

### 3. Operaciones básicas con escalares, vectores y matrices en Matlab

Cualquier cálculo matemático, por lo general, se especifica como una combinación de constantes numéricas y variables, que pueden ser escalares, vectores o matrices y utilizando operadores de tipo aritmético básico.

Las operaciones básicas entre dos escalares o matrices en Matlab son:

Suma	+
Resta	-
Multiplicación	*
División	/
Exponenciación	^
Agrupación	( )

Como ya se comentó anteriormente, existe también la operación especial de **asignación**, que se representa a través del símbolo de = (igual), de la forma:

$$variable = expresión aritmética ;$$

Es muy importante recordar que en la operación de asignación siempre *primero* se evalúa la parte derecha es decir, la de la expresión aritmética y, *segundo* el resultado se asigna a la parte de la izquierda es decir a la variable que toma el resultado

Aplicando esto último tendríamos que una expresión como:

$$x = x + 5 ;$$

Es totalmente válida ya que no importa que la variable x aparezca en ambos lados de la expresión pues *primero* se evalúa la parte derecha con el valor que tenga x en ése momento y, *segundo* el resultado de la operación **sustituye** al valor de x (el valor con el cuál se hizo la suma desaparece ya que sólo puede contenerse un único valor en x y que sería el último valor calculado).

#### 3.1 Precedencia de las operaciones aritméticas

Usualmente combinamos varias operaciones en una sola expresión aritmética por lo tanto es importante conocer en que orden se realizarán las operaciones de cálculo con el fin de determinar si se realizan de acuerdo al modelo algebraico (o fórmula) que se esté utilizando.

La siguiente tabla muestra la precedencia (jerarquía) de las operaciones mostrándolas desde la de mayor prioridad a la de menor prioridad:

Agrupación	( )	(adentro hacia afuera)
Exponenciación	^	(izquierda a derecha)
Mult. y div.	* /	(izquierda a derecha)
Suma y resta	* -	(izquierda a derecha)

Habrá que tener precaución al convertir ecuaciones de tipo algebraico en expresiones de Matlab. Una buena sugerencia es agregar tantos paréntesis como sea necesario para asegurarse de que los cálculos se harán en el orden deseado. Inclusive si una expresión es muy larga y compleja, se recomienda que la dividida en varias instrucciones.

### 3.2 Operaciones con vectores y matrices como arreglos

En muchas ocasiones es necesario efectuar operaciones entre dos vectores o matrices pero con la característica de que el cálculo deberá realizarse **elemento por elemento** en lugar de tomar sus valores como un todo.

De hecho es como si tomáramos a cada elemento como si fuera un escalar y se hiciera una operación con otro escalar porque son operaciones de uno a uno.

Para indicar que queremos realizar *operaciones de tipo arreglo* entre vectores o matrices utilizamos **un punto** que precede al símbolo aritmético, de acuerdo con la siguiente tabla:

Suma	+
Resta	-
Multiplicación	.*
División	./
Exponenciación	.^
Agrupación	( )

En el caso de la suma y resta de vectores y matrices, las operaciones de *arreglos* y las de *matrices* son iguales, así que no es necesario hacer ninguna distinción entre ellas.

En cambio, las operaciones de *arreglos* para multiplicación, división y exponenciación son diferentes a las correspondientes de *matrices* como un todo, así que se necesita especificar el punto antes del símbolo.

Las operaciones elemento por elemento, u operaciones de *arreglos*, se aplican a operaciones entre dos vectores o matrices **del mismo tamaño**. También es posible que se puedan realizar entre un escalar y un vector o matriz.

### Ejercicios

Escriba los comandos Matlab equivalentes de las siguientes expresiones matemáticas. Suponga que las variables son escalares ya definidas previamente; y observe que se cumplan las reglas de precedencia:

$$1) \quad v = \frac{4}{3}pr^3$$

$$2) \quad y = v_0t + \frac{1}{2}at^2$$

$$3) \quad r_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$4) \quad r_p = \frac{1}{\frac{1}{r_1} + \frac{1}{r_2} + \frac{1}{r_3}}$$

$$5) \quad x = \frac{\left(\frac{a^2}{3} + 5\right)b^3}{a + b}$$

A continuación, suponga que se tienen los vectores:

$$x = [1 \ 3 \ 5 \ 7] \quad e \quad y = [2 \ 4 \ 6 \ 8]$$

Mencione el contenido del vector z después de cada una de las siguientes operaciones:

- 6) `>>z = x + y ;`
- 7) `>>z = 2 * x - y ;`
- 8) `>>z = y - x + 5 ;`
- 9) `>>z = y .* x ;`
- 10) `>>z = 2 * x / 3 .* y ;`
- 11) `>>z = 3.^y + x ;`
- 12) `>>z = x ./ y + y .* x ;`

### 3.3 Valores y operaciones especiales en Matlab

Con el fin de facilitar la representación de fórmulas matemáticas y de hacer más eficiente la manera en que se realizan los cálculos, Matlab incluye varias constantes predefinidas y operaciones especiales. Algunas de ellas son las siguientes:

**pi** Representa el valor de  $\pi$

- i , j** Representan la unidad imaginaria  $\sqrt{-1}$ . Se utiliza con frecuencia al representar números complejos.
- inf** Representa *infinito*,  $\infty$ , que por lo general ocurre al dividir entre cero.
- NaN** Indica *Not-a-Number* que significa *No-es-un-número* y que ocurre cuando una expresión no está definida, como en la división de cero entre cero.
- ans** Es la respuesta más reciente, no almacenada en ninguna variable.
- eps** Indica la precisión de punto flotante de la computadora. Es el valor más pequeño en el que dos magnitudes numéricas pueden diferir una de otra.
- %** Representa *comentario*. Sirve para documentar las operaciones y programas. Lo que se escriba a la derecha del % se ignora por Matlab.
- \** Indica la operación aritmética de *división izquierda* para vectores y matrices (funciona como la división normal pero se divide el argumento de la derecha entre el de la izquierda). Por ejemplo  $z = x \ y$ .
- ./** Indica la operación de división izquierda pero de *tipo arreglo*.

### 3.4 Operadores Relacionales y Lógicos

Matlab cuenta con los siguientes **operadores relacionales**:

<	Menor que
<=	Menor o igual que
>	Mayor que
>=	Mayor o igual que
==	Igual. Idéntico.
~=	No igual. Distinto.

Estos operadores sirven para efectuar comparaciones, elemento a elemento, entre matrices de igual tamaño y el resultado será otra nueva matriz de iguales dimensiones. Cada elemento de la matriz resultante contendrá un valor de 1 si la comparación es verdadera; de lo contrario, contendrá un valor de 0 en la posición correspondiente.

A los lados de este tipo de operadores se pueden colocar matrices o expresiones de matrices para proceder a su comparación. Una expresión que contiene algún operador relacional se le conoce como **expresión lógica** porque su resultado será una matriz de unos y ceros que representan valores **verdaderos (1)** y **falsos (0)**.

Por ejemplo, si se tiene la expresión  $x <= y$ , en la cual  $x$  e  $y$  son escalares, el valor que regresa esta expresión será **1** (verdadero) sólo si  $x$  es menor o igual que  $y$ ; de lo contrario, se obtiene **0** (falso) cuando  $x$  es mayor que  $y$ .

Si en lugar de escalares  $x$  e  $y$  fueran vectores, por ejemplo:

$$x = [ 2 \ 4 \ 6 ]; \quad e \quad y = [ 1 \ 4 \ 3 ];$$

Entonces, el resultado de la expresión  $x <= y$ , es el vector  $[ 0 \ 1 \ 0 ]$ , en tanto que el resultado de  $x \sim y$  es el vector  $[ 1 \ 0 \ 1 ]$ , y el de  $x >= y$  proporciona  $[ 1 \ 1 \ 1 ]$ .

Matlab también cuenta con **operadores lógicos** que se utilizan para concatenar expresiones lógicas completas, y son:

~	Not. Negación ( no ).
&	And. Conjunción ( y ).
	Or. Disyunción ( o ).

Al operador **&** se le interpreta como operador de conjunción pues representa *unión*, el operador **|** se interpreta como disyunción pues indica una de dos *alternativas* y al operador **~** se le conoce como *inversor de estado* ó negación

Si  $A$  y  $B$  son dos expresiones lógicas completas, su tabla de combinaciones es:

A	B	$\sim A$	$A   B$	$A \& B$
0	0	1	0	0
0	1	1	1	0
1	0	0	1	0
1	1	0	1	1

Los operadores lógicos también presentan un orden de precedencia. Su jerarquía, listados del mayor nivel al menor nivel, es tal como se mostró en la lista anterior; es decir:  $\sim$ ,  $\&$  y  $|$ .

Al encontrar dos operadores lógicos de igual nivel de precedencia, se resuelven de la misma forma que los aritméticos; es decir, de izquierda a derecha.

En Matlab, al efectuar una operación de tipo lógico, cualquier valor distinto de cero será considerado como verdadero; los valores idénticos con cero son tomados como falsos.

Desde luego que se puede utilizar el operador de agrupación (paréntesis) para modificar el orden de jerarquía de alguna expresión lógica.

De igual forma, en una expresión de tipo lógico completa, se pueden realizar operaciones de tipo aritmético con los argumentos que tomarán los operadores relacionales.

## Ejercicios

Si se tiene que  $x=5$ ,  $y=2$ ,  $z=0$ , determine si las siguientes expresiones lógicas son verdaderas o falsas:

- 1)  $y-z \sim = 0$
- 2)  $x | y \& z$
- 3)  $x^{\wedge}z \& \sim(y-2)$
- 4)  $x >= y \& z \sim = x$
- 5)  $y < x | \sim (z < 3)$
- 6)  $\sim (x-y <= 0) \& \sim z$
- 7)  $x \& (y-4 \sim = 0)$
- 8)  $\sim z | \sim (y-x == -3)$
- 9)  $x \& y | \sim x | \sim y$
- 10)  $\sim ( (2*x) \& \sim(y-2) \& (z-1) )$

### 3.5 Funciones en Matlab

El concepto de **función** se interpreta de las tres formas siguientes en Matlab:

#### ❖ Función *algebraica*

El concepto tradicionalmente matemático:  $y = f(x)$  ó  $z = f(x, y)$ ; que es cuando existe una relación de pareja, uno-a-uno, entre variables y que da lugar a valores en un *dominio* y su *contradominio*.

#### ❖ Función *predefinida o de biblioteca*

Se refiere a un **procedimiento** o **programa** que ha sido almacenado previamente en Matlab y que está disponible al usuario para realizar cálculos de un cierto tipo específico. En su gran mayoría, estas funciones operan *recibiendo* alguna clase de **argumentos** ó **parámetros** y *devolviendo* ó *regresando* como resultado algún **tipo específico de dato**.

Estas funciones pueden tener cero, uno ó múltiples argumentos. Por ejemplo, **pi** es una función que no recibe ningún argumento y regresa como resultado automáticamente el valor 3.141592... . Cuando una función predefinida o de biblioteca requiere parámetros, éstos se especifican dentro de los paréntesis ( ) que se colocan a la derecha del nombre de función y puestos en el orden correcto:

```
>>nombre_de_función( arg1, arg2, ... );
```

Cuando una función recibe más de un argumento, el orden en que se den debe ser el correcto para dicha función. Una "**llamada**" a una función puede ser, a la vez, argumento ó parámetro de otra función. A esto se le llama **anidar funciones**, Por ejemplo:

```
>>y = sin( abs( x ) );
```

#### ❖ Funciones *definidas por el usuario*

A pesar de que Matlab incluye muchas funciones predefinidas o de biblioteca, habrá ocasiones en que el usuario tiene que escribir sus propios procedimientos o programas para resolver sus problemas particulares. A esto se le llama una *función definida por el usuario*.

En este caso, es el propio usuario quién define que parámetros o argumentos recibirá su función y que tipo de resultado entregará. De igual forma, se recomienda que el usuario defina un cierto directorio en donde se guarden los algoritmos desarrollados y dicho directorio sea agregado a la ruta de búsqueda del Matlab.

Las funciones definidas por el usuario se almacenan en un archivo de tipo texto que Matlab identifica con la extensión **.m** . Se dice que dichos archivos son **tipo script** pues están constituidos por instrucciones escritas en notación Matlab y que son **compilados** sólo la primera vez que son "invocados", permaneciendo almacenados en la memoria mientras dure la sesión de trabajo actual.

### 3.5.1 Las funciones en Matlab

Hasta ahora se han estudiado las operaciones matemáticas básicas, pero en un gran número de ocasiones se desea efectuar una operación rutinaria que no cae en dicha categoría. Matlab, en su totalidad, cuenta con cientos de funciones para realizar cálculos matemáticos de diversa índole. En este tema sólo se presentan las funciones matemáticas de mayor utilidad y uso frecuente.

Por ejemplo, si se desea calcular el coseno de un ángulo  $x$  y almacenar su valor en  $y$ , lo hacemos con la función **cos** :

```
>>y = cos( x );
ó
>>y = cos( x * pi/180 ); %(si x está en grados)
```

En esta segunda versión, la conversión a radianes se podría haber realizado previamente usando otra instrucción y con una nueva variable:

```
>>r = x * pi/180 ;
>>y = cos( r );
```

Es muy importante tener en cuenta que en muchas ocasiones, las funciones requieren que los argumentos estén dados en una cierta unidad específica.

A manera de ejemplo:

Todas las **funciones trigonométricas** operan basadas en parámetros **en radianes** y no en grados.

Para el ejemplo anterior, lo más común es que  $x$  sea un escalar aunque también podría ser un vector o matriz; en este caso, la función se aplicará elemento por elemento a cada uno de los valores que contenga dicho arreglo. Para matrices algunos comandos los ejecuta para cada columna, produciendo vectores fila.

### 3.5.2 Funciones matemáticas comunes

- ❖ Valor Absoluto de  $x$

```
[36] >>abs ( x );
```

- ❖ Raíz cuadrada de  $x$

```
[37] >>sqrt ( x );
```

- ❖ Exponencial base  $e$  ( $e^x$ )

```
[38] >>exp ( x );
```

- ❖ Redondear  $x$  al entero mas cercano

```
[39] >>round ( x );
```

- ❖ Truncar decimales de  $x$

```
[40] >>fix ( x );
```

- ❖ Redondear al entero inferior a  $x$

```
[41] >>floor ( x );
```

- ❖ Redondear al entero superior a  $x$

```
[42] >>ceil ( x );
```

- ❖ Residuo entero de  $x / y$ . Función **Módulo**.

```
[43] >>rem ( x , y ); ó mod ( x , y );
```

- ❖ Logaritmo natural de  $x$  con base  $e$

```
[44] >>log ( x );
```

- ❖ Logaritmo decimal de  $x$

```
[45] >>log10 ( x );
```

- ❖ Signo indicador del valor de  $x$

```
[46] >>sign ( x );
```

Esta función devuelve  $-1$  si  $x$  es menor que cero, regresa  $0$  si  $x$  es igual con cero y  $1$  si  $x$  es mayor que cero.

### 3.5.3 Funciones trigonométricas

- ❖ Seno de  $x$

```
[47] >>sin ( x );
```

- ❖ Arco seno de  $x$

```
[48] >>asin ( x );
```

- ❖ Coseno de  $x$

```
[49] >>cos ( x );
```

- ❖ Arco coseno de  $x$

```
[50] >>acos ( x );
```

- ❖ Tangente de  $x$

```
[51] >>tan ( x );
```

- ❖ Arco tangente de  $x$

```
[52] >>atan ( x );
```

- ❖ Secante de  $x$

```
[53] >>sec ( x );
```

- ❖ Arco secante de  $x$

```
[54] >>asec ( x );
```

- ❖ Cosecante de  $x$

```
[55] >>csc ( x );
```

- ❖ Arco cosecante de  $x$

```
[56] >>acsc ( x );
```

- ❖ Cotangente de  $x$

```
[57] >>cot ( x ) ;
```

- ❖ Arco cotangente de  $x$

```
[58] >>acot ( x ) ;
```

### 3.5.4 Funciones matemáticas adicionales

- ❖ Sumar los elementos de un vector o matriz  $x$

```
[59] >>sum ( x ) ;
```

- ❖ Ordenar los elementos de un vector o matriz  $x$

```
[60] >>sort ( x ) ;
```

- ❖ Valor del elemento más grande de un vector o matriz  $x$

```
[61] >>max ( x ) ;
>>[v p] = max ( x )
```

- ❖ Valor del elemento más pequeño de un vector o matriz  $x$

```
[62] >>min ( x ) ;
>>[v p] = min ( x )
```

- ❖ Máximo común divisor de los enteros  $x$  e  $y$

```
[63] >>gcd ( x , y ) ;
```

- ❖ Mínimo común múltiplo de los enteros  $x$  e  $y$

```
[64] >>lcm ( x , y ) ;
```

- ❖ Parte real del número complejo  $x$

```
[65] >>real ( x ) ;
```

- ❖ Parte imaginaria del número complejo  $x$

```
[66] >>imag ( x ) ;
```

- ❖ Angulo de fase del complejo  $x$

```
[67] >>angle ( x ) ;
```

También el argumento puede ser un vector o una matriz de números complejos.

- ❖ Obtener el *conjugado* del número complejo  $x$

```
[68] >>conj ( x ) ;
```

- ❖ Obtener la *magnitud* del número complejo  $x$

```
[69] >>abs ( x ) ;
```

Observe que es igual a la función **abs** para el valor absoluto; lo que cambia es el *tipo de argumento*. A esto se le conoce como “**sobrecarga de funciones**”; es decir, diferente respuesta de una misma función ante diferentes tipos de argumentos de entrada. Es una especie de lo que en **POO** se le llama “**polimorfismo**”.

**Ejercicios**

Si se tiene que  $x = 5.71$ ,  $y = -2.43$ ,  $z = 0$ , determine el resultado de las siguientes expresiones:

- 1) `round ( x + y ) ;`
- 2) `sign ( y - x ) ;`
- 3) `floor ( x * y ) ;`
- 4) `ceil ( abs ( x * y ) ) ;`
- 5) `fix ( 1 : 0.25 : 10 ) ;`
- 6) `sqrt ( [3.56, 7.56, 2.71, 9.81] ) ;`
- 7) `rem ( fix ( abs ( x * y ) ) , ceil ( x ) ) ;`
- 8) `sum ( [2.45 7.32 4.5 6.66 ] ) ;`
- 9) `[v p] = min ( [ 8 6 3 7 4 2 9 ] ) ;`
- 10) `max ( [ 1 4 8 ; 5 2 4 ; 2 1 9 ] ) ;`
- 11) `gcd ( ceil ( fix ( x ) ) , abs ( fix ( y ) ) * 6 ) ;`
- 12) `angle ( 3 + 4i ) * (180 / pi ) ;`
- 13) `abs ( 3 + 4i ) * conj ( 4 - 2j ) ;`

**Notas:**