrant inverse tangent, result in degrees.
atand                   - Inverse tangent, result in degrees.
atanh                   - Inverse hyperbolic tangent.
cot                     - Cotangent of argument in radians.
cotd                    - Cotangent of argument in degrees.
coth                    - Hyperbolic cotangent.
tan                     - Tangent of argument in radians.
tand                    - Tangent of argument in degrees.
tanh                    - Hyperbolic tangent.
fixpt_atan2_demo        - Calculate Fixed-Point Arctangent
cordicatan2             - CORDIC-based four quadrant inverse tangent.
dtansig                 - Hyperbolic tangent sigmoid transfer derivative function.
```

Por los comentarios que se dan a la derecha, deducimos que la función buscada es `tan` si el argumento lo expresamos en radianes, y `tand` si el argumento lo expresamos en grados (degrees).

```
>> tan(45*pi/180)    % Calcula la tangente de 45 grados
ans =
       1.0000
```

Para encontrar la misma función utilizando la Ayuda (Help) en la barra de herramientas los pasos necesarios serían los siguientes:

Help → Documentaion → Matlab → Mathematics → Elementary Math → Trigonometry

Ejecutar las instrucciones de MATLAB desde un fichero *texto ascii* con extensión .m

Podemos escribir en un fichero de texto ascii, con extensión .m, el conjunto de líneas de comando o instrucciones que queremos ejecutar con Matlab. Este fichero, que podemos llamar por ejemplo "tarea.m", tiene que estar guardado en la carpeta de trabajo o "Current folder", que no es más que la carpeta que nos indica Matlab al escribir en la línea de comandos `>> pwd`

Sin más que escribir en la ventana de comandos `>> tarea` y seguidamente `<enter>`, se ejecutarán todas las líneas.
Un conjunto de instrucciones de MATLAB como el anterior, que se lleva a ejecución como un bloque, es lo que se conoce como *script*.

Un script tarea.m que podemos haber escrito tiene el contenido siguiente, que vemos con la función del sistema operativo `>> type`

```
>> type tarea.m

 format rat
 a = 5
 b = 7/3 ;
 5 + 2/3 - b
```

Ejecutando el script obtendremos por pantalla el resultado.

```
>> tarea

a =
      5
ans =
     10/3
```

Matlab cuenta con una ventana para la edición de scripts.

Definición de vectores y matrices

Ejemplo de creación de las siguientes matrices: $A = \begin{bmatrix} 1 & 2 & 1 & 0 \\ 0 & 3 & 1 & 2 \\ -1 & 2 & 3 & 1 \end{bmatrix}$, $f = \begin{bmatrix} -1 & 2 & 3 \end{bmatrix}$, $c = \begin{bmatrix} -1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$

```
>> A = [ 1 2 1 0 ; 0 3 1 2 ; -1 2 3 1]
>> f = [ -1 2 3 ]
>> v = [ -1; 2; 3 ; 4 ]

A =

     1     2     1     0
     0     3     1     2
    -1     2     3     1

f =

    -1     2     3

v =

    -1
     2
     3
     4
```

Nótese como las matrices y vectores se escriben entre corchetes. Los elementos de una misma fila se separan por espacios (o comas), y el cambio de fila se indica con punto y coma.

`f` y `v` son respectivamente matriz fila y matriz columna. En *Álgebra Lineal* se define un vector de $\mathbb{K}^n$ como una matriz columna con n elementos de $\mathbb{K}$. $\mathbb{K}$ es para nuestro caso el cuerpo de los números reales o el cuerpo de los números complejos.

Tamaño de una matriz

```
>> size(A)     % devuelve el numero de filas y el numero de columnas de la matriz A
ans =
     3     4

>> size(f)
ans =
     1     3

>> size(v)
ans =
     3     1

>> size(A,1)   % devuelve el numero de filas de la matriz A
ans =
     3

>> size(A,2)   % devuelve el numero de columnas de la matriz A
ans =
     4
```

Vemos que la función `size` admite dos argumentos, siendo el primero la matriz y el segundo el valor 1 o el valor 2. Para el valor 1 la salida es el número de filas, y para el valor 2 el número de columnas. La función `size` también admite recibir un único argumento, que es la matriz para la que se ha de determinar el tamaño, y en este caso devuelve dos números, el primero el número de filas y el segundo el número de columnas.

Para vectores y matrices fila resulta útil la función `length()` que da como salida el número de componentes del vector o de la matriz fila.

```
>> length(f)
ans =
     3

>> length(v)
ans =
     4
```

Traspuesta y traspuesta conjugada

```
>> transpose(A)  % obtiene la traspuesta de la matriz A
ans =
     1     0    -1
     2     3     2
     1     1     3
     0     2     1

>> A'            % obtiene la traspuesta conjugada de la matriz A
ans =
     1     0    -1
     2     3     2
     1     1     3
     0     2     1
```

Para matrices reales se utiliza `A'` para calcular la traspuesta, ya que al carecer de parte imaginaria, la traspuesta y la traspuesta conjugada coinciden, y la notación de la comilla es más sencilla.

Nótese la diferencia en matrices en el cuerpo de los complejos.

```
>> A = [ 2+i 0; -i 1 ]
A =
     2.0000 + 1.0000i        0
          0 - 1.0000i   1.0000

>> transpose(A)
ans =
     2.0000 + 1.0000i        0 - 1.0000i
          0             1.0000

>> A'
ans =
     2.0000 - 1.0000i        0 + 1.0000i
          0             1.0000
```

Operaciones básicas del Álgebra de Matrices

`+` `A+B` Suma de las matrices A y B, ambas del mismo tamaño.
`-` `A-B` Diferencia $A - B$ siendo A y B matrices del mismo tamaño.
`*` `a*A` Producto del escalar a por la matriz A.
`*` `A*B` Producto de las matrices A y B, siempre que sean multiplicables.
`^` `A^k` Si k es un natural positivo, se eleva la matriz A a la potencia k. A tiene que ser una matriz cuadrada.

Observaciones:

Se obtendrá mensaje de error si en las operaciones de suma y resta las matrices no son del mismo tamaño o si en la operación de multiplicación de dos matrices éstas no son multiplicables (número de columnas de la primera matriz distinto del número de filas de la segunda).

Otras operaciones con matrices

`.*` `A.*B` Producto de matrices elemento a elemento, con A y B del mismo tamaño.
`./` `A./B` División de matrices elemento a elemento, con A y B del mismo tamaño.
`+` `a+A` Suma el escalar a a todos los elementos de la matriz A.
`.^` `A.^k` Eleva cada elemento de la matriz A al exponente k, siendo k un real cualquiera. En este caso no hace falta que la matriz A sea cuadrada.

Varias formas de generar vectores (o matrices fila) equiespaciados

- Utilizando la siguiente sintaxis:

  elemento inicial: incremento: elemento final Por defecto el incremento es 1.

  Genera la matriz fila empezando con el primer elemento y continuando con los elementos incrementados siempre que éstos se encuentren entre el elemento inicial y el elemento final.

  Ejemplos:

```
>> -4:2
ans =
       -4    -3    -2    -1     0     1     2

>> 1:3.5
ans =
        1     2     3

>> 1:-0.5:-1.7
ans =
       Columns 1 through 5

        1.0000    0.5000         0   -0.5000   -1.0000

       Column 6

       -1.5000
```

- Función `linspace(elemento inicial, elemento final, numero de elementos resultantes)`. Genera una matriz fila de `n` elementos equiespaciados, desde el inicial hasta el final incluyendo ambos.

```
>> linspace(1,5,6)
ans =
      Columns 1 through 5
        1.0000    1.8000    2.6000    3.4000    4.2000
      Column 6
        5.0000
```

- La función `linspace(elemento inicial, elemento final)` es igual a la anterior, salvo que en este caso el número de elementos resultantes está fijado en `100`.

Matrices especiales

MATLAB tiene funciones para definir matrices de uso frecuente, entre las que destacamos:

`eye(m,n)` genera la matriz $m \times n$ con elementos iguales a 1 en la diagonal principal.

`eye(n)` genera la matriz identidad de orden n.

`ones(m,n)` genera la matriz $m \times n$ con todos los elementos iguales a 1.

`zeros(m,n)` genera la matriz $m \times n$ nula.

`diag(v)` siendo v matriz fila o columna de n componentes, genera la matriz cuadrada diagonal de orden n con las componentes de v en la diagonal principal.

`diag(A)` siendo A matriz $m \times n$, genera un vector cuyas componentes son los elementos de la diagonal principal de A.

Otras funciones del Álgebra de Matrices

`>> rank(A)` rango de la matriz A

`>> rref(A)` forma escalonada reducida de la matriz A (forma escalonada por filas con pivotes unidad y ceros por encima de los pivotes)

`>> det(A)` determinante de la matriz cuadrada A

`>> inv(A)` inversa de la matriz cuadrada A

`>> trace(A)` traza de la matriz cuadrada A

`>> norm(v)` norma del vector v

`>> dot(u,v)` producto escalar de los vectores u y v

`>> cross(u,v)` producto vectorial del vector u por el vector v

Variables simbólicas

`>> syms x` define `x` como variable simbólica

`>> syms y z t` define `y, z, t` como variables simbólicas

Cualquier expresión que contenga una variable simbólica es una expresión simbólica. Por ejemplo, después de ejecutar los comandos anteriores, podríamos operar con expresiones simbólicas que involucren esas variables:

```
>> x*y - 2*x*t + 5*x*y - 2*x*z/x
    ans =
          6*x*y - 2*t*x - 2*z
```

Para borrar una variable simbólica se utiliza el mismo comando que para borrar el resto de las variables. Ejemplo:

```
>> clear x
```

```
>> syms x real
```
define `x` como variable simbólica de tipo real

Vemos el efecto que tiene declarar una variable como real en la siguiente instrucción:

```
>>  syms a, syms b real, conj(a), conj(b) % esta funci\'on obtiene el conjugado  ...

    de un n\'umero o de una variable

ans =
conj(a)

ans =
b
```

Operaciones combinadas con matrices: operaciones básicas y aplicación de funciones matemáticas

Mostramos ejemplos de operaciones y aplicación de funciones utilizando la matriz $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

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

>> sqrt(A)    % se calcula la raiz cuadrada de cada elemento
   ans =
         1.0000    1.4142    1.7321
         2.0000    2.2361    2.4495
         2.6458    2.8284    3.0000

>> 2*A+A^2+cos(A)    % cos(A) es el coseno de cada elemento de A (considerado en radianes)
   ans =
        32.5403   39.5839   47.0100
        73.3464   91.2837  108.9602
       116.7539  141.8545  167.0889

>> log10(A)  % el logaritmo en base 10 de los elementos de A
   ans =
             0    0.3010    0.4771
        0.6021    0.6990    0.7782
        0.8451    0.9031    0.9542
```

Concatenar matrices

```
>> M1 = [ 1 2 ; 3 4 ]  , M2 = [ 1 2 ; 5 6 ]
M1 =
     1     2
     3     4
M2 =
     1     2
     5     6

>> [M1 M2]   % M2 esta separada por espacio (o coma) y por tanto se concatena a la derecha
ans =
     1     2     1     2
     3     4     5     6

>> [M1; M2]  % M2 esta separada por punto y coma y por tanto se concatena debajo
ans =
     1     2
     3     4
     1     2
     5     6

>> [M1 A]    % A es la matriz del apartado anterior. Da mensaje de error porque
los ordenes no son consistentes

Error using horzcat
Dimensions of matrices being concatenated are not consistent.
```

Extraer y manipular elementos de una matriz

Tomamos la matriz A de los apartados anteriores.

```
>> A
   A =
       1     2     3
       4     5     6
       7     8     9
```

Extraer

```
>> A(2,3)             % devuelve por pantalla el valor del elemento (2,3)
ans =
      6

>> B = A (3 , [2 3] ) % extrae en B los elementos que estan a la vez en la fila 3 y
en las columnas 2 y 3
B =
    8 9

>> B = A (1 , : )     % extrae en la matriz B la primera fila de A (y todas las columnas)
B =
    1 2 3
```

```
>> A(:,[3 1])         % la salida es la submatriz con todas las filas y las columnas ...
                        3 y 1, en ese orden
ans =
       3  1
       6  4
       9  7

>> As=A( [ 2 3],[2 3])  % crea la submatriz de las filas 2 y 3 y las
                          columnas 2 y 3
As =
        5  6
        8  9
```

Manipular

```
>> A(3,2) = 88
A =
     1     2     3
     4     5     6
     7    88     9

>> A(3,3) = 99
A =
     1     2     3
     4     5     6
     7    88    99

>> A
A =
     1     2     3
     4     5     6
     7    88    99

>> A(3,2)
ans=
       88

>> A([1  3] , 2:3 )= [-1 -1 ; -1 -1]   % 2:3 significa del indice 2 al indice 3 ...
                                          incrementando de uno en uno
A =
     1    -1    -1
     4     5     6
     7    -1    -1
```