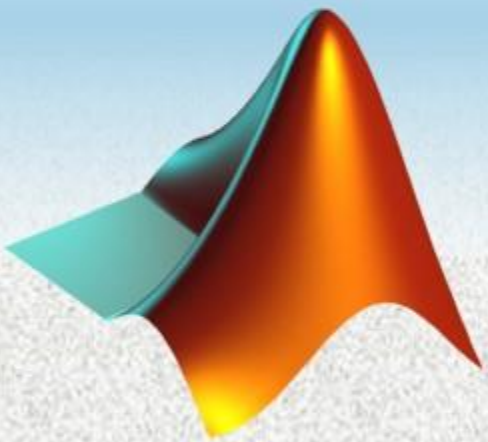
UNIVERSIDAD
NACIONAL DE
INGENIERÍA

MATLAB

R2016a



Maria Pimentel Herrera

uni.kernel@gmail.com

Elementos básicos de Matlab

Constantes

- Números enteros

  -49 -17 0 2 9 28 2007

- Números reales

  -18.8 17 20.28 2017.923

- Números complejos

  7+2i 23j $i, j = (-1)^{1/2}$

Formato numérico

- short — Formato corto decimal fijo con 4 dígitos después del punto decimal. (defecto)

3.1416

- long — Formato largo decimal fijo con 15 dígitos después del punto decimal para valores double y 7 dígitos después del punto decimal para valores single .

3.141592653589793

Formato numérico

- short e — Breve notación científica con 4 dígitos después del punto decimal.

3.1416e+00

- long e — Largo notación científica con 15 dígitos después del punto decimal para valores double y 7 dígitos después del punto decimal para valores single .

3.141592653589793e+00

Formato numérico

- short g — Formato corto, fijo decimal o notación científica, lo que es más compacto, con un total de 5 dígitos.

3.1416

- long g — Formato largo, fijo decimal o notación científica, lo que es más compacto, con un total de 15 dígitos para valores double y 7 dígitos para valores single .

3.14159265358979

Formato numérico

- short eng Notación ingeniería corta (el exponente es un múltiplo de 3) con 4 dígitos después del punto decimal.

3.1416e+000

- long eng Notación ingeniería de largo (el exponente es un múltiplo de 3) con 15 dígitos significativos.

3.141592653589 79e+000

Formato numérico

- \+ Formato de positivo/negativo con +, -y caracteres en blanco aparezca de positivo, negativo y cero elementos.

+

- hex Representación hexadecimal de un número binario de doble precisión.

400921fb54442d18

Formato numérico

- bank — Formato de moneda con 2 dígitos después del punto decimal.

  3.14

- rat — Relación de números enteros pequeños.

  355/113

https://www.mathworks.com/help/matlab/ref/format.html

s0100.m

- New M-file 

```
%aplicación de formato numérico
clc; clear;
n=input('ingrese un valor: ');    % log(2017.2803)
format;    %format short;
disp('format short'); disp(n);
format long; disp('format long'); disp(n);
format +; disp('format +'); disp(n);
format bank; disp('format bank'); disp(n);
format hex; disp('format hex'); disp(n);
format rat; disp('format rat'); disp(n);
```

Run o F5

```
ingrese un valor: log(2017.2803)
format short
    7.6095

format long
   7.609505497064441

format +
+

format bank
          7.61

format hex
   401e702235827ef9

format rat
    3683/484

>>
```

Maria Pimentel Herrera • 991997157 • 996783399

Formato de espaciado de línea

- compact Suprimir líneas en blanco exceso para mostrar salida más en una sola pantalla.

```
theta = pi/2
theta =
  1.5708
```

- loose Agregar líneas en blanco para que la salida sea más legible.

```
theta = pi/2

theta =

  1.5708
```

s0101.m

```
%aplicación de formato numérico
clc; clear;
n=input('ingrese un valor: ');   % pruebe para pi, exp(1)
format compact;
format short; disp('format short'); disp(n);
format long; disp('format long'); disp(n);
format short e; disp('format short e'); disp(n);
format long e; disp('format long e'); disp(n);
format short g; disp('format short g'); disp(n);
format long g; disp('format long g'); disp(n);
format short eng; disp('format short eng'); disp(n);
format long eng; disp('format longng eng'); disp(n);
format +; disp('format +'); disp(n);
format bank; disp('format bank'); disp(n);
format hex; disp('format hex'); disp(n);
format rat; disp('format rat'); disp(n);
```

Run o F5

```
ingrese un valor: exp(1)
format short
    2.7183
format long
   2.718281828459046
format short e
  2.7183e+000
format long e
    2.718281828459046e+000
format short g
       2.7183
format long g
          2.71828182845905
format short eng
     2.7183e+000
format longng eng
    2.718281828459046e+000
format +
+
format bank
          2.72
format hex
   4005bf0a8b14576a
format rat
     1457/536
>>
```

Maria Pimentel Herrera • 991997157 • 996783399

Cálculos básicos

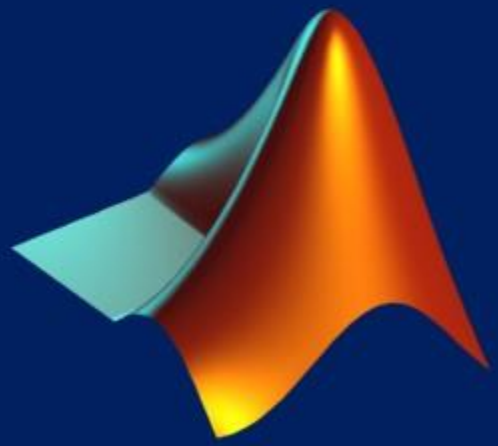
Operaciones aritméticas

- ^ Potenciación
- *, / Multiplicación y división
- +, - Suma y resta

Orden de prioridad de las operaciones: las expresiones se evalúan de izquierda a derecha; la operación potencia tiene el orden de prioridad más alto, seguida por multiplicación y división que tienen ambas igual prioridad y seguidas, finalmente, por suma y resta con igual prioridad.

Operaciones aritméticas

Se pueden emplear paréntesis para alterar esta ordenación, en cuyo caso la evaluación se inicia dentro del paréntesis más interno y se prosigue hacia afuera.



Aplicación

```
>> 3^2    % ^alt + 9,4          ...

>> -3^2                         ...

>> 16^(1/2)                     ...

>>  (-3)^3                      ...

>> -27^(1/3)                    ...
```

Aplicación

$3^2 - 5 - \frac{6}{3} \cdot 2$

```
>> 3^2-5-6/3*2          0
>> 3^2-5-(6/3)*2
```

$3^2 - 5 - \frac{6}{3 \cdot 2}$

```
>> 3^2-5-6/(3*2)        3
```

$4 \cdot 3^2 + 1$

```
>> 4*3^2+1              37
```

$(4 \cdot 3)^2 + 1$

```
>> (4*3)^2+1            145
```

Variables

- La introducción de variables ofrece nuevas posibilidades. Supongamos que queremos calcular el área de un triángulo de base 21.3 m y altura 12.6 m.

```
>> 1/2*21.3*12.6
ans = 134.1900

>> 21.3*12.6/2
ans = 134.1900
```

- Otra posibilidad es utilizar las variables base, altura, area y realizar los cálculos de la siguiente forma:

```
>> base=21.3
```

(Asigna a la variable base el valor 21.3)

```
>> altura=12.6
```

(Asigna a la variable altura el valor 12.6)

```
>> area=1/2*base*altura
```

```
134.1900
```

(Asigna a la variable area el valor correspondiente)

- Las variables base, altura, area permanecen en el espacio de trabajo con los últimos valores asignados y pueden ser llamadas o modificadas cuantas veces se desee.
- ***Observación:*** *La modificación de la variable altura no cambia el valor de la variable área si ésta no se vuelve a calcular.*

```
>> altura=9.7
>> area                          134.1900
>> area=1/2*base*altura          103.3050
```

Reglas para nombrar variables

- Las letras mayúsculas y minúsculas son distintas a efectos de nombrar variables. Por ejemplo, son diferentes las variables base, Base, BASE.
- El nombre de una variable puede tener hasta 63 caracteres; si hubiese más serían ignorados.
- El nombre de una variable debe comenzar obligatoriamente por una letra. Puede contener letras, números y el guión de subrayado (_); no se permiten espacios en blanco.

Reglas para nombrar variables

- No es conveniente nombrar variables mediante expresiones que tengan un significado específico en MATLAB: Por ejemplo, no es aconsejable utilizar log como nombre de variable ya que ésta es la designación de la función logarítmica en MATLAB.
- Como regla general es aconsejable que el nombre de una variable sea indicativo de su contenido.

```
>>namelengthmax                    63
```

- Permite preguntar al programa por el número máximo de caracteres permitidos como nombre de una variable.

```
>>iskeyword
```

- Lista completa de palabras claves (keywords) de MATLAB, que no se pueden utilizar para variables.

>>iskeyword

'break'
'case'
'catch'
'classdef'
'continue'
'else'
'elseif'
'end'
'for'
'function'

'global'
'if'
'otherwise'
'parfor'
'persistent'
'return'
'switch'
'try'
'while'

Algunas variables predefinidas en MATLAB

Hay algunas variables que, por defecto, tienen un valor asignado. Podemos citar:

- ans Es la variable que, por defecto, contiene los resultados.
- pi Contiene el valor del número real π.
- eps Es el número positivo más pequeño que sumado a 1 genera un número mayor que 1 en el ordenador.

Algunas variables predefinidas en MATLAB

- Inf o inf Representa el valor infinito. Se obtiene, por ejemplo, en caso de overflow o división por cero.

- NaN o nan (Not a Number) Representa una expresión indeterminada, por ejemplo: 0/0 .

- i o j Representa la unidad imaginaria i = j = $\sqrt{-1}$.

Signos de puntuación

- En una misma línea pueden definirse varias variables separadas por coma (,) o por punto y coma (;). La diferencia consiste en que el punto y coma inhibe la visualización en pantalla.

```
>> a=2,b=3;c=5
>> d=a*b+c;
>> d
```

Signos de puntuación

- Utilizando tres puntos se puede cambiar de línea sin ejecutar las órdenes escritas.

```
>> x=25,...
y=x/5
```

- Esto puede resultar útil cuando una expresión no cabe en una sola línea de la ventana de comandos.

Comentarios

- MATLAB ignora lo que se encuentra a la derecha del símbolo **%**. Esto permite introducir comentarios.

```
>> a=2;b=3;   %a es la base, b es la altura

>> area=a*b/2 %área del triangulo
```

Movimientos del cursor

↑ "Flecha arriba" (Recupera la línea anterior)

↓ "Flecha abajo" (Recupera la línea siguiente)

← "Flecha izquierda" (Mueve el cursor hacia la izquierda un carácter)

→ "Flecha derecha" (Mueve el cursor hacia la derecha un carácter)

- *Se recomienda hacer uso de esta utilidad siempre que sea posible ya que permite ahorrar mucho tiempo.*

Aplicación

- Calcular la suma a=1+2+3+4+5+6+7+8+9+10.
- Utilizando los movimientos del cursor, calcular b=1-2+3-4+5-6+7-8+9-10.

```
>> a=1+2+3+4+5+6+7+8+9+10

>> ↑ "pulse Flecha arriba" (editando la línea)

>> b=1-2+3-4+5-6+7-8+9-10
```

Funciones predefinidas

exp	sin	asin	sinh	asinh
log	cos	acos	cosh	acosh
log10	tan	atan	tanh	atanh
log2	cot	acot	coth	acoth
sqrt	sec	asec	sech	asech
rem	csc	acsc	csch	acsch

Notación que se usa habitualmente en matemáticas

Matemática	Matlab	Ejemplo
e^x	`exp(x)`	`exp(1)`
$\ln(x)$	`log(x)`	`log(3)`
$\log_{10}(x)$	`log10(x)`	`log10(3)`
$\log_2(x)$	`log2(x)`	`log2(3)`
$\sqrt{x}$	`sqrt(x)`	`sqrt(64)`
$\text{sen}(x)$	`sin(x)`	`sin(pi/3)`
...	...	...

Aplicación

$\log_b a = c$

$a = b^c$

```
>> sqrt(16)

>> 16^(1/2)

>> e=exp(1)

>> log10(e)
```

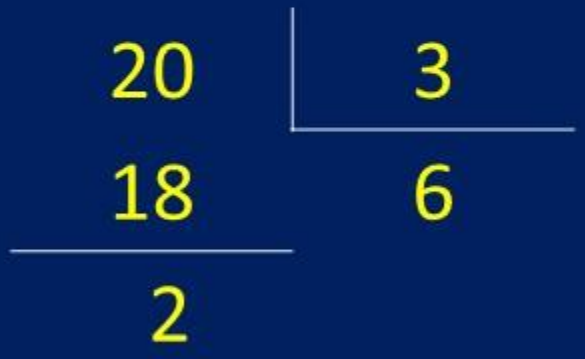
```
>> log2(e)

>> pi/4

>> atan(1)

>> log(e)
```

Funciones predefinidas en MATLAB de uso frecuente

Matlab	Definición
`abs(x)`	Valor absoluto
`fix(x)`	Redondea hacia cero
`floor(x)`	Redondea hacia menos infinito
`ceil(x)`	Redondea hacia más infinito
`round(x)`	Redondea hacia el entero más próximo
`rem(m,n)`	Resto al dividir m entre n
`rand`	Número aleatorio entre 0 y 1

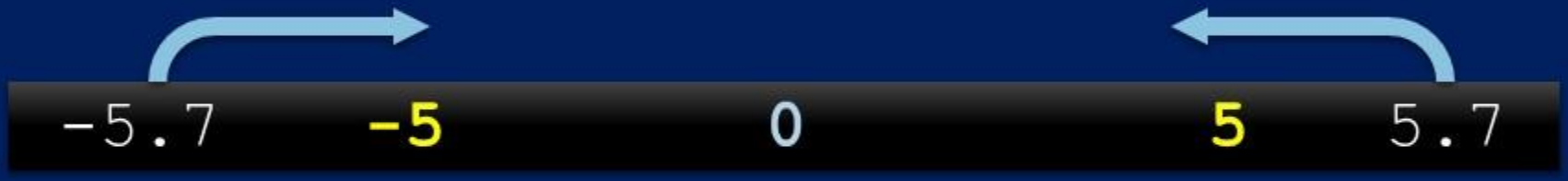
Aplicación

```
>> x=-28;
>> y=abs(x)
>> a=20/3

>> rem(20,3)

>> rand
```

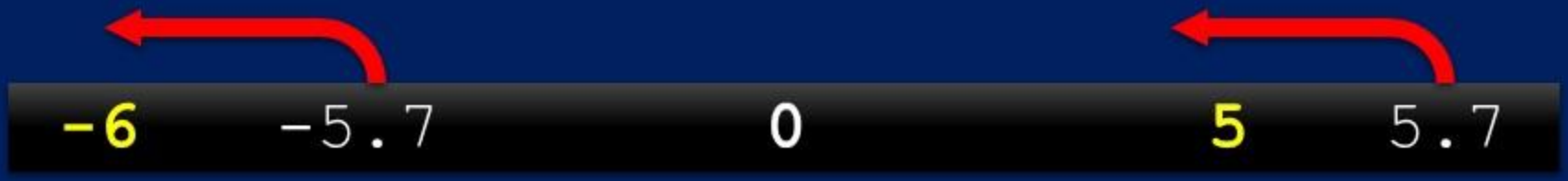


Aplicación

```
>> fix(-5.7), fix(5.7)
```




```
>> floor(-5.7), floor(5.7)
```



Aplicación

```
>> ceil(-5.7), ceil(5.7)
```


```
>> round(-5.7), round(5.7)
```

Vectores y matrices

Vectores

- *Los vectores se introducen escribiendo, entre corchetes, cada una de sus coordenadas separadas por un espacio en blanco o una coma. Por ejemplo, para introducir el vector (1, 3, 5, -7, 0.33, π, e) se escribe*

```
>> [1 3 5 -7 0.33 pi exp(1)]
```

- *o bien,*

```
>> [1, 3, 5, -7, 0.33, pi, exp(1)]
```

- *Un vector puede asignarse a una variable, por ejemplo:*

```
>> v = [1 3 5 -7 0.33 pi exp(1)]
```

x=[x1:h:xn]

- *En algunos casos se puede generar vectores sin necesidad de introducir explícitamente sus coordenadas:*

>> [10:2:20]

>> [15:1:19]

>> [1:0.5:2]

>>[20:-0.3:18]

x=[x1:xn]

- Hace lo mismo que x=[x1:1:xn].

>> [11:15]

>>[-3:3]

>>[17:20]

linspace(a,b,n)

- Genera un vector de n coordenadas espaciadas uniformemente, comenzando por a y terminando por b.

- El vector x = (-3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7, 8) se puede generar de las siguientes formas:

```
>> x=[-3 -2 -1 0 1 2 3 4 5 6 7 8]
```

linspace(a,b,n)

```
>> x=[-3:1:8]
>> x=[-3:8]
>> x=(-3:8)
>> x=-3:8
>> x=-3:8.7
>> x=linspace(-3,8,12)
```

- Apreciar la diferencia entre los dos modos descritos para generar vectores, observemos la diferencia entre los vectores x e y:

```
>> x=0:0.1:1.07

>> y=linspace(0,1.07,11)
```

Acceso a las coordenadas de un vector

- Una vez definido un vector se puede acceder a sus coordenadas para conocer su valor, utilizarlo o modificarlo.
- v(i)

Devuelve la coordenada i-ésima del vector v.

- v(i1:h:i2)

Devuelve las coordenadas de v cuyos índices van desde i1 hasta i2, con incremento h.

Acceso a las coordenadas de un vector

- v(i1:i2)

Hace lo mismo que v(i1:1:i2).

- v([i,j,k])

Devuelve las coordenadas i, j y k de v.

- v(end)

Devuelve la última coordenada de v.

Aplicación

```
>> v=[-1 3 2 4 -8 7]
```

```
>> v(2)
```

(Muestra la segunda coordenada de v)

```
>> v(end)
```

(Muestra la última coordenada de v)

```
>> v(2)+5
```

(Se utiliza la segunda coordenada de v en una operación)

Aplicación

>> v(2)=6

(Asigna el valor 6 a la segunda coordenada del vector v)

>> v(2:4)

(Muestra las coordenadas del vector v desde la segunda a la cuarta)

>> w=v(2:4)

(Define el vector w con las coordenadas segunda, tercera y cuarta de v)

>> u=v([1,5,6])

(Define u con las coordenadas primera, quinta y sexta de v)

Matrices

$$A = \begin{pmatrix} 1 & 2 & 3 \\ -1 & 4 & 8 \\ 7 & 2 & 1 \end{pmatrix}$$

- se escribe

```
>> A=[1 2 3; -1 4 8; 7 2 1];
```

- Para visualizar la matriz A en pantalla escribimos:

```
>> A
```

Definición de matrices por cajas

- Dadas dos matrices A y B con igual número de filas se puede definir una matriz C formada por todas las columnas de A y B:

```
>> A=[1 2 3; -1 4 8]
```

$$A = \begin{bmatrix} 1 & 2 & 3 \\ -1 & 4 & 8 \end{bmatrix}$$

```
>> B=[5 5; 6 6]
```

$$B = \begin{bmatrix} 5 & 5 \\ 6 & 6 \end{bmatrix}$$

```
>> C=[A B], D=[A';B]
```

Definición de matrices por cajas

- Dadas dos matrices A y B con igual número de columnas se puede definir una matriz C formada por todas las filas de A y B:

```
>> A=[1 2 3; -1 4 8]

>> B=[5 5 5; 6 6 6; 7 7 7]

>> C=[A;B]
```

Matrices especiales

- Para introducir algunas matrices de uso frecuente (matrices de ceros, matrices de unos, matrices unidad, …) MATLAB dispone de órdenes específicas:
- ones(n)

Genera una matriz cuadrada n x n formada por unos.

- ones(m,n)

Genera una matriz m x n formada por unos.

```
>> A=ones(5)
>> B=ones(3,4)
```

Matrices especiales

- zeros(n)

Genera una matriz cuadrada $n \times n$ formada por ceros.

- zeros(m,n)

Genera una matriz $m \times n$ formada por ceros.

```
>> C=zeros(5)

>> D=zeros(3,4)
```

Matrices especiales

- `eye(n)`

Genera una matriz unidad n x n.

- `eye(m,n)`

Genera una matriz m x n con unos en la diagonal principal y ceros en el resto.

```
>> E=eye(4)
```

```
>> F=eye(3,4),  G=eye(4,3)
```

Acceso a los elementos de una matriz

- Una vez definida una matriz se puede acceder a sus elementos para conocer sus valores, utilizarlos o modificarlos.
- A(i,j)

Devuelve el elemento (i, j) de la matriz A.

- A(i1:h:i2,j1:k:j2)

Devuelve la submatriz de A formada por las filas de la i1 a la i2 con incremento h y las columnas j1 a la j2 con incremento k.

Acceso a los elementos de una matriz

- A(i1:i2,j1:j2)

  Devuelve la submatriz de A formada por las filas de la i1 a la i2 y las columnas j1 a la j2.

- A(i1:h:i2,:)

  Devuelve la submatriz de A formada por las filas de la i1 a la i2 con incremento h.

- A(i1:i2,j)

  Devuelve el elemento j de las filas comprendidas entre la i1 y la i2.

Acceso a los elementos de una matriz

- A(i,j1:j2)

Devuelve el elemento i de las columnas comprendidas entre la j1 y la j2.

- A(:,j)

Devuelve el elemento j de todas las filas, o sea, la columna j.

- A(i,:)

Devuelve el elemento i de todas las columnas, o sea, la fila i.

- A(i,[j k])

Devuelve la submatriz (aij, aik)

Aplicación

```
>> A=[1 2 3 4 5; 6 7 8 9 10; 11 12 13 14 15;
16 17 18 19 20]
>> A(2,5)
>> A(1:3,2:5)      % filas 1 a 3, columnas 2 a 5
>> A(1:2:4,1:3:5)
% filas 1, 3, columnas 1, 4 A([1 3],[1 4])
>> A(:,1:2:5) % todas las filas, columnas 1, 3 y 5
```

Aplicación

```
>> A(2:4,3)        % filas 2 a 4, columna 3
>> A(2,1:4)      % fila 2, columnas 1 a 4
>> A(:,3)
>> B=[A(1:3,:) A(2:4,:)]
>> C=[A(1:3,1:3); A(:,[1 3 5])]
>> D=A([2 4], [3 5])
```