MATLAB CLASE 1 lo mas básico

1. SINTAXIS GENERAL

En MATLAB, en general, las letras minúsculas y mayúsculas NO SON IGUALES.
La ejecución de cualquier comando puede abortarse mediante **CONTROL + C**.
Se pueden escribir varios comandos en una misma línea, separándolos por "**coma**" o por "**punto y coma**".
Se pueden recuperar comandos anteriores, usando las teclas de flechas arriba y abajo. Con las flechas izquierda y derecha nos podemos desplazar sobre la línea de comando y modificarlo.

1.1 Constantes

Enteros: `12 -3`
Reales: `8.01 -5.2 .056 1.4e+5 -.567e-21`
Complejos: `1+2i -3+j` (`i j` son símbolos que representan la unidad imaginaria)
Caracteres (entre apóstrofes): `'esto es una cadena de caracteres' 'string'`

1.2 Operaciones aritméticas elementales

Suma:	+
Resta:	-
Multiplicación:	*
División:	/
Exponenciación:	^

Se puede utilizar MATLAB como simple calculadora, escribiendo expresiones aritméticas y terminando por **RETURN** (**<R>**). Se obtiene el resultado inmediatamente a través de la variable del sistema **ans** (de answer). Si no se desea eco en el terminal, deben terminarse las órdenes por "**punto y coma**".

1.3 Variables

Los nombres de variables pueden tener a lo sumo 19 caracteres, letras y números. El primero debe ser una letra. No se pueden utilizar los caracteres especiales:

`+ - = * ^ < > ...`

Las variables en MATLAB no necesitan ningún tipo de declaración y pueden almacenar sucesivamente distintos tipos de datos: enteros, reales, escalares, matriciales, caracteres, etc. Se crean, simplemente, asignándoles un valor.
Se pueden eliminar variables mediante el comando `clear`

`clear`	elimina todas las variables que existan en ese momento
`clear a,b,c`	elimina las variables a, b y c

Atención: recuérdese que las variables **AB ab Ab** y **aB** SON DISTINTAS.

Para conocer en cualquier instante el valor almacenado en una variable basta con teclear su nombre.
Se pueden conocer todas las variables definidas hasta el momento tecleando el comando

`who`	lista las variables actuales
`whos`	como el anterior, pero más detallado

EJEMPLO

```
>> a=10; <R>
>> pepito=2.4/3, <R>
>> b=a+pepito; <R>
>> b <R>
b =
    10.800
>> b=b+4-0.5i <R>
b =
    14.800 - 0.5000 i
```

1.4 Formatos

Por defecto, cuando MATLAB nos muestra un valor real, nos muestra sólo cinco cifras significativas (formato corto). Se puede modificar la forma de mostrar los valores mediante el comando **format**:

`format long`	14 cifras significativas
`format short`	vuelve al formato corto (5 cifras significativas)
`format`	vuelve al formato por defecto (corto)
`format short e`	formato corto y notación exponencial
`format long e`	formato largo y notación exponencial
`format rat`	formato racional: aproximación en forma de fracción

EJEMPLOS

```
>> a=.0001234567
a =
       1.2346e-004
>> format long
>> a
a =
       1.234567000000000e-004
>> format rat
>> a
a =
       1/8100
```

1.5 Variables predefinidas

Algunos nombres están pre-definidos por MATLAB:

`ans`	variable del sistema para almacenar el resultado de evaluar expresiones
`i , j`	unidad imaginaria : raiz cuadrada de -1
`pi`	número π
`Inf`	"Infinito": número mayor que el más grande que se puede almacenar
`NaN`	"Not a Number : magnitud no numérica resultado de cálculo indefinidos

EJEMPLOS

```
>> 5/3
ans =
    1.6667
>> b=5/0
Warning: Divide by zero.
b =
   Inf
>> b/b
ans =
   NaN
```

1.6 Funciones matemáticas elementales

`sqrt(x)`	raiz cuadrada	`sin(x)`	seno
`abs(x)`	módulo	`cos(x)`	coseno
`conj(z)`	complejo conjugado	`tan(z)`	tangente
`real(z)`	parte real	`asin(x)`	arcoseno
`imag(z)`	parte imaginaria	`acos(x)`	arcocoseno
`angle(z)`	argumento	`atan(x)`	arcotangente
`exp(x)`	exponencial	`rats(x)`	aprox. racional
`log(x)`	logaritmo natural	`rem(x,y)`	resto de dividir x por y
`log10(x)`	logaritmo decimal	`sign(x)`	signo (1 / -1 / 0)

Los argumentos pueden ser, siempre que tenga sentido, reales o complejos y el resultado se devuelve en el mismo tipo del argumento.

1.7 Algunos comandos utilitarios y de ayuda

El principal comando de ayuda en MATLAB es **help**, que nos da una lista de tópicos sobre las que pedir ayuda. También se puede pedir ayuda directamento sobre un comando:

2. VECTORES Y MATRICES

El nombre MATLAB viene de **MAT**rix **LAB**oratory. En MATLAB todos los datos son matrices. Los vectores y escalares son casos particulares de matrices.
En MATLAB no es necesario especificar previamente las dimensiones de una matriz. Se definen de forma interactiva, al "crearla". De hecho, una vez creada, se pueden modificar sus dimensiones

2.1 Definición de vectores

Un vector-fila de dimension **n** (una matriz de dimensiones **1xn**) se puede definir en MATLAB escribiendo sus componentes entre corchetes rectos (**[ ]**) y separándolos por comas o espacios en blanco:

```
>> v=[1,-1,0,2.88]
```

La orden anterior crea en MATLAB una variable de nombre **v** que "contiene" un vector-fila de longitud 4 (una matriz 4x1).
Un vector-columna se crea igual, pero separando las componentes por "punto y coma":

```
>> w=[0;1;2;3;4;5]
```

crea una variable de nombre **w**, que "almacena" un vector-columna de longitud 6 (una matriz de dimensiones 6x1).

Otras órdenes para definir vectores son:

```
>> v1=a:h:b
```
define un vector-fila cuyas componentes van desde **a** hasta un número **c<b**, en incrementos de **h**

```
>> v2=linspace(a,b,n)
```
define un vector de longitud **n**, partición regular del intervalo **[a,b]**
(como **a:h:b**, con **h=(b-a/(n-1)**; la última componente es **=b**)

```
>> v'
```
es el vector traspuesto del vector **v** (ídem para matrices)

Las componentes de un vector se designan mediante el número de su subíndice:

```
v(1), w(4), v1(3)
```

ATENCIÓN: En MATLAB, los subindices de los vectores y matrices comienzan siempre por 1

También se puede acceder a un bloque de componentes de un vector, indicando los subíndices mínimo y máximo, o indicando un subconjunto de índices

```
>> v(2:3)
>> w(1:4)
>> ii=[2,6,21,34]; wz(ii)
```

2.2 Operaciones con vectores y escalares

Si **v** es un vector (fila o columna) y **k** es un escalar, las operaciones siguientes dan el resultado que se indica:

v+k	vector de componentes {v_i+k}
v-k	vector de componentes {v_i-k}
k*v	vector de componentes {k*v_i}
v/k	vector de componentes {v_i/k}
k./v	vector de componentes {k/v_i}
v.^k	vector de componentes {(v_i)^k}
k.^v	vector de componentes {k^(v_i)}

- Ejercicio:
- Obtener el producto interior a . b
  a=[1,2,3,4]
  b=[5,6,7,8]

Rta: a*b'

2.3 Operaciones entre vectores

Si **v** y **w** son dos vectores (fila o columna) de las mismas dimensiones:

`>> v+w`	vector de componentes $\{v_i+w_i\}$
`>> v-w`	vector de componentes $\{v_i-w_i\}$
`>> v.*w`	vector de componentes $\{v_i*w_i\}$ (producto componente a componente)
`>> v./w`	vector de componentes $\{v_i/w_i\}$ (división componente a componente)
`>> v.^w`	vector de componentes $\{v_i \wedge w_i\}$ (exponenc. componente a componente)
`>> v*w`	Si **v** es un vector-fila de dimension **n** y **w** es un vector columna de dimensión **n**, es el producto escalar de v y w

2.4 Funciones específicas de vectores

Las funciones matemáticas elementales admiten vectores como argumentos y se interpretan componente a componente.
Algunas funciones específicas para vectores son:

`sum(v)`	suma de las componentes del vector v
`prod(v)`	producto de las componentes del vector v
`dot(v,w)`	producto escalar de dos vectores del mismo tipo y las mismas dimensiones
`cross(v,w)`	producto vectorial de dos vectores del mismo tipo y dimensión 3
`max(v)`	máximo de las componentes del vector v (atención: sin valor absoluto)
`norm(v)`	norma euclídea del vector v
`norm(v,p)`	norma-p del vector v: sum(abs(v).^p)^(1/p)
`norm(v,inf)`	norma infinito del vector v

OBSERVACIÓN 2.4

Las funciones MATLAB pueden devolver un número variable de resultados. Cuando una función tiene, en su definición, más de un argumento de salida, puede ser utilizada de varias formas. La función **max** nos proporciona un ejemplo:
- si, simplemente, utilizamos la función en la forma:

```
>> max(v)
```

MATLAB nos devolverá el máximo valor de entre las componentes del vector **v**
- si la utilizamos en la forma:

```
>> [m,k]=max(v)
```

utilizando dos variables para almacenar la salida, en la variable **m** quedará almacenado el máximo de las componentes, y en la variable **k** se guardará el valor del subíndice de la componente que produce el máximo.

EJEMPLO 2.4

Si **v** es un vector-fila (respectivamente columna) de componentes v_i, entonces **exp(v)** es otro vector fila (resp. columna) de componentes **exp(**v_i**)**.

```
>> v=[1,2,4,-5,0,-1];
>> sum(v)
ans =
     1
>> max(v)
ans =
     4
>> [m,k]=max(abs(v))
m =
     5
k =
     4
>> sqrt(sum(v.^2))
ans =
   6.8557
```

- Ejercicio: Pruebe los siguientes comandos:

```
>>A=25;
save nombre_archivo
clear A
clear all
```

load nombre_archivo

- Respuestas:

save nombre_archivo	Guarda el Workspace
clear sup Borra del	Workspace la variable A
clear all	Borra todas las variables del Workspace
load nombre_archivo	Carga el Workspace previamente guardado
en el archivo nombre_archivo	

2.5 Definición de matrices

Las matrices se definen de forma similar a los vectores, introduciendo sus filas como vectores-fila y separando unas filas de otras mediante punto y coma o saltos de línea.

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

Las componentes de una matriz se designan mediante los números de sus subíndices.
Un vector-fila de dimensión **n** es en realidad una matriz de dimensión **1xn**.
Un vector-columna de dimensión **m** es en realidad una matriz de dimensión **mx1**.

EJEMPLO 2.5

```
>> A=[1,2,3 ; 4,5,6 ; 7,8,9];
>> A(1,3)
ans =
     3
>> A(1,:)           % primera fila de A
ans =
     1     2     3
>> A(:,2)           % segunda columna de A
ans =
     2
     5
     8
>> A(1:2;2:3)       % submatriz de A
ans =
     2     3
     5     6
```

2.5 Operaciones con matrices

A+B	suma de matrices
A-B	diferencia de matrices
A*B	producto de matrices (habitual)
A^2	producto de la matriz A por si misma
A\B	A^{-1} B
A/B	A B^{-1}
A.*B	producto componente a componente (a_{ij} b_{ij})
A.^2	cuadrado componente a componente (a_{ij} ^2)
A./B	división componente a componente (a_{ij} / b_{ij})
A'	transpuesta de A (la adjunta si A es compleja)

Las matrices pueden también utilizarse como argumento de las funciones intrínsecas.

EJEMPLO 2.6 (a)

```
>> diag([1,2,3,1])
ans =
     1     0     0     0
     0     2     0     0
     0     0     3     0
     0     0     0     1
>> eye(3,3)+diag([-1,-1],1)
ans =
     1    -1     0
     0     1    -1
     0     0     1
```

También se pueden definir matrices por bloques:

[A,B]	es la matriz (A B)
[A;B]	es la matriz (A B) transpuesta (columna)
[ ]	representa la matriz "vacía" (0x0)
A(:,3)=[]	elimina la tercera columna de la matriz A
A(1,:)=[]	elimina la primera fila de A

EJEMPLO 2.6 (b)

```
>> A=diag([1,2,3,4]); A(:,3)=[];
>> A
A =
     1     0     0
     0     2     0
     0     0     0
     0     0     4
>> A(1,:)=[]
A =
     0     2     0
     0     0     0
     0     0     4
```

Ejercicio:
Que hace la secuencia de operadores en MATLAB:
>> A=zeros(2);
>> B=ones(2,3);
>> C=[A,B;[1:5]]

Más ejercicios:

Definir las siguientes matrices en MATLAB:

$$A=\begin{pmatrix}1&3&5&8\\2&6&5&3\\4&1&9&7\\1&8&0&2\end{pmatrix};\ B=\begin{pmatrix}1&9&5&8\\12&5&5&9\\4&2&9&74\\0&6&0&3\end{pmatrix};\ C=\begin{pmatrix}1&9\\10&2\end{pmatrix}$$

Realizar los siguientes cálculos básicos con estas matrices:

3·A, A-7, A·B^T, A^{-1}, B^{-1}

Realizar ahora los siguientes cálculos, siendo D la submatriz de A formada por las 1ª y 3ª filas y columnas, y E la submatriz de B formada por las 2ª y 4ª filas y columnas:

D·E^T, D·C, C·E

Resolver las siguientes ecuaciones:

A·x=B, D·x=C

Siendo F la submatriz de A formada por las filas 2, 3 y 4, y G la submatriz de B formada por las columnas 1, 2 y 4, calcular:

F·G

2.8 Polinomios

En MATLAB los polinomios se identifican con el vector-fila de sus coeficientes:

p=[3,5,0,1,2] representa el polinomio $3x^4+5x^3+x+2$

`roots(p)`	Calcula las raices del polinomio p (es un vector-columna y, en general, calcula aproximaciones)
`poly(raices)`	Si raices es un vector-columna, devuelve el polinomio que tienes dichas raices. Se obtiene normalizado y puede ser de coeficientes complejos
`poly(A)`	Si A es una matriz cuadrada, es el polinomio característico
`polyval(p,x)`	Calcula el valor del polinomio p en el punto x (x puede ser un vector
`conv(p1,p2)`	Producto de los polinomios p1 y p2
`deconv(p1,p2)`	División de polinomios
`polyder(p)`	Derivada del polinomio p

EJEMPLO 2.8

```
>> p=[3,5,0,1,2];polival(p,0)
ans =
     2
>> raices=roots(p)
raices =
      -1.6394
       0.3716 + 0.6243 i
       0.3716 - 0.6243 i
      -0.7704
```

2.9 Sistemas lineales

`det(A)`	Calcula el determinante de la matriz A
`inv(A)`	Calcula la inversa de la matriz cuadrada A
`rank(A)`	Calcula el rango de la matriz A

A\b	Calcula la solución del sistema lineal Ax=b. Previamente a los cálculos, MATLAB realiza un análisis de la matriz A para decidir cual es el método más adecuado: triangular, Cholesky, LU, QR, Gauss, etc., y además lleva a cabo un pre-ordenamiento si A es hueca.

Ejercicios en MATLAB:

Sistemas de ecuaciones:

a) Sean las matrices $A=\begin{pmatrix} 1 & 2 & -2 & 0 \\ 2 & 4 & -1 & 0 \\ -3 & -6 & 12 & 2 \\ 1 & 2 & -2 & -4 \end{pmatrix}$ y $b=\begin{pmatrix} 1 \\ -4 \\ -12 \\ 3 \end{pmatrix}$. Comprobar que el sistema lineal cuya matriz ampliada es $(A\ b)$ no tiene solución.