Curso basico de MATLAB

5. Variables, constantes y formatos disponibles

5.1 Manejo de variables

En ***Matlab*** como en cualquier otro lenguaje de programación, y/o asistente matemático se utilizan variables. En ***Matlab*** todas las variables son matrices. Incluso un valor escalar es una matriz de dimensión 1x1. Las variables deben tener un nombre según ciertas reglas. Estas reglas son:

- NO pueden comenzar con un número, aunque si pueden tener números y/o algunos caracteres especiales (variable_1 es un nombre de variable válido).
- Las mayúsculas y minúsculas se diferencian en los nombres de variables por trabajar con C nativo. (A y a son dos variables diferentes)
- Los nombres de variables no pueden contener operadores ni puntos. (No es válido usar /, *, -, +, ...)
- Si se trabaja con complejos sólo puede utilizarse un de los nombres i y/o j para variables.
- No es necesario definir el tipo de variable o tamaño (si se usa un vector y después se expande, no hay problema)
- ***ans*** es la variable por omisión provista por ***Matlab.***
- ***Matlab*** realiza la asignación de memoria a variables durante la ejecución.

```
» x=3                x es de tipo real
» x='mensaje'        x es de tipo literal (use comillas simples ALT-39)
» syms x             x es un símbolo
» x=[2 7 4]          x es un vector
» x=2+3i             x es de tipo complejo
```

Si se omite el nombre de la variable y el signo "=", ***Matlab*** automáticamente crea la variable ***ans*** para guardar el resultado. Todos los nombres de funciones deben estar dados en letras minúsculas.

5.2 Expresiones y variables

Matlab es un lenguaje basado en expresiones formadas por variables, operadores y funciones. Una sentencia es la asignación de la evaluación de una expresión a una variable (variable = expresión). Tras interpretar y evaluar expresión, el resultado se visualiza en pantalla y se asigna a la variable. Si se omite variable, el resultado es asignado a la variable por defecto ***ans (answer)***. Se debe tener en cuenta lo siguiente:

- Una sentencia termina con un retorno de carro.
- Si una sentencia ocupa más de una línea se puede continuar en la siguiente siempre que terminemos la línea con tres o más puntos (...) seguidos de un retorno de carro.
- Es posible escribir varias sentencias en la misma línea siempre que se separen por una coma (,) o un punto y coma (;).
- Si no se quiere que el resultado de una expresión se visualice en pantalla se debe terminar la sentencia correspondiente con punto y coma (;).
- Por defecto, ***Matlab*** diferencia las mayúsculas de las minúsculas. Por ejemplo, la variable ***temp*** no es igual que la variable ***Temp***. Sin embargo, es posible modificar esta opción de modo que no se produzca distinción entre mayúsculas y minúsculas.

Una ***variable*** es un nombre que se da a una entidad numérica, que puede ser una matriz, un vector o un escalar. El valor de esa variable, e incluso el tipo de entidad numérica que representa, puede cambiar a lo largo de una sesión de ***Matlab*** o a lo largo de la ejecución de un programa. La forma más normal de cambiar el valor de una variable es colocándola a la izquierda del *operador de asignación* (=).

Una expresión en ***Matlab***, puede ser:

- Una variable o un número. ***Ej: variable1, x, 3, 22.3***
- Un comando aplicado. ***Ej: norm(A), sin(2*pi)***
- Una expresión matemática. ***Ej: 2+3*variable1^ 4.5***

Si cualquiera de las anteriores se escribe en la línea de comandos o ***prompt (»)*** del ***Matlab***, este devolverá el nombre de la variable y su valor (en caso de que la expresión tenga nombre, de no tenerlo, ***Matlab*** devolverá ***ans = resultado***).
Ya que ***Matlab*** se basa en el álgebra de matrices, estas pueden estar formadas por un sólo elemento (***escalar***), por una fila o una columna (***vector***) o por una serie de filas y columnas (***matriz***).

5.3 Asignación de escalares, vectores y matrices

- ***Asignación de un escalar real:***

$a \leftarrow 3.15$

```
» a = 3.15
```

- ***Asignación de un escalar complejo***:

$a \leftarrow 3.15+i5.8$

```
» a = 3.15+5.8i
» a = 3.15+5.8*i
» a = 3.15+i*5.8
```

- ***Asignación de un vector fila:***

$a \leftarrow [3\ 5\ 8]$

```
» a = [3 5 8]
» a = [3, 5, 8]
```

- ***Asignación de un vector columna:***

$$b \leftarrow \begin{pmatrix} 3 \\ 5 \\ 8 \end{pmatrix}$$

```
» b = [3; 5; 8]
```

- ***Asignación de una matriz:***

$$c \leftarrow \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

```
» c = [1, 2, 3; 4, 5, 6; 7, 8, 9]
```

5.4 Precisión a utilizar

Matlab representa los resultados con exactitud, pero aunque internamente siempre trabaja con cálculos exactos para no arrastrar errores de redondeo, pueden habilitarse diferentes formatos de representación aproximada, que en ocasiones facilitan la interpretación de los resultados. A continuación se citan los comandos que permiten aproximaciones numéricas. Formatos de salida, tomando el valor 4/3 como ejemplo

- format short 1.3333
- format short e 1.3333e+000
- format long 1.33333333333333
- format long e 1.33333333333333e00
- format bank 1.33
- format hex 3ff5555555555555
- format rat 4/3

format hex	Ofrece los resultados en el sistema hexadecimal
vpa ('operaciones',n)	Ofrece el resultado de las operaciones con n dígitos decimales exactos
numeric('expr')	Ofrece el valor de la expresión de forma numérica aproximada según el formato actual activo.
digits(n)	Ofrece los resultados con n dígitos exactos.

5.5 Constantes disponibles

pi: Número pi: 3'1415926535897...
» pi
ans=
3.1416

i ó j: Unidad imaginaria (raíz cuadrada de-1)
» i
ans =
0 + 1.0000i

Matlab mantiene una forma especial para los *números muy grandes* (más grandes que los que es capaz de representar), que son considerados como ***infinito***. ***Inf*** es el símbolo ∞. Así pues, para ***Matlab*** el ***infinito*** se representa como ***inf*** ó ***Inf***.
Ejemplo:
>> 1/0
ans =
Inf
Matlab tiene también una representación especial para los resultados que no están definidos como números. ***NaN*** significa ***"Not A Number"*** (valor indeterminado)
Ejemplo:
>> 0/0
ans =
NaN

Este tipo de respuesta, así como la de ***Inf***, son enormemente importantes en ***Matlab*** pues permiten controlar la fiabilidad de los resultados de los cálculos matriciales. Los ***NaN*** se propagan al realizar con ellos cualquier operación aritmética, en el sentido de que, por ejemplo, cualquier número sumado a un ***NaN*** da otro ***NaN***. ***Matlab*** tiene esto en cuenta. Algo parecido sucede con los ***Inf***.

ans: ***Variable creada automáticamente para representar el último resultado procesado al que no se le ha asignado previamente ninguna variable.***

realmin: ***El menor número real positivo utilizable***

```
» realmin
ans =
2.2251e-308
```

realmax: ***El mayor número real positivo utilizable***

```
» realmax
ans =
1.7977e+308
```

eps: ***Variable permanente cuyo valor es inicialmente la distancia desde 1.0 al siguiente número en coma flotante más elevado. Se trata de la tolerancia por defecto para operaciones en coma flotante (precisión relativa en punto flotante). En máquinas actuales (máquinas IEEE) su valor es 2^(-52)***

```
» eps
ans =
2.2204e-016
```

La forma como se presenta cada uno de las constantes anteriores depende del formato que haya configurado al ***Matlab***.

6. Operadores matemáticos

6.1 Operadores matemáticos básicos

\+ Suma de escalares, vectores o matrices. En el caso matricial, suma elemento a elemento, pero las matrices deben ser de la misma dimensión.

\- Resta de escalares, vectores o matrices. Idem suma matricial.

* Producto de escalares o producto matricial. En el caso matricial, las matrices deben ser compatibles.

.* Producto de escalares o multiplicación punto a punto (elemento por elemento) entre dos vectores o matrices. Realiza la multiplicación término a término de los elementos de dos vectores o matrices de igual dimensión.

/ Cociente de escalares o cociente matricial. Como bien sabemos si queremos hacer B/A (siendo A y B matrices), debemos realizar el producto matricial de B por la inversa de la matriz A, es decir B*inv(A). Es por ello que por ejemplo cuando en ***Matlab*** escribamos B / A significara B*inv(A). Las matrices deben ser compatibles.

./ Cociente de escalares o cociente punto a punto (elemento por elemento) entre dos vectores o matrices. Realiza el cociente término a término de los elementos de dos vectores o matrices de igual dimensión.

^ Potencia de escalares o potencia matricial. Por ejemplo B^2 equivale a B*B.

.^ Potencia de escalares o potencia de cada elemento de un vector o matriz.

\ Operador barra invertida. Equivale a premultiplicar por la inversa de esa matriz. Por ejemplo A\B = inv(A)*B, siendo A y B matrices compatibles.

.\ A.\B cociente elemental de B entre A siendo A y B matrices de igual dimensión.

' Transposición de vectores o matrices.

6.2 Ejemplos de uso de los operadores matemáticos básicos

Siendo

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

Suma de matrices:

```
>> S = A+B
S =
     10 10 10
     10 10 10
     10 10 10
```

Resta de matrices:

```
>> S1 = A-B
S1 =
     -8 -6 -4
     -2  0  2
      4  6  8
```

Producto de matrices:

```
>> Z = A*B
Z =
      30   24  18
      84   69  54
     138 114  90
```

Producto de matrices (elemento por elemento):

```
>> Z1 = A.*B
Z1 =
     9  16  21
    24  25  24
    21  16   9
```

Suma o resta de una matriz con un escalar:

Siendo

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

Hacemos:

```
>> T = A-4
T =
    -3 -2
    -1  0
```

Producto de un vector por un escalar:

Siendo

```
>> a = [2;1;2];
>> b = [1;2;3];
```

Hacemos

```
>> c = a*3
c =
     6
     3
     6

>> c = b.*3
c =
     3
     6
     9
```

Producto de vectores (elemento por elemento):

Siendo

```
>> a = [2 3 5];
>> b = [4 3 6];
```

Hacemos

```
>> c = a.*b
c =
     8 9 30
```

Cociente de un vector por un escalar:

Siendo

```
>> a = [2;1;2];
>> b = [1;2;3];
```

Hacemos

```
>> d = a/3
d =
        0.6667
        0.3333
        0.6667

>> d = a./3
d =
        0.6667
        0.3333
        0.6667
```

Potenciación con un escalar:

Siendo

```
>> a = [2;1;2];
>> b = [1;2;3];
```

Hacemos

```
>> x = a.^2
>> x =
        4
        1
        4

>> x = 2.^a
>> x =
        4
        2
        4
```

Potenciación de matrices (elemento por elemento):

Siendo

```
>> a = [2;1;2];
>> b = [1;2;3];
```

Hacemos

```
>> u = a.^b
>> u=
        2
        1
        8
```

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

Hacemos
>> C = A^2 que es lo mismo que A*A
C =
30 36 42
66 81 96
102 126 150

Transposición de un vector:
Siendo
>> a = [2;1;2];
a =
2
1
2

Hacemos
>> z = a'
z =
2 1 2

6.3 Las funciones DOT, CROOSS e INV

DOT
Calcula el producto escalar de los vectores A y B. A y B deben ser vectores de la misma longitud. La sintaxis de la orden es:

C = DOT(A,B)

*Cuando A y B son ambos vectores columna o vectores fila, DOT(A,B) es lo mismo que A'*B ó A*B'*

CROSS
Calcula el producto cruz o producto vectorial entre dos vectores. Los vectores deben ser de 3 elementos y la sintaxis de la orden es:

Vector1 = cross(Vector2, Vector 3)

Vector2 y Vector3 son los vectores a los que se les quiere aplicar el producto cruz.
Vector1 es el vector resultante del producto cruz de Vector2 y Vector3.
El siguiente ejemplo ilustra el uso de cross:
x = [1 0 0];
y = [0 1 0];
z = cross(x, y)
z =
0 0 1

INV
Calcula la inversa de una matriz siempre que esta sea cuadrada. La sintaxis de la orden es:
La sintaxis de la orden es:

C = inv(A)

6.4 Calculo de sistema de ecuaciones lineales

Supongamos que tenemos el siguiente sistema:

$$2x + 3y + z + k = 2$$
$$x + 4y + 2z + 9k = 3$$
$$3x - y + z - k = 0$$
$$5x + 2y - 3z + k = 10$$

Se escribe las matrices

```
» A = [2 3 1 1;1 4 2 9;3 -1 1 -1;5 2 -3 1]; ↵
» C = [2 3 0 10]; ↵
» X = inv(A)*C'; ↵
```

O ingresando el vector en forma de columna de la siguiente manera:

```
» A = [2 3 1 1;1 4 2 9;3 -1 1 -1;5 2 -3 1]; ↵
» C = [2; 3; 0; 10]; ↵
» X = inv(A)*C; ↵
```

Pruebe ahora con X = A\C'↵

La función *linsolve* es la forma más eficiente de que dispone MATLAB para resolver sistemas de ecuaciones lineales. A diferencia del operador barra invertida \, esta función no trata de averiguar las características de la matriz que permitan hacer una resolución más eficiente: se fía de lo que le dice el usuario. Si éste se equivoca, se obtendrá un resultado incorrecto sin ningún mensaje de error.

La forma general de la función *linsolve* para resolver Ax=C :

```
» x = linsolve(A,C') ↵
```

7. Funciones matemáticas elementales

Estas funciones, que comprenden las funciones matemáticas trascendentales y otras funciones básicas, actúan sobre cada elemento de la matriz como si se tratara de un escalar. Se aplican de la misma forma a escalares, vectores y matrices. Algunas de las funciones de este grupo son las siguientes.
La librería ***Matlab*** dispone de una gama muy completa de funciones predefinidas que se corresponden con las funciones matemáticas más utilizadas.

7.1 Funciones trigonometricas e hiperbolicas

sin(Z)	asin(Z)	sinh(Z)	asinh(Z)
cos(Z)	acos(Z)	cosh(Z)	acosh(Z)
tan(Z)	atan(Z)	tanh(Z)	atanh(Z)
atan2(Z)			
sec(Z)	asec(Z)	sech(Z)	asech(Z)
csc(Z)	acsc(Z)	csch(Z)	acsch(Z)
cot(Z)	acot(Z)	coth(Z)	acoth(Z)

7.2 Funciones exponenciales

log(x)	Logaritmo natural
log10(x)	Logaritmo decimal
exp(x)	Función exponencial
sqrt(x)	Raíz cuadrada

7.3 Funciones específicas de variable numérica

gcd(x)	Máximo común divisor
lcm(x)	Mínimo común múltiplo
ceil(z)	Redondea los decimales al mayor entero más próximo. Si z es un número complejo, la aplica tanto a la parte real como a la imaginaria.
floor(z)	Redondea los decimales al menor entero más próximo. Si z es un número complejo, la aplica tanto a la parte real como a la imaginaria.
round(z)	Redondea al entero más próximo. Si z es un número complejo, la aplica tanto a la parte real como a la imaginaria.
fix(z)	Elimina la parte decimal de z. Si z es un número complejo, la aplica tanto a la parte real como a la imaginaria.
real(z)	Parte real de z.
imag(z)	Parte imaginaria de z.
conj(z)	Complejo conjugado de z.
abs(z)	Valor absoluto de de z. Si z es un número complejo, calcula el modulo.
angle(z)	Angulo de fase del numero complejo z.
sign(z)	Función signo. Si z es real, devuelve -1 si <0, 0 si =0 y 1 si >0. Si z es un número complejo, devuelve z/abs(z).
rem(x,y)	Da el resto de la división entre los reales x e y, haciendo fix(x./y)
mod(x,y)	Da el resto de la división entre los reales x e y, haciendo floor(x./y)