Práctica 2

Archivos .M. Programación

2.1 Archivos .M

En Matlab, se pueden crear archivos de extensión.m,que se denominan archivos de función ó archivos de guión (o de instrucciones). Con ellos podemos definir nuevas funciones que se añadirán a las que ya trae Matlab.
Los archivos de instrucciones se llaman así porque pueden consistir en una serie de instrucciones a ejecutar. Veremos que en éstos puede ocurrir que en el momento de su ejecución nos pida una serie de valores (inputs). En todos los casos, se deberá ir a $File - New-$ y aparece una ventana ($Editor - Untitled$) donde podemos escribir el programa correspondiente.

Una vez terminado, vamos al $File$ de dicha ventana y hacemos clic en $Save\ As$ y debemos guardar el archivo .m con el nombre que le hayamos dado a la función (nombre.m). A partir del momento en que se crea la fución, si se desea utilizar, hay que situar el $current\ folder$ en la carpeta en la que hayamos guardado la función.

El nombre de la función debe empezar con una letra y para evitar confusiones, debemos asegurarnos que no coincide con el nombre de alguna de las funciones de que dispone Matlab.

Las dos primeras líneas de un archivo de función tienen la forma $function[y, z, \ldots] = nombre(a, b, \ldots)$

Donde $a, b, \ldots$ denotan las variables de entrada (las variables independientes), mientras que $y, z, \ldots$ son las variables de salida (dependientes), en ambos casos separadas por comas. El comando $function$, es de Matlab, mientras que en $nombre$, se debe introducir algo que nos sirva para identificar la función en el futuro.

Ejemplo 2.1.1.– Crear un archivo de función cuya entrada sea una matriz cuadrada x y cuyas salidas sean su número de componentes total y su determinante.

$function[numero, determinante] = ejemplo1(x)$
$[n, m] = size(x);$
$numero = n * m;$

$determinante = det(x)$; $fprintf('$El número de elementos de la matriz es :$')$
$numero$
$fprintf('$El determinante de la matriz es :$')$
$determinante$
end

Ejemplo 2.1.2.– Crear un archivo de función con el nombre somb.m para añadir al listado de funciones de Matlab la función $f(x) = \frac{sen(x)}{x}$, si $x \neq 0$. Como aún no hemos estudiado la función if, este programa quedará incompleto en el caso en que $x = 0$.

$function[y] = somb(x)$
$y = sin(x)./x$
end
Una vez creado el archivo $somb.m$, cuando tecleamos $somb(x)$, obtenemos el valor de $y(x)$,

si x es un escalar. Si $x = (aij)$ es una matriz, obtenemos una matriz cuyos elementos son $somb(aij)$.

2.1.1 Subfunciones

Un archivo de función puede contener el código de más de una función. La primera función (la función principal) que aparece en el archivo es la que da el nombre a éste y las otras se llamarán subfunciones (solo son visibles para la funcion principal o para otra subfuncion en el mismo archivo).

Ejemplo 2.1.1.– Un archivo de funcion que determina la suma de las componentes y dicha suma dividida entre el número de componentes (media) de un vector fila.

$function[a, media] = media(x)$
$n = length(x)$;
$suma = sum(x)$;
$function[a] = med(suma, n)$
$\%Calculalamedia$
$a = suma/n$;
end
$fprintf('$La suma de las componentes es :$')$
$suma$
$fprintf('$La media es :$')$
$media = med(suma, n)$
end

En Programacion, nos encontramos a menudo con la necesidad de ejecutar una misma serie de instrucciones. Por ello, puede resultar conveniente crear un archivo de guion con dichas instrucciones, al que podremos recurrir cada vez que lo necesitemos.

El nombre de uno de tales archivos solo tiene la restriccion, como ocurre con los archivos de funcion, de que debe comenzar por una letra. La estructura de estos archivos es la siguiente:

Una primera linea explicativa que describa brevemente el objetivo del archivo. Esta línea de comentarios debe comenzar con %. De esta forma se consigue que el ordenador no considere esta línea. A continuacion se escriben todas las instrucciones que deben ejecutarse en el orden que corresponda.

2.2 Operadores y Funciones lógicas

2.2.1 Operadores lógicos.

Matlab emplea los tres operadores lógicos siguientes: & (y), |(ó) y $\sim$ (negación).

Cuando operan sobre una matriz, lo hacen elemento a elemento. Cualquier número no nulo se considera como verdadero y el cero como falso. Consideremos, por ejemplo, la instrucciones siguientes:

$A = [1\ 0; 2\ 1];$
$B = [4\ 0; 0\ 3];$
$C = A\&B$
$C =$
1 0
0 1

Es decir, produce una matriz con unos y ceros , dando como resultado un 1 cuando ambos elementos sean distintos de cero, y 0, cuando al menos uno sea igual a 0.

Ejercicio 2.2.1.– Probar qué ocurre al utilizar los otros dos operadores lógicos.
También nos serán útiles los siguientes operadores:

- $<$ menor que
- $<=$ menor o igual que
- $>$ mayor que
- $>=$ mayor o igual que
- $==$ igual que
- $\sim=$ no igual a

2.2.2 Funciones lógicas .

- **find** Esta función determina los índices de una matriz que verifican una determinada expresión lógica.

  Ejemplo 2.2.1.– Sustituir en la matriz $A = [1:3; linspace(10, 15, 3); 90, 15, 1]$ los elementos mayores de 5 por 1.

  Suponiendo ya introducida la matriz, se podría operar como sigue:

  $i = find(A > 5);\ A(i) = 1;\ A$

Si se suprime el punto y coma al final de la línea $i = find(A > 5)$, en pantalla se despliega el valor de i y vemos que se trata de una matriz columna con los índices de los elementos de A que superan a 5. Los índices de A se numeran por columnas de izquierda a derecha y de arriba abajo.

- **all.** Si x es un vector con todas sus componentes no nulas, $all(x)$ es igual a 1; en otro caso su valor es cero. Si A es una matriz, $all(A)$ es una matriz fila de unos y ceros, que no es otra cosa que el resultado de aplicar all a cada columna de A (de izquierda a derecha).

  Ejemplo 2.2.2.– Sea $A = [1,0,1;1,1,5;7,3,4]$, al hacer $all(A)$, el resultado será $ans = 1\ 0\ 1$, ya que en la primera y tercera columna no hay ceros, y en la segunda columna sí.

- **any.** Si x es un vector con alguna componente no nula, entonces $any(x)$ produce un uno. Por el contrario, si x es nulo, el valor de $any(x)$ también es cero. Sobre una matriz actúa como en el caso de all.

  Ejemplo 2.2.3.– Al aplicar sobre la matriz anterior $any(A)$, el resultado será $ans = 1\ 1\ 1$, ya que todas sus columnas tienen algún elemento no nulo.

- **xor.** Si x e y son vectores, $xor(x,y)$ es un vector fila de ceros y unos, obtenido de la siguiente forma. Si en determinada posición, sólo uno de los dos vectores tiene componente no nula, entonces $xor(x,y)$ tiene un uno en dicha posición. En cualquier otro caso, tiene un cero.

  Ejemplo 2.2.4.– Dados los vectores $x = [1,0,2,3,0,1,1]$, e $y = [0,1,1,2,0,0,2]$, al aplicar $xor(x,y)$, el resultado será $ans = 1\ 1\ 0\ 0\ 0\ 1\ 0$

2.3 If, else y elseif

If evalúa una expresión lógica y ejecuta un grupo de instrucciones si la expresión lógica es verdadera. La forma más simple es la siguiente

ifexpresión lógica
instrucciones
end

Ejemplo 2.3.5.– Supongamos que queremos
obtener las raíces de una ecuación de segundo grado $ax^2 + bx + c = 0$. Podemos crear un archivo de instrucciones que nos pida los coeficientes a, b y c y determine las raíces, una vez que haya comprobado que son reales. Le daremos el nombre raices.m.

$a = input('$dame el coeficiente de $x^2\ ')$;
$b = input('$dame el coeficiente de $x\ ')$;
$c = input('$dame el término independiente $')$;
$Delta = b^2 - 4 * a * c$;
$if Delta < 0$
$disp('$no hay raíces reales$')$

$end\ x1 = (-b + sqrt(Delta))/2 * a$
$x2 = (-b - sqrt(Delta))/2 * a$

$elseif$ evalúa la expresión lógica que aparece en su misma línea si if o los anteriores $elseif$ resultan falsos. Si dicha expresión lógica es verdadera se ejecutan las instrucciones siguientes hasta el próximo $elseif$ o $else$. Por el contrario, $else$ no lleva expresión lógica a evaluar y matlab realiza las instrucciones siguientes hasta end si las expresiones lógicas de if y los $elseif$ anteriores son falsos. La forma general de un if es la siguiente:

if expresión lógica
$instrucciones$
$elseif$ expresión lógica
$instrucciones$
$elseif$ expresión lógica
$instrucciones$
$\ldots$
$else$
$instrucciones$
end

Para entender cómo funciona $elseif$ y $else$, vamos a considerar el mismo ejemplo anterior, pero vamos a establecer distintas salidas, según sea el valor de $Delta$.

Ejemplo 2.3.6.– $a = input('\text{dame el coeficiente de } x^2\ ')$;

$b = input('\text{dame el coeficiente de } x\ ')$;
$c = input('\text{dame el término independiente }')$;
$Delta = b^2 - 4 * a * c$;
$if Delta < 0$
$disp('\text{no hay races reales}')$
$elseif\ Delta > 0$
$x1 = (-b + sqrt(Delta))/2 * a$
$x2 = (-b - sqrt(Delta))/2 * a$
$else$
$x = -b/(2 * a)$
$disp('\text{raíz doble}')$
end

2.4 For

La sintaxis es la siguiente:

for índice $= inicio : incremento : final$
$instrucciones$
end

Si el incremento es la unidad, no hace falta indicarlo.

Ejemplo 2.4.7.– Crear un archivo de función para determinar $n!$.
Vamos a $File - New$ para abrir un nuevo archivo .m y escribimos:

$function[y] = factorial(n)$
$y = 1;$
$for\ k = 1 : n$
$y = y * k$
end

Ejemplo 2.4.8.– Construir una matriz A, $n \times m$, tal que $aij = \frac{1}{(i+j-1)}$.

$for\ i = 1 : n$
$for\ j = 1 : m$
$A(i,j) = 1/(i + j - 1);$
end
end
A

2.5 while

Su sintaxis es la siguiente:

$while$ expresión
$instrucciones$
end
Mientras que la expresión que controla el $while$ sea verdadera, se ejecutan todas las instrucciones comprendidas entre $while$ y end.

Ejemplo 2.5.9.– Calcular el primer natural n tal que $n!$ es un número con 100 dígitos.

$n = 1;$
$m = 1;$
$while\ m < 1e100$
$n = n + 1;$
$m = m * n;$
end

2.6 EJERCICIOS DE PROGRAMACION RESUELTOS

1. Crear un mensaje que nos diga el número de elementos positivos negativos y nulos de un vector.

$positivos = 0;$
$negativos = 0;$
$nulos = 0;$
$x = input('\text{da el vector}x');$
$n = length(x);$
$fori = 1 : n$
$ifx(i) < 0$
$negativos = negativos + 1;$
$elseifx(i) > 0$
$positivos = positivos + 1;$
$else$
$nulos = nulos + 1;$
end
end
$fprintf('\text{El número de elementos positivos es :}')$
$positivos$
$fprintf('\text{El número de elementos negativos es :}')$
$negativos$
$fprintf('\text{El número de elementos nulos es :}')$
$nulos$

2. Dar la reducción de Gauss de una matriz dada (si no necesita realizar cambios de fila).

$A = input('\text{dame la matriz A}');$
$[n, m] = size(A);$
$i = 1;$
$for\ i = 1 : (n - 1)$
$if\ A(i, i) \sim= 0$
$A(i, :) = A(i, :)/A(i, i);$
$for\ j = (i + 1) : n$
$A(j, :) = A(j, :) - A(j, i) * A(i, :);$
end
end
end
A