



**Universidad Nacional  
Autónoma de México**

**Facultad De Ingeniería**

**Ingeniería Eléctrica**

**Electrónica**

**Estructura de Datos y**

**Algoritmos I**

**ActividadViernes01-Acordeon  
de lenguaje C Y MATLAB**

**Hernández Torres Mario Ivan**

**1 de marzo de 2021**

## Acordeón de Lenguaje C

La estructura general de un programa en lenguaje C es la siguiente:

- **Directivas del preprocesador.** Son las instrucciones al compilador antes de que se cumple el programa principal. Las más utilizadas son `#include` y `#define`.
- **Declaraciones globales.** Indican al compilador que las funciones definidas por el usuario o variables así declaradas son comunes a todas las funciones de su programa. Estas se sitúan antes de la función `main ()`. Pueden incluir la declaración de variables globales y de prototipos de las funciones en ese código fuente.
- **Función principal.** La cual es `main ()`. Es la primera función que se ejecuta en un programa y solo debe existir un solo `main` en todo el programa.
- **Otras funciones.** Las funciones definidas por el usuario se invocan por su nombre y los parámetros que puedan tener. Cuando la función es llamada, el código asociado con la función se ejecuta, cuando termina de ejecutarse, el control se regresa a la función desde donde fue llamada para ejecutar las instrucciones siguientes. Las funciones requieren la declaración de prototipo en la sección de declaraciones globales. Estas declaraciones indican al compilador la forma por la que la función que será invocada en el programa.

Existen distintas formas de declarar una variable.

Hay de tipo `char`(8 bits), `int`(16 bits), `float`(32 bits), `double`(64 bits). `Char` se refiere a datos de tipo `character`. `Int` se refiere a valores enteros. `Float` y `double` se refieren a valores reales.

Código	Descripción
%d	El dato se considera como entero decimal.
%o	El dato se considera como octal.
%X, %x	El dato se considera como hexadecimal.
%u	El dato entero se toma como entero sin signo.
%c	El dato se considera de tipo carácter.
%e	El dato se considera de tipo float. Se convierte a notación científica, de la forma (-)n.mmmmmE(+I-)dd.
%f	El dato se considera de tipo float. Se convierte a notación decimal, con parte entera y los dígitos de precisión.
%s	El dato ha de ser una cadena de caracteres.
%lf	El dato se considera de tipo double.

Secuencia especial	Símbolo	Secuencia especial	Símbolo
\a	Alarma	\V	Tabulación vertical
\b	Retroceso de espacio	\\	\
\f	Avance de página	\?	?
\n	Salto de línea	\"	"
\r	Retorno de carro	\0	Cero, nulo
\t	Tabulación		

Los distintos operadores aritméticos que hay en lenguaje C son los siguientes.

Operador	Operación	Ejemplo	Resultado
==	Igual que	4 == 4 "casa" == "caza"	Verdadero Falso
!=	Diferente a	4 != 4 "casa" != "caza"	Falso Verdadero
<	Menor que	4 < 7	Verdadero
>	Mayor que	4 > 7	Falso
<=	Menor o igual que	5 <= 5 5.1 <= 5	Verdadero Falso
>=	Mayor o igual que	5 >= 5 5.1 >= 5	Verdadero Verdadero

Operador	Operación	Ejemplo	Resultado
&&	Se cumplan 2 o más condiciones indicadas al mismo tiempo	(5 == 5) && ( "Hola" == "Adios")	Falso
	Se cumpla una de dos condiciones indicadas	(5 == 5)    ( "Hola" == "Adios")	Verdadero
!	Negación de sentencia indicada	!(5 == 5)	Falso

La sintaxis para la sentencia if es la siguiente:

<i>if ( expresión )</i>	
{	
	<i>...Bloque de instrucciones</i>
}	

La sintaxis para la sentencia if else es la siguiente:

<i>if ( expresión )</i>	
{	
	<i>...Bloque de instrucciones que se ejecutan si la EXPRESIÓN es VERDADERA</i>
}	
<i>else</i>	
{	
	<i>...Bloque de instrucciones que se ejecutan si la EXPRESIÓN es FALSA</i>
}	

La sintaxis para la sentencia switch es la siguiente:

```
switch( expresión)
{
    case valor1:
        bloque de instrucciones para valor1
        break;
    case valor2:
        bloque de instrucciones para valor2
        break;
    ... ..
    default:
        bloque de instrucciones por default
}
```

La estructura para la sentencia y el ciclo while y do while

```
do
{
    ...Bloque de instrucciones que se ejecutan mientras
    EXPRESIÓN sea VERDADERA
} while( expresión);
```

```
while( expresión)
{
    ...Bloque de instrucciones que se ejecutan mientras
    EXPRESIÓN sea VERDADERA
}
```

Operadores de incremento y decremento.

Expresión	Acción
k = --i;	Resta a i una unidad y el resultado lo asigna a k
k = ++i;	Suma a i una unidad y el resultado lo asigna a k
k = i--;	Asigna el valor de i a k, y después decrementa el valor de i
k = i++;	Asigna el valor de i a k, y después incrementa el valor de i

El ciclo for.

```
for (VC=VI ; Condición de VC respecto a VF; VC=VC + ID)
{
    ...Bloque de instrucciones
}
```

Los arreglos que son una secuencia de datos del mismo tipo. Cada uno de estos datos se enumeran e identifican a través de un índice de valores enteros partiendo desde el 0.

Arreglos de dos dimensiones, se trata de un arreglo donde cada uno de los elementos es otro arreglo. Equivale a una matriz de 2 dimensiones.

```
for(indiceRenglon=0; indiceRenglon<m; indiceRenglon++)
{
    for(indiceColumna=0 ; indiceColumna < n ; indiceColumna++)
    {
        Operaciones con
        arregloBidimensional[indiceRenglon][indiceColumna];
    }
}
```

El manejo de archivos. Se utilizaba la función fopen para crear y abrir archivos. Existen distintos modos para abrir un archivo. Y la función fclose escribe el contenido y cierra el archivo previamente abierto con fopen.

Modo	Significado
r	Abre el archivo para lectura.
w	Crea un archivo para escritura. Sobreescribe el archivo si ya existe.
a	Abre el archivo para agregar contenido después de lo ultimo que contiene.
r+	Abre el archivo para lectura y escritura. No sobreescribe el archivo si ya existe.
w+	Crea un archivo para escritura y lectura. Sobreescribe el archivo si ya existe.
a+	Abre el archivo para agregar y leer contenido después de lo ultimo que contiene. Si no existe lo crea.

Existe también la función fprintf la cual escribe un valor con formato en un archivo se ocupa la siguiente sintaxis:

*Fprintf(apuntadorAlArchivo, "cadena opcional e indicadores de formato", "argumentos");*

La función fputc, escribe un carácter a la vez del archivo. Se utiliza la siguiente sintaxis.

*fputc(caracter, apuntadorAlArchivo);*

Funcion fgetc, lee un carácter desde un archivo. Se ocupa con la siguiente sintaxis.

*fgetc(apuntadorAlArchivo);*

La funcion feof sirve para determinar si se ha llegado al EOF. Tiene como valor de retorno un entero. Si el carácter leído es diferente de EOF regresa un valor de 0. Si el valor leído es EOF regresa un valor distinto de 0. A través de un ciclo recorre el contenido del archivo:

```
while ( feof(archivo) == 0) /*Mientras no se detecte el fin de archivo */  
{  
...  
}
```

**Los apuntadores** Un apuntador es una variable que sirve para apuntar a otra variable. La manera en que una variable apuntador apunta a otra variable es almacenando la dirección de memoria donde reside ésta ultima. Para declarar un apuntador se usa el operador \* antes del identificador de la variable. Para hacer referencia a la dirección de memoria de una variable se usa el operador &.

**Paso de parámetros por valor.** Cuando se invoca a una función y se le proporcionan argumentos, la función recibe una copia de los valores de los argumentos y los guarda en variables locales a ésta. Si se cambia el valor de una variable local en la función invocada, el cambio sólo afecta a la función y no tiene efecto en las variables que se le pasaron como argumentos.

**Paso de parámetros por referencia.** Se pasa a la función invocada las direcciones de memoria donde se almacenan los valores. La función invocada recibe las direcciones de memoria y las guarda en variables de tipo apuntador. Cuando en la función invocada se modifican los valores recibidos, éste valor queda almacenado en la misma dirección de memoria. Por lo tanto, los valores originales que se pasaron como argumentos se habrán modificado.

## Mario – MATLAB

Matlab tiene diferentes operaciones que son las básicas para trabajar.

Operación	Símbolo	Expresión en Matlab
suma	+	$a + b$
resta	-	$a - b$
multiplicación	*	$a * b$
división	/	$a / b$
potencia	^	$a ^ b$

Las operaciones tienen un orden de precedencia el cual es el siguiente.

Orden de precedencia de operaciones	
1º	^
2º	* /
3º	+ -

Si se quiere evaluar la línea pero que no escriba la respuesta, solo hay que escribir punto y coma (;) al final de la sentencia.

Si la sentencia es demasiado larga para que quepa en una sola línea podemos poner tres puntos (...) después de un enter.

Ejemplos:

```
>> a = 7      % damos valor a la variable a y la escribe por pantalla
a =
    7

>> b = 4;     % no escribe el valor de b por el ; del final
```

Otra forma sería guardar el estado de una sesión de trabajo con el comando save antes de salir:

**>> save**

Al teclear esto, automáticamente se crea un fichero llamado matlab.mat. Puede recuperarse la siguiente vez que se arranque el programa con el comando load:

**>> load**



Matlab no cambia la representación interna de un número cuando se escogen distintos formatos, sólo se modifica la forma de visualizarlo.

Tipo	Resultado	Ejemplo: >> pi
<b>format short</b>	Formato coma fija con 4 dígitos después de la coma (es el formato que viene por defecto)	3.1416
<b>format long</b>	Formato coma fija con 14 o 15 dígitos después de la coma	3.14159265358979
<b>format short e</b>	Formato coma flotante con 4 dígitos después de la coma	3.1416e+000
<b>format long e</b>	Formato coma flotante con 14 o 15 dígitos después de la coma	3.141592653589793e+000
<b>format short g</b>	La mejor entre coma fija o flotante con 4 dígitos después de la coma	3.1416
<b>format long g</b>	La mejor entre coma fija o flotante con 14 o 15 dígitos después de la coma	3.14159265358979
<b>format short eng</b>	Notación científica con 4 dígitos después de la coma y un exponente de 3	3.1416e+000
<b>format long eng</b>	Notación científica con 16 dígitos significantes y un exponente de 3	3.14159265358979e+000
<b>format bank</b>	Formato coma fija con 2 dígitos después de la coma	3.14
<b>format hex</b>	Hexadecimal	400921fb54442d18
<b>format rat</b>	Aproximación racional	355/113
<b>format +</b>	Positivo, negativo o espacio en blanco	+

Tecleando *clear* podemos borrar todas las variables del espacio de trabajo, pero no borra lo de las demás ventanas, es decir, no desaparece lo que hay escrito en la ventana de comandos.

Tecleando *clc* borramos lo que hay en la ventana de comandos pero no borra las variables de la memoria del espacio de trabajo.

Los comentarios se escriben después del símbolo de tanto por ciento (%), de este modo todo lo que se escriba a continuación en la misma línea no será leído por Matlab. Podemos colocar varias órdenes en una línea si se separan correctamente, puede ser:

*por comas (,) que hacen que se visualicen los resultados*

*puntos y comas (;) que suprimen la impresión en pantalla*

Función	¿Qué hace?	Ejemplo x = 5.92
<b>ceil (x)</b>	redondea hacia infinito	6
<b>fix (x)</b>	redondea hacia cero	5
<b>floor (x)</b>	redondea hacia menos infinito	5
<b>round (x)</b>	redondea hacia el entero más próximo	6



Función	¿Qué hace?
... (x)	función trigonométrica con el ángulo expresado en radianes
<b>sin (x)</b>	seno (radianes)
<b>cos (x)</b>	coseno
<b>tan (x)</b>	tangente
<b>csc (x)</b>	cosecante
<b>sec (x)</b>	secante
<b>cot (x)</b>	cotangente
...d (x)	función trigonométrica con el ángulo expresado en grados
<b>sind (x)</b>	seno (grados)
...	...
...h (x)	función trigonométrica hiperbólica con el ángulo expresado en radianes
<b>sinh (x)</b>	seno hiperbólico (radianes)
...	...
a... (x)	inversa de la función trigonométrica con el resultado expresado en radianes
<b>asin (x)</b>	arco seno (radianes)
...	...
a...d (x)	inversa de la función trigonométrica con el resultado expresado en grados
<b>asind (x)</b>	arco seno (grados)
...	...
a...h (x)	inversa de la función trigonométrica hiperbólica con el resultado expresado en radianes
<b>asinh (x)</b>	arco seno hiperbólico (radianes)
...	...

Función	¿Qué hace?
<b>abs (x)</b>	valor absoluto o magnitud de un número complejo
<b>sign (x)</b>	signo del argumento si x es un valor real (-1 si es negativo, 0 si es cero, 1 si es positivo)
<b>exp (x)</b>	exponencial
<b>gcd (m,n)</b>	máximo común divisor
<b>lcm (m,n)</b>	mínimo común múltiplo
<b>log (x)</b>	logaritmo neperiano o natural
<b>log2 (x)</b>	logaritmo en base 2
<b>log10 (x)</b>	logaritmo decimal
<b>mod(x,y)</b>	módulo después de la división
<b>rem (x,y)</b>	resto de la división entera
<b>sqrt (x)</b>	raíz cuadrada
<b>nthroot (x,n)</b>	raíz n-ésima de x

Ejemplos:

>> lcm (10,25)      % mínimo común múltiplo

ans =

50

>> mod (-12,5)      % módulo de la división de -12 entre 5

ans =

3

Función	¿Qué hace?	Ejemplos: $x = 3 + 4i$ $y = 2$ $z = 7$
<b>abs (x)</b>	magnitud del número complejo x	5
<b>angle (x)</b>	ángulo (en radianes) del complejo x	0.9273
<b>complex (y,z)</b>	genera el complejo $y + zi$	$2.0000 + 7.0000i$
<b>conj (x)</b>	conjugado del número complejo x	$3.0000 - 4.0000i$
<b>imag (x)</b>	parte imaginaria del número complejo x	4
<b>real (x)</b>	parte real del número complejo x	3
<b>sign (x)</b>	divide el complejo x por su magnitud, devuelve un número complejo con el mismo ángulo de fase pero con magnitud 1	$0.6000 + 0.8000i$
<b>isreal (x)</b>	devuelve 1 si es real, y 0 si es complejo	0

## Vectores y matrices.

Para crear un vector se introducen los valores deseados separados por espacios (o comas) todo ello entre corchetes []. Si se quiere crear una matriz lo hacemos de forma análoga, pero separando las filas con puntos y comas (;).

Generalmente se usan letras mayúsculas cuando nombremos a las matrices y minúsculas para vectores y escalares. Esto no es imprescindible y Matlab no lo exige, pero resulta útil.

Para acceder a los elementos individuales de un vector lo haremos utilizando subíndices, así  $x(n)$  sería el n-ésimo elemento del vector x. Si queremos acceder al último podemos indicarlo usando end como subíndice.

Para acceder a un bloque de elementos a la vez, se usa la notación de dos puntos (:), así  $x(m:n)$  nos da todos los elementos desde el m-ésimo hasta el n-ésimo del vector x.

Si se introduce un número entre el primero y el segundo también separado por dos puntos (:) se mostrarán los elementos del primero al último indicado, incrementados según el número que aparece en el centro (o decrementados si el número es negativo).

A parte de definir un vector introduciendo cada uno de sus elementos, también podemos crearlo haciendo uso de las siguientes sentencias:

$(a:b)$  crea un vector que comienza en el valor a y acaba en el valor b aumentando de 1 en 1.

$(a:c:b)$  crea un vector que comienza en el valor a y acaba en el valor b aumentando de c en c.

$\text{linspace}(a,b,c)$  genera un vector linealmente espaciado entre los valores a y b con c elementos.

$\text{linspace}(a,b)$  genera un vector linealmente espaciado entre los valores a y b con 100 elementos.

logspace (a,b,c) genera un vector logarítmicamente espaciado entre los valores  $10^a$  y  $10^b$  con c elementos.

logspace (a,b) genera un vector logarítmicamente espaciado entre los valores  $10^a$  y  $10^b$  con 50 elementos.

Al igual que pasa con los vectores, existen unas sentencias que nos ayudan a crear más rápidamente algunas matrices que Matlab ya tiene predefinidas (m y n deben tomar valores naturales):

zeros (n) crea una matriz cuadrada n x n de ceros.

zeros (m,n) crea una matriz m x n de ceros.

ones (n) crea una matriz cuadrada n x n de unos.

ones (m,n) crea una matriz m x n de unos.

rand (n) crea una matriz cuadrada n x n de números aleatorios con distribución uniforme (0,1).

rand (m,n) crea una matriz m x n de números aleatorios con distribución uniforme (0,1).

randn (n) crea una matriz cuadrada n x n de números aleatorios con distribución normal (0,1).

randn (m,n) crea una matriz m x n de números aleatorios con distribución normal (0,1).

eye (n) crea una matriz cuadrada n x n de unos en la diagonal y ceros el resto.

eye (m,n) crea una matriz m x n de unos en la diagonal y ceros el resto.

magic (n) crea una matriz cuadrada n x n de enteros de modo que sumen lo mismo las filas y las columnas.

hilb (n) crea una matriz cuadrada n x n de Hilbert, es decir, los elementos (i,j) responden a la expresión  $(1/(i+j-1))$ .

invhilb (n) crea una matriz cuadrada n x n que es la inversa de la matriz de Hilbert.

## OPERACIONES BÁSICAS CON MATRICES

Símbolo	Expresión	Operación
+	$A + B$	Suma de matrices
-	$A - B$	Resta de matrices
*	$A * B$	Multiplicación de matrices
.*	$A .* B$	Multiplicación elemento a elemento de matrices
/	$A / B$	División de matrices por la derecha
./	$A ./ B$	División elemento a elemento de matrices por la derecha
\	$A \setminus B$	División de matrices por la izquierda
.\	$A \setminus B$	División elemento a elemento de matrices por la izquierda
^	$A ^ n$	Potenciación (n debe ser un número, no una matriz)
.^	$A .^ B$	Potenciación elemento a elemento de matrices
'	$A '$	Trasposición compleja conjugada
.'	$A .'$	Trasposición de matrices

## FUNCIONES PARA OPERAR CON VECTORES

Función	¿Qué hace?
<b>cross (x,y)</b>	producto vectorial entre los vectores x e y
<b>dot (x,y)</b>	producto escalar entre los vectores x e y

## FUNCIONES PARA EL ANÁLISIS DE MATRICES

Función	¿Qué hace?
<b>cond (A)</b>	número de condición
<b>det (A)</b>	determinante
<b>diag (v)</b>	crea una matriz diagonal con el vector v sobre la diagonal
<b>diag (A)</b>	extrae la diagonal de la matriz A como un vector columna
<b>eig (A)</b>	valores propios
<b>inv (A)</b>	matriz inversa
<b>length (A)</b>	máxima dimensión
<b>norm (A)</b>	norma
<b>norm (A,n)</b>	norma-n
<b>normest (A)</b>	estimación de la norma-2
<b>null (A)</b>	espacio nulo
<b>orth (A)</b>	ortogonalización
<b>pinv (A)</b>	pseudoinversa
<b>poly (A)</b>	polinomio característico
<b>rank (A)</b>	rango
<b>rref (A)</b>	reducción mediante la eliminación de Gauss de una matriz
<b>size (A)</b>	dimensiones
<b>trace (A)</b>	traza
<b>tril (A)</b>	matriz triangular inferior a partir de la matriz A
<b>triu (A)</b>	matriz triangular superior a partir de la matriz A

## OTRAS OPERACIONES CON MATRICES

Función	¿Qué hace?
<b>find (A)</b>	devuelve los índices donde las entradas de A son distinto de cero
<b>flipr (A)</b>	intercambia la matriz de izquierda a derecha
<b>flipud (A)</b>	intercambia la matriz de arriba a abajo
<b>reshape (A,m,n)</b>	devuelve una matriz m x n cuyos elementos se toman por columnas de A, si A no contiene m x n elementos daría un error
<b>rot90 (A)</b>	gira la matriz 90° en sentido contrario a las agujas del reloj
<b>rot90 (A,n)</b>	gira la matriz n x 90°
<b>expm (A)</b>	matriz exponencial
<b>logm (A)</b>	matriz logarítmica
<b>sqrtn (A)</b>	matriz de raíces cuadradas
<b>funm (A,@función)</b>	evalúa la función que indiquemos en la matriz A
<b>exp, log, sqrt...</b>	operan elemento a elemento
<b>[VE,VA] = eig (A)</b>	VE son los vectores propios y VA son los valores propios
<b>[L,U] = lu (A)</b>	factorización LU
<b>[Q,R] = qr (A)</b>	factorización QR

## Estructuras

Es una agrupación de datos de tipo diferente bajo un mismo nombre. A los datos les llamamos campos. No hace falta definir previamente el modelo de la estructura, podemos ir creando los distintos campos uno a uno o bien con el comando struct, donde los nombres de los campos se escriben entre apóstrofes (') seguidos del valor que se les quiere asignar.

### OPERAR CON ESTRUCTURAS

Función	¿Qué hace?
<b>fieldnames (E)</b>	devuelve el nombre de los campos de la estructura E
<b>isfield (E, 'c')</b>	devuelve 1 si c es un campo de la estructura E y 0 si no lo es
<b>isstruct (E)</b>	devuelve 1 si E es una estructura y 0 si no lo es
<b>rmfield (E, 'c')</b>	elimina el campo c de la estructura E

### OPERAR CON VECTORES Y MATRICES DE CELDAS

Función	¿Qué hace?
<b>cell (m,n)</b>	crea una matriz de celdas con m filas y n columnas
<b>celldisp (c)</b>	muestra el contenido de todas las celdas de c
<b>cellplot (c)</b>	muestra la representación gráfica de las celdas de c
<b>iscell (c)</b>	devuelve 1 si es una matriz de celdas y 0 si no lo es
<b>num2cell (x)</b>	convierte el vector o matriz numérica en celdas

## ***OPERADORES RELACIONALES***


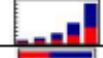
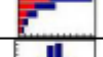

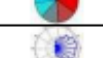


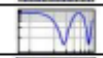



Operador	¿Qué significa?
<	menor que
<=	menor o igual que
>	mayor que
>=	mayor o igual que
==	igual a
~=	distinto de

## ***OPERADORES LÓGICOS***

Operador	¿Qué significa?
&	y
	o
~	no

Además de los operadores relacionales y lógicos básicos anteriores, Matlab proporciona una serie de funciones relacionales y lógicas adicionales que incluyen:

Función	¿Qué significa?
<b>xor (x,y)</b>	operación “o” exclusiva, devuelve 0 si ambas son falsas o ambas verdaderas y devuelve 1 si una es falsa y la otra verdadera
<b>any (x)</b>	devuelve 1 si algún elemento en un vector x es no nulo y devuelve 0 si son todos nulos, si se trata de una matriz da una respuesta por cada columna
<b>all (x)</b>	devuelve 1 si todos los elementos en un vector x son no nulos y 0 si existe alguno nulo y si se trata de una matriz da una respuesta por cada columna
<b>exist ('x')</b>	devuelve uno si existe y cero si no existe
<b>isnan (x)</b>	devuelve unos en magnitudes no numéricas (NaN) en x
<b>isinf (x)</b>	devuelve unos en magnitudes infinitas (Inf) en x
<b>isfinite (x)</b>	devuelve unos en valores finitos en x

Orden	¿Qué hace?	Imagen
<b>area</b>	colorea el area bajo la gráfica	
<b>bar</b>	diagrama de barras (verticales)	
<b>barh</b>	diagrama de barras (horizontales)	
<b>hist</b>	histograma	
<b>pie</b>	sectores	
<b>rose</b>	histograma polar	
<b>stairs</b>	gráfico de escalera	
<b>stem</b>	secuencia de datos discretos	
<b>loglog</b>	como plot pero con escala logarítmica en ambos ejes	
<b>semilogx</b>	como plot pero escala logarítmica en el eje x	
<b>semilogy</b>	como plot pero escala logarítmica en el eje y	

## Polinomios

Función	¿Qué es?
<b>conv (p,q)</b>	multiplica los dos polinomios p y q
<b>deconv (c,q)</b>	divide el polinomio c entre q
<b>polyder (p)</b>	calcula la derivada del polinomio p
<b>polyder (p,q)</b>	calcula la derivada del producto de los polinomios p y q
<b>polyval (p,A)</b>	evalúa el polinomio p en todos los valores de la matriz A

## Analisis numerico

Función	¿Qué hace?
<b>diff ('f')</b>	derivada de la función respecto a x
<b>diff ('f,t')</b>	derivada parcial de la función respecto a t
<b>diff ('f,n')</b>	derivada n-ésima de la función respecto a x
<b>feval ('f,a')</b>	evalúa la función en a
<b>fminbnd ('f,a,b')</b>	calcula el mínimo de una función de una variable
<b>fzero ('f,a')</b>	busca el cero de una función unidimensional f más próximo al punto a
<b>quad ('f,a,b')</b>	aproxima la integral definida (según la cuadratura de Simpson)
<b>trapz (x,y)</b>	integral numérica trapezoidal de la función formada al emparejar los puntos de los vectores x e y



## Bibliografía

Fernández, C. C. (28 de febrero de 2021). Obtenido de  
<http://webs.ucm.es/centros/cont/descargas/documento11541.pdf>