Capítulo 4

Introducción a la programación en MATLAB

4.1. Operadores relacionales y lógicos

4.1.1. Operadores relacionales y lógicos

Una condición lógica es una expresión determinada mediante operadores relacionales y lógicos que puede ser *verdadera*, en cuyo caso tomará el valor 1, o *falsa*, en cuyo caso tomará el valor 0.

Operadores relacionales :

Son los que permiten comparar datos y cuyos resultados son valores lógicos, es decir, **1**(verdadero)o **0**(falso)

Operador	Descripción
<	Menor
<=	Menor o igual
>	Mayor
>=	Mayor o igual
==	Igual
~=	Distinto

El *símbolo* $\sim$ se obtiene normalmente pulsando simultáneamente Alt Gr y 4, o bien manteniendo pulsada la tecla Alt pulsar, en ese orden, 126. El 126 debe marcarse con el teclado numérico.

Es importante distinguir el *símbolo* = de asignación de un valor a una variable, del símbolo == que compara el valor de dos variables.

Veamos unos ejemplos. Si MATLAB devuelve un 1, la expresión es cierta; si devuelve un 0, es falsa.

```
>> 1>2
>> x=3, x==3
```

```
>> x>3
>> sin(pi/2)==1
>> cos(pi/2)==0 % Explica que ha pasado en este caso!
```

También se pueden comparar matrices elemento a elemento, siempre que tengan el mismo tamaño

```
>>  [1 2 3]==[1 2 5]
ans =
     1     1     0
```

e, incluso, matrices con escalares

```
>> [1 2;1 3]>1 % Compara cada elemento de la matriz con 1
ans =
     0     1
     0     1
```

4.1.2. Operadores lógicos

Permiten relacionar varias expresiones lógicas entre sí usando, entre otros, los operadores `not`, `and`, `or`.

Operador	Forma corta	Descripción
`not(p)`	`~(p)`	Negación: toma el valor de verdad 1 cuando la expresión `p` toma el 0 y viceversa
`and(p,q)`	`(p) & (q)`	Y lógico: Sólo toma el valor 1 cuando `p` y `q` lo toman simultáneamente. 0 en los otros tres casos
`or(p,q)`	`(p) | (q)`	O lógico (inclusivo): Sólo toma el valor 0 cuando `p` y `q` lo toman simultáneamente. 1 en los otros tres casos

Ejercicio 4.1 *Completa la siguiente tabla y comprueba con* MATLAB *los valores lógicos obtenidos. Observa la importancia del uso de los paréntesis y el orden de precedencia de los operadores.*

Expresión Lógica	*Descripción*	*Valor lógico*
`~(1>2)`	*No se verifica que 1 es mayor que 2*	*1*
`(1>=2)& (0==0)`		
A=eye(5);b=ones(5,1); `(rank(A)==5) & (rank(A)==rank([A,b]))`		
niter=101;err=1e-6; `~((niter<100) &(err>1e-8))`		
niter=101;err=1e-6; `((niter>=100) |(err<=1e-8))`		
`1==2 & 2==2	3==3`		
`1==2 &(2==2 | 3==3)`		

4.2. Estructuras de control

En los programas de MATLAB se manejan principalmente tres estructuras de control:

- Decisión: `if ... elseif ... else ... end`
- Repetición un número fijo de veces: `for ... end`
- Repetición bajo condiciones: `while ... end`

Se puede, pero no es habitual, utilizar estas estructuras en la ventana de trabajo de MATLAB. Estudiaremos solamente las dos primeras.

4.2.1. Condicional

Si queremos ejecutar un conjunto de instrucciones en el caso de que se cumpla una condición, usaremos una estructura `if`.

- La manera más sencilla de usarla es la siguiente:

if *expresión lógica 1*
conjunto de ordenes 1 (de Matlab)
end

El conjunto de ordenes 1 se ejecuta si la expresión lógica 1 es verdadera.

Ejemplo 4.2.1 *Cree un programa que al introducir el número del carnet diga si se ha aprobado un examen en caso afirmativo o no conteste en caso contrario. Suponga que los números de carnet de los alumnos aprobados son 34000001; 34000002; 34000003.*

Solución

```
dni=input('Introduce tu numero de carnet dni=' );
a1=34000001;a2=34000002;a3=34000003;
if (dni== a1)|(dni== a2 )|(dni== a3 )
    disp('aprobado')
end
```

Observa que si proponemos como dato un número que no se corresponda con el de los aprobados el programa no devuelve ninguna respuesta.

Ejemplo 4.2.2 *Cree un programa, de nombre* **sistemacramer1**, *que pida la matriz de los coeficientes y la de los términos independientes de un sistema del mismo número de ecuaciones que de incógnitas y en el caso de ser de Cramer lo resuelva.*

Solución

```
% Uso de la estructura if-end
A=input('Matriz cuadrada de los coeficiente del sistema A=');
b=input('Introduce la matriz columna de los términos independientes b=');
if det(A)~=0
   x=A\b
end
```

Observa que si proponemos como dato una matriz de determinante nulo el programa no devuelve ninguna respuesta.

- Necesitamos una manera de introducir una *alternativa* cuando no se cumpla la condición. Esto se hace mediante el comando **else**. La estructura habitual para este comando es:

  if *expresión lógica 1*
  conjunto de ordenes 1 (de Matlab)
  else
  conjunto de ordenes (de Matlab) si la expresión 1 es falsa
  end

  Vamos a modificar los programas anteriores para que den un mensaje alternativo.

Ejemplo 4.2.3 *Cree un programa que al introducir el número del carnet diga si un alumno ha aprobado un examen o no (bien porque haya suspendido, bien porque no se haya presentado o bien porque no se haya matriculado).*

Solución

```
% Uso de la estructura if-else-end
dni=input('Introduce tu numero de carnet dni=' );
a1=34000001;a2=34000002;a3=34000003;
if (dni== a1)|(dni== a2 )|(dni== a3 )
   disp('aprobado')
else
   disp('no aprobado')
end
```

Ejemplo 4.2.4 *Cree un programa, de nombre* **sistemacramer2**, *que pida la matriz de los coeficientes y la de los términos independientes de un sistema del mismo número de ecuaciones que de incógnitas, en el caso de ser de Cramer devuelva la solución y en caso contrario un mensaje.*

Solución

```
% Uso de la estructura if-else-end
A=input('Matriz cuadrada de los coeficiente del sistema A=');
b=input('Introduce la matriz columna de los términos independientes b=');
if det(A)~=0
   x=A\b
else
   disp('El sistema no es de Cramer')
end
```

Ejemplo 4.2.5 *Constrúyase un fichero* ***.m*** *que dados los sistemas generadores de dos subespacios V_1 y V_2 y un vector v de $\mathbb{R}^n$ nos diga si v pertenece o no a los subespacios V_1, V_2 y V_1+V_2. Utilícese el fichero creado para determinar si el vector $v = (0,1,1,1)$ pertenece o no a los subespacios $V1 =< (1,2,1,0),(-1,1,1,1) >$, $V2 =< (2,-1,0,1),(1,-1,3,7) >$ y $V_1 + V_2$.*

Solución

Para construir este programa nos basaremos en el hecho de que dados $v, v_1, ..., v_m$ vectores de $\mathbb{R}^n$, $v \in < v_1, ..., v_m > \Longleftrightarrow dim < v_1, ..., v_m >= dim < v_1, ..., v_m, v >$

En el fichero **pertenece** escribimos:

```
%Indica la pertenencia de un vector a varios subespacios dados.
A1=input('Matriz cuyas filas generan el subespacio V1: A1=');
A2=input('Matriz cuyas filas generan el subespacio V2: A2=');
A3=[A1;A2];% A3 es la matriz cuyas filas generan V1+V2 .
v=input('Dado el vector fila v=');
if rank([A1;v])==rank(A1)
     disp('v pertenece a V1')
else
     disp('v no pertenece a V1')
end
```

```
if rank([A2;v])==rank(A2)
          disp('v pertenece a V2')
else
          disp('v no pertenece a V2')
end
if rank([A3;v])==rank(A3)
         disp('v pertenece a V1+V2')
else
   disp('v no pertenece a V1+V2')
end
```

La ejecución en la ventana de comandos sería la siguiente:

```
>> pertenece
Matriz cuyas filas generan el subespacio V1: A1=[1 2 1 0;-1 1 1 1]
Matriz cuyas filas generan el subespacio V2: A2=[2 -1 0 1;1 -1 3 7]
Dado el vector v=[0 1 1 1]
v no pertenece a V1
v no pertenece a V2
v pertenece a V1+V2
```

- Para introducir más alternativas, utilizaremos la orden `elseif`. Con ella se pueden incluir consecutivamente varias condiciones de modo que sólo se ejecutará el conjunto de órdenes correspondientes a la primera condición que se cumpla.

  if *expresión lógica 1*
  conjunto de ordenes 1 (de Matlab)
  elseif *expresión lógica 2*
  conjunto de ordenes 2 (de Matlab) si la expresión 2 es verdadera y la anterior es falsa
  ⋮ ⋮

  else
  conjunto de ordenes (de Matlab) si las expresiones anteriores son falsas
  end

Ejemplo 4.2.6 *Cree un programa que al introducir el número del carnet diga si el alumno ha aprobado un examen, si ha suspendido, si no se ha presentado o si no se ha matriculado. Suponga que dentro de los alumnos matriculados aquéllos con número de carnet 34000001, 34000002 o 34000003 han aprobado ; el de número 34000004 ha suspendido y el de número 34000005 no se ha presentado.*

Solución

```
% Uso de la estructura if-elseif-...-else-end
dni=input('Introduce tu numero de carnet dni=' );
a1=34000001;a2=34000002; a3=34000003;a4=34000004;a5=34000005;
if (dni== a1 )|(dni== a2 )|(dni== a3 )
   disp('aprobado')
```

```
 elseif (dni== a4 )
    disp('suspenso')
 elseif (dni== a5 )
    disp('no presentado')
 else
     disp('no matriculado')
 end
```

Ejemplo 4.2.7 *Basándose en el Teorema de Rouché-Fröbenius, cree un programa que pida la matriz de los coeficientes y el vector de los términos independientes de un sistema, clasifique el sistema y devuelva :*

1. *La solución única cuando el sistema sea compatible y determinado.*
2. *La reducida escalonada de la matriz ampliada, el vector $A \setminus b$, una base del conjunto de soluciones del sistema homogéneo asociado y un mensaje indicando como será la solución general, cuando sea compatible indeterminado.*

Utilícese para clasificar y resolver el siguiente sistema:

$$\begin{cases} x+2y & -z & +t & +u & = & 0 \\ 3x- & y & +t & -u & = & 6 \\ 6x+ & y & +t & +u & = & 1 \\ x-2y & +2z & -2t & & = & -5 \end{cases}$$

En el fichero **Rouche** *escribimos:*

```
%Si el tamaño de las matrices no es correcto detenemos el programa
A=input('Matriz de los coeficientes de un sistema  lineal, A=');
b=input('Vector correspondiente a los términos independientes b=');
if ~(size(A,1)==size(b,1) & size(b,2)==1)
    error('El tamaño de las matrices no es correcto')
end
 if rank(A)==rank([A,b]) & rank(A)==size(A,2)
       disp( 'sistema compatible determinado')
       E=rref([A b]);r=rank(A);X=E(1:r,end);  disp('Solución única:'),disp(X)

 elseif rank(A)==rank([A,b])
      disp('sistema compatible indeterminado')
BaseHomogeneo=null(A),G=rref([A b]),Xp=A\b %a veces no puede calcularse;
 disp('Solución general=Solución particular + Solución general del homogéneo')

else
         warning('sistema incompatible')
end
```

El cuadro siguiente muestra algunas órdenes que se pueden utilizar para sacar mensajes en pantalla

Función	Salida
disp('mensaje')	presenta la variable mensaje en la pantalla
error('mensaje')	presenta la variable mensaje en la pantalla y detiene la ejecución del programa
warning('mensaje')	presenta la variable mensaje en la pantalla al igual que la orden disp . Además con este tipo de mensajes también podemos detener la ejecución del programa, acceder directamente al editor desde Command Window, etc. (Veáse la opción debug del menu de MatLab)

Ejercicio 4.2 *En* $\mathbb{R}^4$ *se considera el subespacio* $V = \langle (1,0,1,1) \rangle$. *Constrúyase un fichero que dados tres vectores cualesquiera de* $\mathbb{R}^4$ *nos diga si forman una base de un subespacio suplementario de* V. *Utilícese para determinar si los subespacios*

$$U = \langle\ (2,0,0,-1),\ (0,1,-1,0),\ (0,2,0,-2)\ \rangle$$

y

$$W = \langle\ (1,1,1,1),\ (0,-1,0,0),\ (0,1,0,-1)\ \rangle$$

son suplementarios de V.

Ejercicio 4.3 *Constrúyase un fichero* ***.m*** *tal que dados los generadores de dos subespacios cualesquiera de* $\mathbb{R}^n$ *nos indique si son suplementarios o no. Compruébese el funcionamiento del fichero con los subespacios* U *y* W *del ejercicio anterior.*

4.2.2. Repetición un número fijo de veces

Si queremos repetir determinadas instrucciones un número fijo de veces, usamos una estructura `for ... end`. Para ello necesitamos un vector. Las ordenes se repetirán tantas veces como elementos tenga el vector. Además dispondremos de una *variable de iteración* que en cada paso irá tomando el valor correspondiente que se encuentre en el vector. El esquema general es el siguiente:

for k=v
ordenes de Matlab
end

En este caso el vector es `v` y la variable de iteración es `k`. Veamos un ejemplo. Escribe en el programa **repeticion.m**

```
for k=[1,2,3]
      k^2
end
```

La variable de iteración `k` va tomando consecutivamente los valores que encuentra en el vector y se van ejecutando las órdenes.

El siguiente fichero permite calcular, mediante la estructura `for ... end`, el factorial de un número:

```
% Calcula el factorial de un número
n=input('Introduce un número:');
if n~=round(n)|n<0
      error('Debe introducir un entero no negativo')
end
f=1;
for k=1:n
      f=f*k;
end
disp(f)
```

El uso de `for` es cómodo, pero ralentiza la ejecución de los programas. En MATLAB casi siempre hay alternativas más rápidas. Vamos a comparar nuestro programa con la orden `prod`, que multiplica todos los elementos de un vector, mediante el uso de **tic ... toc**

```
>> tic, facto(20);toc
>> tic, prod(1:20);toc
```

Ejercicio 4.4 *Escribe un programa, usando la estructura* `for ... end`*, que cree un vector de 10 componentes del que se conocen las dos primeras componentes,* $a_1 = 2,$, $a_2 = 5$ *y* $a_n = 3a_{n-1} + 5a_{n-2}$.

Ejercicio 4.5 *Escribe un programa, usando la estructura* `for ... end`*, que al introducir n, cree un vector de n componentes del que se conocen las primeras componentes* $a_1 = 1,$, $a_2 = 3$ *y* $a_n = 2a_{n-1} - 4a_{n-2}$.

Ejercicio 4.6 *Escribe un programa, usando la estructura* `for ... end`*, que al introducir n,* a_1 *y* a_2 *cree un vector de n componentes tal que* $a_n = -a_{n-1} + 3a_{n-2}$.