Introducción a MATLAB

Sistemas Conexionistas - Curso 07/08

MATLAB es un sistema interactivo basado en matrices para cálculos científicos y de ingeniería. Se pueden resolver problemas numéricos relativamente complejos sin escribir un programa en realidad. El nombre MATLAB es la abreviatura de MATrix LABoratory.

1. Comandos básicos

Matlab puede usarse como una calculadora. Los cálculos que no se asignan a una variable en concreto se asignan a la variable de respuesta por defecto que es ans (del inglés, answer):

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

Sin embargo, si el cálculo se asigna a una variable, el resultado se almacena en ella:

```
>>x=2+3
x =
5
```

Para conocer el valor de una variable, basta teclear su nombre:

```
>>x
x =
5
```

Si se añade un punto y coma (;) al final de la instrucción, se realiza el cálculo pero no se muestra la respuesta.
Las operaciones se evalúan por orden de prioridad: primero las potencias (^), después las multiplicaciones y divisiones y, finalmente, las sumas y restas. Las operaciones de igual prioridad se evalúan de izquierda a derecha:

```
>>2/4*3
ans =
1.5000

>>2/(4*3)
ans =
0.1667
```

Se pueden utilizar las funciones matemáticas habituales:

```
>>cos(pi) % pi es una variable con valor predeterminado 3.14159...
ans =
-1

>>exp(1)    % Función exponencial evaluada en 1, es decir, el número e
ans =
2.7183
```

Para conocer las variables que se han usado hasta el momento:

```
>>who
Your variables are:
ans x
```

o, si se quiere más información (obsérvese que todas las variables son arrays):

```
>>whos
Name  Size  Bytes      Class
 ans   1x1    8     double array
   x   1x1    8     double array
Grand total is 2 elements using 16 bytes
```

Para eliminar todas las variables del workspace se utiliza el comando `clear`. Si se desea eliminar sólo una variable:

```
>>clear x
>>who
Your variables are:
ans
```

Para más informacion, consultar la ayuda del programa (`help ops`).
Notas:

- Matlab distingue mayúsculas de minúsculas
- Los comentarios van precedidos por %
- La ayuda de un comando se invoca con `help <comando>`
- La documentación de un comando se invoca con `doc <comando>`
- Con `doc` se accede a la documentación general de Matlab

2. Matrices y vectores

Para crear una matriz basta con escribir sus elementos entre corchetes, separando las filas por puntos y comas (;):

```
>> M = [ 1 2 3; 4 5 6 ]
M =
     1     2     3
     4     5     6
```

Para acceder a un elemento de la matriz se indica el nombre de la matriz seguido de la fila y la columna a la que deseamos acceder:

```
>> M(2,3)
M =
6
```

El operador dos puntos (:) es el operador de rango. `J:K` es equivalente a `[J, J+1, ..., K]` siempre que `J` $<$ `K` y `J:D:K` es equivalente a `[J, J+D, J + 2 * D, ..., K]`. El operador dos puntos también se puede utilizar para seleccionar filas, columnas o partes de una matriz. Con el operador ' se realiza la transposición de matrices:

```
>> M'
ans =
     1     4
     2     5
     3     6
```

Matlab permite aplicar una función a todos los elementos de una matriz y además realizar operaciones entre los elementos de matrices situados en la misma posición. Para esto último, se antepone un punto (.) al operador multiplicación, división o potencia:

```
>> N = [10 20 30; 40 50 60];
>> N ./ M
ans =
    10    10    10
    10    10    10
>> N / 10
ans =
     1     2     3
     4     5     6
```

Matlab dispone de una serie de funciones para trabajar con matrices tales como `ones`, `zeros`, `size`, `length`, `max`, `min`, `minmax`, `rand`, `inv`, `det`, `sum`, ... Para más información, consultar la ayuda del programa (`help elmat`).

3. Lenguaje de programación

3.1. Sentencias

If

```
IF expresión
  sentencias
ELSEIF expresión
  sentencias
ELSE
  sentencias
END
```

```
if x < 0
  x = -x;
elseif x > 0
  x = x * 2;
else
  x = 0;
end
```

For

```
FOR variable = expresión,
 sentencias
END;
```

```
for i = 0:2*pi,
  for j = 0:0.1:2*pi,
    a = a + 2 * cos(i) + sin(j);
  end
end
```

While

```
WHILE expresión,
 sentencias
END
```

```
while (i ~= 0),
   x = x * x;
   i = i - 1;
end
```

Switch

```
SWITCH expresión
 CASE caso1,
   sentencias
 CASE {caso2, caso3,...}
   sentencias
 OTHERWISE
   sentencias
END
```

```
switch x
  case 1,
     f = funcion1(x);
  case 2,
     f = funcion2(x);
  otherwise
     f = funcion1(x);
end
```

3.2. Funciones y scripts

Los *scripts* permiten al usuario escribir un conjunto de órdenes en un editor de texto (incluido en el programa Matlab) para ejecutarlas posteriormente en el orden en el que se han escrito. En lugar de introducir las órdenes una a una en el command window o ventana de comandos las órdenes se escriben una tras otra en un fichero con extensión .m. Un *script* tiene acceso a las variables definidas en el workspace. Además, las variables creadas mediante el *script* son globales, es decir, se mantienen en el workspace tras la ejecución del *script*.

Otra forma de crear programas en Matlab es a través de funciones, almacenadas también en un fichero .m. Una función tiene un nombre, se le pasan 0 o varios parámetros y puede devolver uno o varios resultados:

```
% Función propia 1
function resultado = f (x, y)
% variables de entrada: x e y
% variable de salida: resultado
resultado = (x * 2) + (y * 3)+ sqrt((x + y) /2);
```

Una vez está definida la función, ésta se ejecuta si tecleamos en el command window (o escribimos en otra función o script):

```
>> a = f (c, d) ;
```

Una función no puede acceder a las variables definidas en el workspace. Además, las variables que utiliza son locales, esto es, desaparecen una vez finaliza la llamada a la función. Para ejecutar una función o un script, es necesario que el directorio de trabajo coincida con el directorio donde se encuentra el script o la función.

4. Ejercicios

4.1. Vectores y matrices

1. Crea un vector V1 con enteros desde 1 hasta 20
2. Crea un vector V2 con números de 0 a 4, espaciados 0,2 unidades
3. Halla el seno de todos los elementos del vector V2
4. Define las siguientes matrices

$$M1 = \begin{bmatrix} 1 & 4 & 6 \\ 3 & 4 & 5 \\ 9 & 10 & 2 \\ 8 & 3 & 7 \end{bmatrix}, \; M2 = \begin{bmatrix} 3 & 4 & 2 \\ 4 & 6 & 1 \\ 7 & 3 & 4 \\ 5 & 1 & 5 \end{bmatrix}$$

y realiza las siguientes operaciones:

a) Suma las matrices M1 y M2

b) Multiplica M1 por 3

c) Multiplica cada elemento de M1 por el elemento situado en la misma posición en M2

d) Obtén el tamaño de la matriz M1

e) Obtén el número de filas de la matriz M2

f) Accede al elemento situado en la fila 4 y columna 1 de la matriz M2

g) Muestra la tercera fila de la matriz M2

h) Muestra la segunda columna de la matriz M1

i) Obtén la matriz transpuesta de M1

j) Crea una submatriz SM1 que abarque las tres primeras filas y las dos últimas columnas de M1

k) Obtén el valor máximo y el valor mínimo de la matriz M1

5. Multiplica las siguientes matrices

$$M3 = \begin{bmatrix} 5 & 3 & 1 \\ 2 & 7 & 5 \end{bmatrix}, \; M4 = \begin{bmatrix} 1 & 2 \\ 2 & 5 \\ 3 & 5 \end{bmatrix}$$

6. Crea una matriz M5 de 3 filas y 2 columnas formada por ceros

7. Crea una matriz M6 de 4 filas y columnas formada por unos

8. Crea una matriz M7 de 2 filas y 3 columnas formada por valores aleatorios

4.2. Programación

1. El uso de bucles es común en la mayoría de los lenguajes de programación. En Matlab, muchos bucles consistentes en aplicar una función a un conjunto de datos pueden reescribirse como operaciones sobre matrices, incrementando así la eficiencia del proceso.

   Convierte los siguientes bucles a operaciones matriciales:

   a)
   ```
   M = [1 2 3; 4 5 6; 7 8 9];
   for i = 1:size(M,1)
       for j = 1:size(M,2)
           N(i,j) = M(i,j) * M(j,i);
       end
   end
   ```

   b)
   ```
   j = 1;
   for i = 0:0.1:pi
   ```

```
        y(j) = sin(2 * i);
        j = j + 1;
    end
```

c)
```
M = [ 1 2; 3 4 ; 5 6; 7 8; 9 0];
for i = 1:2:size(M,1)
    for j = 1:size(M,2)
        N(floor(i/2) + 1,j) =  M(i,j) * 2;
    end
end
```

2. Crea un script que cree dos matrices del mismo tamaño, las sume y divida el resultado por 2. Ejecuta el script y comprueba el estado del workspace antes y después de la ejecución.

3. Implementa una función llamada `stat` que devuelva la media, el máximo y el mínimo de un vector pasado como parámetro. Crea un vector, aplícale la función y comprueba el estado del workspace tras la llamada a la función.