Operaciones Aritméticas		
Operación	**Símbolo**	**Ejemplo**
Suma	**+**	`5 + 3`
Resta	**-**	`5 - 3`
Multiplicación	*****	`5 * 3`
División	**/**	`5 / 3`
Exponenciación	**^**	`5 ^ 3`

Operadores relacionales		**Operadores lógicos**	
Operador	**Descripción**	**Nota:** Funcionan con dos operandos **A** y **B**. Ejemplos: **A&B, A|B, ~A, ~B**	
<	Menor que		
>	Mayor que	**Operador**	**Descripción**
<=	Menor o igual que	**&** (Y) vector **&&** (Y) escalar	Verdadero si ambos son verdaderos
==	Igual a	**|** (O) vector **||** (O) escalar	Verdadero si alguno es verdadero
~=	Distinto de	**~** (NO)	Negación del operando

Orden de precedencia	
Precedencia	**Operación matemática**
Primero	Paréntesis. Para paréntesis anidado (paréntesis dentro de paréntesis), el más interno es el que primero se ejecuta
Segundo	Exponenciación
Tercero	Multiplicación, división
Cuarto	Suma y resta

Opciones del elemento de formato		
Formato	**Descripción**	**Ejemplo**
%d	Entero (ej. 17)	`>> 17`
%f	Notación punto fijo (decimales)	`>> 17.0213`
%e	Notación exponencial	`>> 1.70213e+001`
%c	Carácter (letra o símbolo)	`>> t`
%s	Cadena de caracteres (Texto)	`>> texto`
\n	Salto de línea (Se utiliza al final del texto o para separar el texto en diferentes líneas)	`texto` `>>`

Funciones matemáticas elementales		
Función	**Descripción**	**Ejemplo**

sqrt(x)	Raíz cuadrada	`>> sqrt(81)`
		`ans=`
		`9`

exp(x)	Exponencial (e^x)	`>> exp(5)`
		`ans=`
		`148.4132`

abs(x)	Valor absoluto	`>> abs(-24)`
		`ans=`
		`24`

log(x)	Logaritmo natural
	Logaritmo de base *e* (ln)
		`>> log(1000)`
		`ans=`
		`6.9078`

log10(x)	Logaritmo en base 10	`>> log10(1000)`
		`ans=`
		`3.0000`

factorial(x)	Función factorial $x!$
	(x debe ser un entero positivo)
		`>> factorial(5)`
		`ans=`
		`120`

sin(x)	Seno del ángulo x
	(x en radiantes)
		`>> sin(pi/6)`
		`ans=`
		`0.5000`

sind(x)	Seno del ángulo x
	(x en grados)
		`>> sind(30)`
		`ans=`
		`0.5000`

cos(x)	Coseno del ángulo x
	(x en radiantes)
		`>> cos(pi/6)`
		`ans=`
		`0.8660`

cosd(x)	Coseno del ángulo x
	(x en grados)
		`>> cosd(30)`
		`ans=`
		`0.8660`

tan(x)	Tangente del ángulo x
	(x en radiantes)
		`>> tan(pi/6)`
		`ans=`
		`0.5574`

tand(x)	Tangente del ángulo x
	(x en grados)
		`>> tand(30)`
		`ans=`
		`0.5574`

<table>
<tr><th colspan="5">Comandos de Entrada y Salida</th></tr>
<tr><th></th><th>Comando</th><th>Descripción</th><th colspan="2">Ejemplo</th></tr>
<tr><td>Entrada</td><td>variable=input('Mensaje')</td><td>Se muestra el texto ingresado dentro de las comillas simples.</td><td>>> x = input('Ingresa x: ')
Ingresa el valor de x: |</td><td>Muestra el texto "Ingresa x:", texto que estaba entre las comillas simples. Queda en espera a recibir un valor. Puede ser en escalar, vector o matriz.</td></tr>
<tr><td></td><td>datos = xlsread('Libro1.xlsx', 'Hoja1', 'Rango Celdas')</td><td>Abre el archivo llamado Libro1.xlsx y se posiciona en la Hoja1 de ese libro, guardando en la variable datos la matriz correspondiente al Rango de Celdas.</td><td>>> z = xlsread('Prueba.xlsx', 'Hoja1', 'A1:C3')</td><td>Guarda la matriz obtenida del rango A1:C3 (3 x 3) del archivo Pruena.xlsx y la Hoja1 en una variable llamada z.</td></tr>
<tr><td rowspan="4">Salida</td><td>disp(nombre de variable)</td><td>Permite mostrar el contenido de la variable.</td><td>>> disp([5 9 1; 7 2 4])

5 9 1
7 2 4</td><td>Muestra la matriz de 2 x 3, en el formato por defecto en MATLAB.</td></tr>
<tr><td>disp('Mensaje')</td><td>Muestra la información que está dentro de las comillas simples.</td><td>>> disp('Problema 1')

Problema 1</td><td>Muestra "Problema 1" texto dentro de comillas simples</td></tr>
<tr><td>fprintf('Mensaje')</td><td>Muestra la información que está dentro de las comillas simples.</td><td>>> fprintf('Problema 1')

Problema 1>></td><td>Muestra "Problema 1" texto dentro de comillas simples</td></tr>
<tr><td>fprintf('Texto %5.2f texto adicional', nombre_variable)</td><td>Elementos del formato '%5.2f'
El símbolo % marca el lugar donde se insertará el número dentro del texto.
El nombre de la variable cuyo valor será visualizado (nombre_variable)
El primer número (5) indica cuántos números son antes del punto decimal.
El segundo número (2) indica el número de decimales a mostrar.
La letra final (f) indica el carácter de conversión y es obligatorio.</td><td>>> a = 17;
>> fprintf('El valor de la variable a es %d unidades\n', a)

El valor de la variable a es 5 unidades
>>

>> b = 170000;
>> fprintf('El valor de la variable b es %.2e unidades\n', a)

El valor de la variable b es 1.70e+005 unidades
>></td><td>Muestra el texto y el valor de la variable a con el formato indicado (valor entero).

Muestra el texto y el valor de la variable a con el formato indicado (notación científica con dos decimales).</td></tr>
</table>

Creación de variables

Notas:

- El símbolo **=** se llama operación de asignación.
- Para no visualizar el valor de una variable o instrucción incluir punto y coma (**;**) al final de cada sentencia de asignación.

Reglas nombre de variables:

- Longitud máxima: 63 caracteres.
- Debe empezar por una letra
- Pude contener letras, dígitos y el carácter de guion bajo (**_**)
- Distingue entre mayúsculas y minúsculas. Por ejemplo: **AA, Aa, aA** y **aa** son nombres de cuatro variables diferentes
- No poner nombres de variables reservadas por Matlab (**cos, sin, exp, sqrt**, etc)

Variable	Método de Creación	Ejemplo	Descripción

Escalar	**nombre_variable** = Valor numérico o expresión computable	`>> a = 5`
		`ans =`
		`5`
			Se asigna el número 5 a la variable **a**.

Vector a partir de serie de datos	**nombre_variable** = [elementos del vector]
	
	Los elementos del vector pueden ser separados por coma (,) o espacios.
		`>> years = [1984, 1987, 1991, 2005]`
		`years =`
		`1984 1887 1991 2005`
			Se asignan la lista de años al vector fila **years**.

Vector con distancia constante	**nombre_variable = [xi:d:xf]**
	
	Se debe especificar el primer término (**xi**), la distancia (**d**) y el último término (**xf**).
	Si no se especifica la distancia por defecto es de 1.
		`>> x = [1:2:13]`
		`x =`
		`1 3 5 7 9 11 13`
			Se asignan los elementos desde 1 hasta 13 con distancia de 2 al vector fila **x**.

	nombre_variable = **linspace(xi,xf,n)**
	
	Se debe especificar el primer término (**xi**), el último término (**xf**) y el número de términos (**n**).
	Si no se especifica el número de términos es de 100.
		`>> va = linspace(0,8,6)`
		`x =`
		`0 1.6000 3.2000 4.8000 6.4000 8.0000`
			Se asignan los 6 elementos desde 0 hasta 8 con distancia constante al vector fila **va**.

Matrices	**nombre_variable** = [elementos de la 1ª fila; elementos de la 2ª fila: elementos de la 3ª final; …; elementos de la última fila]
	
	Los elementos entre filas deben ser separados por punto y coma (;).
	Todas las filas deben de tener el mismo número de elementos,
		`>> B = [5 35 43; 4 76 81; 21 32 40]`
		`B =`
		`5 35 43`
		`4 76 81`
		`21 32 40`	

Funciones para vectores y matrices		
Comando	**Descripción**	**Ejemplo**

length(vector)	Regresa el tamaño (longitud) de un vector	`>> a = [1, 2, 3];`
		`>> length(a)`
		`ans =`
		`3`

size(matriz)	Regresa el tamaño (número de filas y columnas) de una matriz	`>> B = [35 43; 21 40];`
		`>> size(B)`
		`ans =`
		`2, 2`

sum(vector)	Realiza la operación de sumar todos los elementos del vector	`>> a = [1, 2, 3, 4];`
		`> > sum(a)`
		`ans =`
		`10`

sum(**sum**(matriz))	Realiza la operación de sumar todos los elementos de la matriz	`>> B = [1 2; 3 4];`
		`>> sum(sum(B))`
		`ans =`
		`10`

prod(vector)	Realiza la operación de multiplicar todos los elementos del vector	`>> a = [1, 2, 3, 4];`
		`> > prod(a)`
		`ans =`
		`24`

prod(**prod**((matriz))	Realiza la operación de multiplicar todos los elementos de la matriz	`>> B = [1 2; 3 4];`
		`>> prod(prod(B))`
		`ans =`
		`24`

Derivación e Integración		
Comando	**Descripción**	**Ejemplo**
syms variable1 variable2	Crea las variables simbólicas **variable1** y **variable2** Generalmente se usa para después hacer una derivada o integral	`>> syms x`

diff(función, variable, valor)	Permite realizar la derivada con respecto a la **variable** simbólica introduciendo una **función** que contenga dicha **variable**, de caso contrario el resultado será **0**. El **valor** se utiliza para evaluar la derivada. Si no se especifica el **valor**, se realiza la derivación implícita.	`>> dif(x^2+5, x)`
		`ans =`
		`2*x`

int(función, variable, limiteInferior,limiteSuperior)	Permite realizar la integral con respecto a la **variable** simbólica, introduciendo una **función** que contenga dicha **variable**. Los límites inferior y superior se utilizan para evaluar la integral. Si no se especifican los límites, se realiza la integración implícita.	`>> int(x^2+5, x)`
		`ans =`
		`x^3 + 5*x`

Programación en MATLAB			
	Estructura	**Descripción**	**MATLAB**
Sentencia condicional	**if-end**	**if** expresión_condicional_1 Grupo_de_instrucciones_1 **end**	Si la expresión es verdadera (**1**), el programa continúa ejecutando el Grupo de Instrucciones 1. Por el contrario, si la expresión es falsa (**0**) al ser evaluada, el programa salta el primer Grupo de Instrucciones y sale de la sentencia **if-else**.

	if-else-end	**if** expresión_condicional_1
		  Grupo_de_instrucciones_1
		else
		  Grupo_de_instrucciones_2
		end
			Ejecuta uno entre dos Grupos de Instrucciones posibles en función de la evaluación de la condición lógica. Trata de discriminar entre dos opciones.

	if-elseif-else-end	**if** expresión_condicional_1
		  Grupo_de_instrucciones_1
		elseif expresión_condicional_2
		  Grupo_de_instrucciones_2
		else
		  Grupo_de_instrucciones_3
		end
			Si la primera expresión es verdadera (**1**), el programa continúa ejecutando el Grupo de Instrucciones 1. Posteriormente acabará (sin ejecutar el resto de Grupo de Instrucciones). Si la condición evaluada es falsa, el programa saltará a la siguiente sentencia **elseif**. Si la condición que evalúa **elseif** es verdadera (**1**), entonces e ejecutará el Grupo de Instrucciones 2 y acabará. Por el contrario, si la condición de **elseif** resulta falsa (**0**), el programa saltará al **else** y ejecutará el Grupo de Instrucciones 3.
			Puede tener más de un comando **elseif**, sigue la misma lógica presentada anteriormente.

	switch-case	**switch** expresión
		case valor1
		  Grupo_de_instrucciones_1
		case valor2
		  Grupo_de_instrucciones_2
		case valor3
		  Grupo_de_instrucciones_3
		otherwise
		  Grupo_de_instrucciones_4
		end
			El valor de la expresión **switch** se compara con los valores de cada comando **case**. Si encuentra una coincidencia, entonces se ejecutan las instrucciones que siguen a dicho comando **case** (sólo el Grupo de Instrucciones que se encuentra después de cada comando **case**).
			Si hay más de una coincidencia se atenderá a la primera.
			Si no hay coincidencias se ejecutan las instrucciones justo después del comando **otherwise**, si ha sido declarado (es opcional).
			Puede tener más de un valor con el que establecer coincidencias.

Comandos Especiales para ciclos for y while	**break**	**for** k = f:s:t
		  Grupo_de_instrucciones_1
		  break
		  Grupo_de_instrucciones_2
		end
			Termina de forma completa la ejecución del ciclo.
			Dentro del ciclo anidado, terminará la ejecución del ciclo anidado.
			Se suele utilizar dentro de sentencias condicionales. Donde proporciona una forma de acabar el proceso iterativo en función de alguna condición establecida.

	continue	**for** k = f:s:t
		  Grupo_de_instrucciones_1
		  continue
		  Grupo_de_instrucciones_2
		end
			Normalmente forma parte de alguna sentencia condicional. Cuando se llega a este comando, no sigue ejecutando el resto de los comandos del ciclo, salta a la sentencia **end** y continua la siguiente iteración.

<table>
<tr><th colspan="4">Programación en MATLAB</th></tr>
<tr><th></th><th>Estructura</th><th>Descripción</th><th>MATLAB</th></tr>
<tr><td rowspan="2">Ciclos</td><td>for-end</td><td>for k = f:s:t
  Grupo_de_instrucciones
end</td><td>donde
k: la variable del ciclo
f: valor inicial de k (valor que tomará en el primer paso)
s: El incremento de k (después de cada paso)
t: último valor que puede tomar k (valor final de k, fin del ciclo)

Notas:
- La variable índice del ciclo puede tener cualquier nombre (normalmente se utiliza i, j, k, m o n)
- El incremento puede ser negativo
- Si el valor del incremento se omite, por defecto será 1.
- Si f es igual a t, se ejecutará sólo una vez.
- La variable k también puede tomar valores específicos. Para ello hay que introducir un vector de valores que se deseen. Por ejemplo, k = [7 9 -1 3 3 5].</td></tr>
<tr><td>while-end</td><td>while expresión_condicional
  Grupo_de_instrucciones
end</td><td>Cuando el programa llega a esta estructura, comprueba la expresión. Si es falsa (0), salta al final del ciclo (end). Por el contrario, si es verdadera (1), ejecuta el grupo de comandos dentro de la estructura while y la sentencia end. Seguidamente retrocede de nuevo a la instrucción while y comprueba de nuevo la expresión hasta que sea falsa.

Notas:
- La expresión del comando while debe incluir al menos una variable.
- Las variables de la expresión deben tener valores asignados previamente.
- Al menos una variable de la expresión debe cambiar de valor por la ejecución del grupo de instrucciones entre la sentencia while y end. En caso contrario, si ninguna de las variables cambia, la condición siempre se mantendrá igual (ciclo infinito).</td></tr>
<tr><td colspan="2">Ciclos anidados</td><td>for k = 1:n
  for h = 1:m
  Grupo_de_instrucciones
  end
end</td><td>Cada vez que k incrementa en 1, el ciclo anidado (h) se ejecuta m veces. En total, el grupo de instrucciones internas se ejecuta n x m veces.

Nota:
- Los ciclos y sentencias condicionales se pueden anidar unos dentro de otros.
- No existe límite sobre número de ciclos y sentencias condicionales que se pueden anidar.
- Cada sentencia if, case, for y while debe tener su correspondiente sentencia de finalización end.</td></tr>
</table>

Gráficas bidimensionales (2-D)			
Comando	**Descripción**	**Ejemplo**	
plot(x, y, 'especficadores de línea')	Los argumentos **x** e **y** son vectores y deben de tener el mismo número de elementos. Eje x (horizontal), eje y(vertical) La curva se construye mediante segmentos de recta que tienen los puntos cuyas coordenadas están definidas por los elementos **x** e **y**	`>> x = [1 2 3 5 7 7.5 8]` `>> y = [2 6.5 7 7.5 4 6]`	Coordenadas de **x** y **y** para graficar
		`>> plot(x, y)`	Línea azul que conecta los puntos, sin marcadores
		`>> plot(x, y, 'r')`	Línea roja que conecta los puntos, sin marcadores
		`>> plot(x, y, '--y')`	Línea amarilla discontinua que conecta los puntos, sin marcadores
		`>> plot(x, y, '*')`	Puntos con marcadores tipo asterisco (sin línea)
		`>> plot(x, y, 'g:d')`	Línea verde punteada que une puntos con marcadores en forma de diamante
fplot('función', límites, 'especificadores de línea')	Representa gráficamente una función de la forma y = f(x) entre unos límites especificados por el usuario	`>> plot('x^2+4*sin(2*x)-1', [-3,3])`	La función que se quiere representar se construye dentro de la cadena utilizando cualquier letra como variable. Los límites se especifican mediante un vector **[xmin, xmax,ymin,ymax]**
subplot(f, c, p)	Muestra en forma matricial la Ventana de Gráficos de acuerdo al número de filas (**f**) y columnas (**c**) indicado en cada posición (**p**). Por ejemplo, si hay dos gráficas se pueden agrupar como 2 filas y 2 renglones, entonces los comandos para cada gráfica serán: **subplot(2,1,1)** con la gráfica que se desea en la primer posición y **subplot(2,1,2)** para el gráfico en la segunda posición.		

Estilo de línea	Especificador
sólida	**-**
discontinua	**--**
punteada	**:**
rayas y puntos	**-.**

Color de línea	Especificador
rojo	r
verde	g
azul	b
cian	c
magenta	m
amarillo	y
negro	k

Tipo de marcador	Especificador
signo más	+
círculo	o
asterisco	*
punto	.
cuadrado	s
diamante	d
estrella cinco puntas	p
estrella seis puntas	h

Opciones para gráficas	
Comando	**Descripción**
hold on	Mantiene la Ventana de Gráficos activada. Ingresar los comandos **plot** que se desean para graficar en la misma ventana con los mismos ejes.
hold off	Sin más gráficas sobre la misma ventana.
grid on	Añade cuadricula a la gráfica
grid off	Elimina cuadricula a la gráfica
axis([xmin, xmax, ymin, ymax])	Crea los ejes basándose en los valores máximo y mínimo de los valores posibles de x e y.
xlabel('texto')	Título en el eje x
ylabel('texto')	Título en el eje y
title('texto')	Título de la gráfica
legend('texto1', 'texto2',…)	Leyenda en la representación gráfica. Cada texto corresponde a una gráfica

Gráficas tridimensionales (3-D)			
Comando	**Descripción**	**Ejemplo**	

plot3(x, y, z, 'especificadores de línea')	Los tres vectores con las coordenadas de los puntos deben tener el mismo número de elementos.	`>> t=0:0.1:6*pi;`
		`>> x=sqrt(t)*.*sin(2*t);`
		`>> y=sqrt(t)*.*cos(2*t);`
		`>> z = 0.5*t;`
		`>> plot3(x,y,z)`
			Representar gráficamente las coordenadas **x**, **y**, y **z**.

[X, Y] = meshgrid(x, y)	Crea un conjunto de puntos correspondientes al plano *x-y*.
	
	La densidad de la rejilla (número de puntos utilizados para definir el dominio) debe ser definida por el usuario.
	
	X e **Y** contienen las coordenadas de todos los puntos x e y respectivamente
		`>> x = -1:3;`
		`>> y = 1:4;`
		`>> [X, Y] = meshgrid(x, y)`
		`X =`
		`-1 0 1 2 3`
		`-1 0 1 2 3`
		`-1 0 1 2 3`
		`-1 0 1 2 3`
		
		`Y =`
		`1 1 1 1 1`
		`2 2 2 2 2`
		`3 3 3 3 3`
		`4 4 4 4 4`
			- **x** es un vector que representa el dominio de *x*
			- **y** es un vector que representa el dominio de *y*
			- **X** es la matriz de coordenadas *x* de la rejilla de puntos.
			- **Y** es la matriz de coordenadas *y* de la rejilla de puntos.

		`Z = X.*Y.^2./(X.^2+Y.^2)`	Valor de **Z** para cada punto de la rejilla.
mesh(X, Y, Z)	Gráfico de malla	`>> mesh(X, Y, Z)`	NA
surf(X, Y, Z)	Gráfico de superficie	`>> surf(X, Y, Z)`	NA
meshz(X, Y, Z)	Gráfico de malla con cortina (dibuja una cortina alrededor de la malla)	`>> meshz(X, Y, Z)`	NA
meshc(X)	Gráfico de malla con contorno (dibuja un contorno debajo de la superficie)	`>> meshc(X, Y, Z)`	NA
surfc(X)	Gráfico de superficie con contorno (dibuja un contorno debajo de la superficie)	`>> surfc(X, Y, Z)`	NA
surfl(X)	Gráfico de superficie con alumbrado	`>> surfl(X, Y, Z)`	NA
contour3(X, Y, Z, n)	Gráfico de Contorno 3-D n es el número de niveles de contorno (opcional)	`>> contour3(X, Y, Z, n)`	NA
contour(X, Y, Z, n)	Gráfico de contorno 2-D (dibuja proyecciones de niveles de contorno sobre el plano *x-y*) n es el número de niveles de contorno (opcional)	`>> contour3(X, Y, Z, n)`	NA