1.- *Funciones y condicionales*

En esta práctica definiremos las funciones y utilizaremos estructuras condicionales.

(a) Programar en MATLAB

Al igual que en los lenguajes de alto nivel, MATLAB permite crear programas utilizando programación estructurada. Para ello cuenta con condicionales, bucles y funciones. Asimismo utiliza muchos de los recursos de la programación orientada a objetos.

I. Definición de funciones simples

La instrucción *inline* permite crear funciones simples desde el teclado. Su sintaxis es la siguiente:

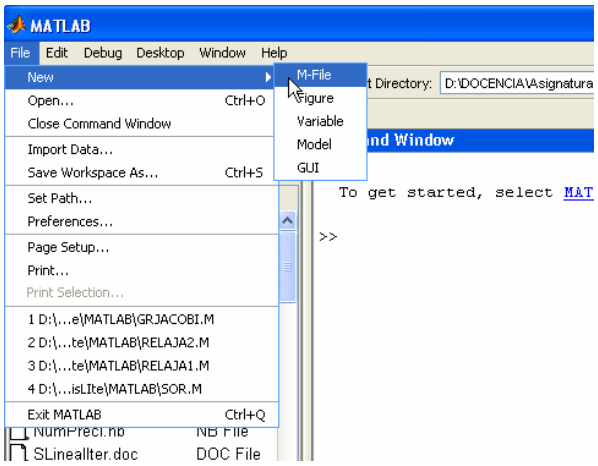
Nombre_funcion = **inline(**'*definicion en una sola línea entre comillas simples*'**)**

Ejemplo:

```
» f=inline('sqrt(exp(x))')
» p=f(4)          % Evaluación en el punto 4
» X=linspace(-1,4);Y=f(X);plot(X,Y) % Representación en el intervalo [-1,4]
```

II. Definición general de funciones

Una función se define mediante un m-fichero, cuyo nombre coincide con el de la función. MATLAB tiene integrado su propio editor, al que se accede con la siguiente selección de menús: "File"→"New"/"Open" → "Mfile" La selección "New/Open" depende de si vamos a crear un nuevo archivo o utilizamos un archivo creado previamente.



Edición de un archivo nuevo en MATLAB

En ordenadores cuyo S.O. es windows NT, XP o 2003, los directorios suelen estar protegidos contra intrusiones, por lo que no es posible salvar los ficheros nuevos en cualquier lugar. En este

caso suele haber un directorio "C:\alumnos" o "C:\usuarios" donde se guardan los trabajos realizados.

MATLAB sólo puede ejecutar funciones que estén en memoria, en sus librerías o en el directorio actual; por ello es necesario cambiar al directorio donde salvamos nuestro archivo antes de poder ejecutarlo.

La primera línea del fichero tiene la siguiente sintaxis:

f**unction** [*argumentos_salida*]= *nombre_función (argumentos_entrada)*

Después irán el resto de instrucciones necesarias para resolver el problema planteado. Cuando hay más de un argumento (variable) de salida, éstos deben ir entre corchetes y separados por comas. Es conveniente utilizar las primeras líneas del fichero como comentarios (iniciándolas con '%'), explicando cómo debe usarse la función y sus argumentos (tanto de entrada como de salida). Así, dicha definición será visible mediante la instrucción **help** *nombre-función.*

La función puede finalizarse en cualquier punto utilizando la instrucción **return**.

Las variables definidas en la función (salvo los argumentos) son locales (no se propagan fuera del entorno de ejecución de la función). Para que el valor de una variable sea compartido por varias funciones se emplea la instrucción **global**, cuya sintaxis es **global** *variable*, y debe aparecer en todas las funciones que la compartan

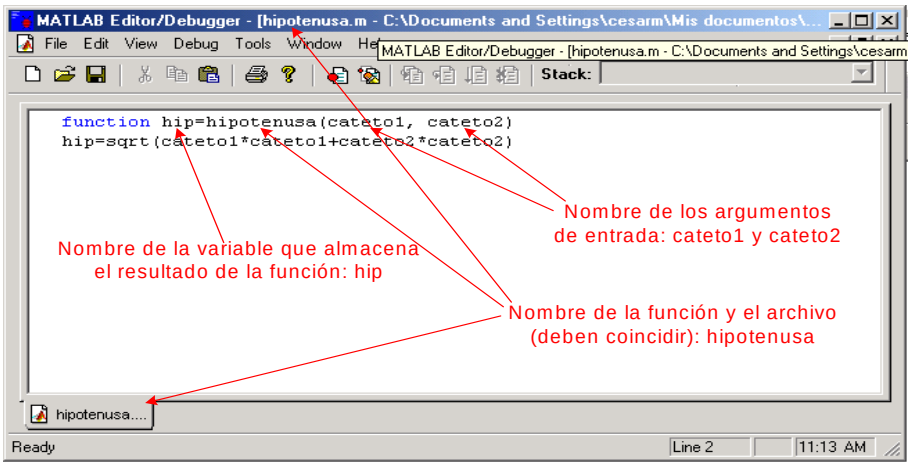
Una función utiliza las siguientes variables para verificar el número de argumentos:

nargin número de argumentos de entrada que el usuario ha pasado a la función.

nargout número de argumentos de salida que el usuario desea recibir de la función

Como ejemplo de lo anterior, comenzaremos creando una función que calcule el valor de la hipotenusa de un triángulo rectángulo a partir de sus dos catetos. A continuación se indican las instrucciones a programar:

```
function hip = hipotenusa(cateto1, cateto2)
hip = sqrt(cateto1*cateto1+cateto2*cateto2);
```



Función de cálculo de la hipotenusa.

Tras guardar el archivo, damos valores a dos variables y ejecutamos la nueva función desde MATLAB mediante

```
» x=3, y=4, hipotenusa(x,y)
x =
     3
y =
     4
ans =
     5
```

Observemos que la solución de la función se almacena en la variable *ans*, puesto que no se ha realizado la asignación a ninguna variable. Para recuperar el valor de una variable **existente** es suficiente con escribir su nombre (o acudir al *workspace*).

```
» x
x =
    3
```

Si se hubiera eliminado el símbolo ";" al final de la 2ª línea, habríamos obtenido el valor de la variable *hip* **durante la ejecución de la función**. Al acabar la función, dicha variable desaparece, por lo que no podremos usar su valor (no aparece en *workspace*).

```
» hip
??? Undefined function or variable 'hip'
```

Si se quiere que las variables *x* e *y* toman ciertos valores, pero no muestran por pantalla el resultado de la asignación, acabaremos la instrucción con ";". Probemos ahora a escribir

```
» x=3; y=4;z=hipotenusa(x,y)
z =
    5
```

En resumen, se pueden separar varias instrucciones en una misma línea por una coma o un punto y coma, diferenciándose en que el segundo caso no muestra los resultados de las operaciones. De ahí que todas las instrucciones de una función acaben con punto y coma. Cuando una instrucción es demasiado larga para escribirla en una sola línea, se puede acabar con tres puntos ("`...`") indicando que continúa en la línea siguiente.

Una función predefinida por MATLAB y muy útil es **error**, cuya sintaxis es **error('Mensaje')**. Esta función finaliza la ejecución del programa actual, enviando a la pantalla un mensaje. La misma funcionalidad se puede obtener combinando las instrucciones **disp** y **return**, mediante **disp(**'mensaje'**);return**

III. Estructuras de control condicionadas

Permite seleccionar entre dos conjuntos alternativos de instrucciones dependiendo de que se verifique una condición lógica (cuyo resultado es cierto o falso). Su sintaxis es de la forma:

if *condición*

Instrucciones que deben ejecutarse si la condición 1 es cierta.

else

Instrucciones a ejecutar si no se verifica la condición anterior

end

Cuando no hay instrucciones que ejecutar si la condición no se cumple, la sintaxis anterior se reduce a

if *condición*

Instrucciones que deben ejecutarse

end

Por el contrario, cuando se encadenan varios bloques alternativos, la sintaxis queda como:

if *condición_1*

Instrucciones a ejecutar cuando se verifica la condición 1

elseif *condición_2*

Instrucciones a ejecutar cuando no se verifica la condición 1 y sí la condición_2

elseif *condición_3*

…

else

Instrucciones a ejecutar cuando no se verifican las condiciones anteriores

end

Las siguientes instrucciones dan el signo de un número *x*.

```
» if x>0; disp('signo positivo');
  elseif x<0;disp('signo negativo');
  else;disp('valor nulo');
  end
```

Podemos imponer más de una condición, o condiciones complejas, utilizando los operadores relacionales (condiciones cuyo resultado es cierto o falso) combinados con operadores lógicos (sirven como nexo entre varios relacionales).

Símbolo	Operador Lógico
<	Menor (para complejos sólo a las partes reales)
<=	Menor o igual (sólo afecta a partes reales)
>	Mayor (sólo afecta a partes reales)
>=	Mayor o igual (sólo afecta a partes reales)
x==y	Igualdad (afecta a los números complejos)
x~=y	Desigualdad (afecta a los números complejos)

Símbolo	Operador Relacional
-A	Negación lógica o complementario (NO)
A&B	Conjunción lógica o intersección (Y)
A | B	Disyunción lógica o unión de A y B (O)
xor(A,B)	O exclusivo
any	Algún elemento cumple la condición
all	Todos los elementos cumplen la condición

Ejercicios recomendados:

1. Programar la función

$$f(x)\begin{cases} -2x-1 & x\le -1 \\ x^2 & -1<x<1 \\ 2x-1 & x\ge 1 \end{cases}$$

2.- *Anexo: Instrucciones necesarias*

I. Introducciones de manejo de vectores y matrices

Repasamos aquí las instrucciones ya introducidas cursos anteriores.

vector=[a, b, c, d, …m] Define un vector fila, cuyos elementos son los valores dados.

vector=[a; b; c;, d; …m] Idem con un un vector columna.

variable=[primer_elemento:último_elemento] Define el vector cuyos primeros y último elementos son los especificados, y los elementos intermedios se diferencian en una unidad.

variable=[primer_elemento:incremento:último_elemento] Define el vector cuyos primeros y último elementos son los especificados, y los elementos intermedios se diferencian en la cantidad especificada por el incremento

variable=linspace(primer_elemento,último_elemento,n) Define el vector cuyos primeros y último elementos son los especificados, y que tiene en total n elementos uniformemente espaciados.

variable=logspace(primer_elemento,último_elemento,n) Define el vector cuyos primeros y último elementos son los especificados, y que tiene en total n elementos en escala logarítmica uniformemente espaciados entre sí.

Para definir una matriz en Matlab, basta con introducir entre corchetes todos sus vectores fila separados por punto y coma. Los vectores se pueden introducir separando sus componentes por espacios en blanco o por comas.

II. Selección de los elementos de un vector x o matriz A

x(n)	Devuelve el n-ésimo elemento del vector x
x([n,m,p])	Devuelve los elementos del vector x situados en las posiciones n-ésima, m-ésima y p-ésima.
x(n:m)	Devuelve los elementos del vector x situados entre el n-ésimo y el m-ésimo, ambos inclusive
x(n:p:m)	Devuelve los elementos del vector x situados entre el n-ésimo y el m-ésimo, ambos inclusive pero separados de p en p unidades
A(m,n)	Devuelve el elemento (m,n) de la matriz A (fila m y columna n)
A([m, n],[p, q])	Devuelve la submatriz de A formada por la intersección de las filas n-ésima y m-ésima y las columnas p-ésima y q-ésima.
A(n,:)	Devuelve la fila n-ésima de la matriz A
A(:,p)	Devuelve la columna p-ésima de la matriz A
A(:)	Devuelve un vector columna cuyos elementos son las columnas de A situadas por orden
A(:,:)	Devuelve toda la matriz A
[A,B,C]	Devuelve la matriz formada por las submatrices A,B,C,...

III. Funciones trigonométricas e hiperbólicas

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

IV. Funciones exponenciales

exp(Z)	Función exponencial de base e	log(Z)	Función Logaritmo neperiano
sqrt(Z)	Función Raíz cuadrada	log10(Z)	Función Logaritmo decimal

V. Funciones específicas de variable numérica

abs(Z)	Módulo o valor absoluto	rem(a,b)	Da el resto de la división entre los reales a y b
angle(Z)	Argumento	fix(x)	Elimina la parte decimal del real x
conj(Z)	Complejo conjugado	floor(x)	Redondea los decimales al menor entero más cercano
imag(Z)	Parte imaginaria	ceil(x)	Redondea los decimales al mayor entero más cercano
real(Z)	Parte real	round(x)	El entero más próximo al real x
sign(x)	Signo del real x (1 si x>0, -1 si x<0)		

VI. Funciones Matriciales

max(x)	Máximo de los números los elementos de x	gcd(x)	Máximo común divisor de los elementos de x
min(x)	Mínimo de los números los elementos de x	lcm(x)	Mínimo común múltiplo de los elementos de x
size(A)	Filas y columnas de la matriz A	length(x)	Longitud del vector x

VII. Operaciones de matrices

transpose(A)	Traspuesta de la matriz A	det(A)	Determinante de la matriz A
A'	Traspuesta de la matriz A	trace(A)	Suma de los elementos de la diagonal de A
inv(A)	Inversa de la matriz A (A^{-1})	norm(A)	Norma de A (mayor valor singular de la matriz A)
zeros(m,n)	Crea una matriz con m filas y n columnas cuyos elementos son ceros	ones(m,n)	Crea una matriz con m filas y n columnas cuyos elementos son unos
eye(n)	Crea la matriz identidad de orden n	diag(v)	Crea una matriz diagonal con los elementos de v

VIII. Funciones lógicas vectoriales

isempty	Cierto si el vector es vacío	find(A)	Cierto si encuentra índices de elementos no nulos de A
isequal	Cierto si ambos son iguales	any(A)	Cierto si algún elemento de A es no nulo
isfinite	Cierto si es un valor finito	all(A)	Cierto si todos los elementos de A son no nulos

IX. Variable especiales

En MATLAb existen variables de uso común, cuyo valor viene ya preasignado.

pi	3'1415926535897...	NaN	Indeterminación (Not a Number, por ejemplo 0/0)
i ó j	Unidad imaginaria (raíz cuadrada de-1)	realmin	El menor número real positivo utilizable
inf	Infinito, por ejemplo 1/0	realmax	El mayor número real positivo utilizable
eps	Menor valor positivo que sumado a la unidad tiene representación diferente a 1.	ans	Variable creada automáticamente para representar el último resultado

X. Gráficos bidimensionales (2-D)

plot(X)	Representa los puntos (k,X_k). Si X es una matriz, hace lo mismo para cada columna de la matriz. Si X es un vector complejo, representa Real(X) frente a IMAG(X)..
plot(X,Y)	Representa el conjunto de puntos (X,Y). Si X o Y son matrices, representa por filas o columnas los datos de X frente a los datos de Y, dependiendo si el otro vector es fila o columna. Para valores complejos de X e Y, se ignoran las partes imaginaria.
grid	Sitúa rejillas en los ejes de un gráfico 2-D o ·3-D. La opción grid on coloca las rejillas y greed off las elimina. La opción grid permuta entre on y off
hold	Permite mantener el gráfico existente con todas sus propiedades, de modo que el siguiente gráfico que se realice se sitúe sobre los mismos ejes y se superponga al existente. La opción hold on activa la opción y hold off la elimina. La opción hold permuta entre on y off. Válido para 2-D y 3-D
subplot(m,n,p)	Divide la ventana gráfica en mxn subventanas y coloca el gráfico corriente en la ventana p-ésima, empezando a contar por la parte superior izquierda y de izquierda a derecha hasta acabar la línea, para pasar a la siguiente
plot(X,Y,S)	Gráfica de plot(X,Y) con la opciones definidas en S. Usualmente, S se compone de dos caracteres entre comillas simples, el primero de los cuales fija el color de la línea del gráfico, y el segundo fija el carácter a usar en el graficado. Los valores posibles de colores y caracteres son, respectivamente, los siguientes:

y	amarillo	g	verde	.	puntos	-	sólido
m	magenta	b	azul	o	círculos	:	a puntos
c	cyan	w	blanco	x	x-marcas	-.	guiones y puntos
r	rojo	k	negro	+	signo más	--	semisólidos
				*	estrellas		

XI. Titulos, etiquetas, mallas y textos

title('texto')	Añade el texto como título del gráfico en la parte superior del mismo	text(x,y,'texto')	Sitúa el texto en el punto (x,y)
xlabel('texto')	Sitúa el texto al lado del eje x	ylabel('texto')	Sitúa el texto al lado del eje y
gtex('texto')	Permite situar el texto en un punto seleccionado con el ratón	ginput(n)	Permite recuperar las coordenadas de n puntos de un grafico mediante ratón