

Pre-Laboratorio 2 MA-33A 2007:

Comandos para Funciones Matemáticas

Gonzalo Hernández - Gonzalo Rios

UChile - Departamento de Ingeniería Matemática

1 Funciones matemáticas

En Matlab están predefinidas una gran variedad de funciones matemáticas. Algunas de ellas son:

1.1 Funciones elementales

Función	Descripción	Ejemplo
<code>sqrt(x)</code>	Raíz cuadrada	<code>>> sqrt(81)</code> <code>ans =</code> <code>9</code>
<code>exp(x)</code>	Exponencial	<code>>> exp(5)</code> <code>ans =</code> <code>148.4132</code>
<code>abs(x)</code>	Módulo absoluto	<code>>> abs(-5)</code> <code>ans =</code> <code>5</code>
<code>log(x)</code>	Logaritmo natural	<code>>> log(1000)</code> <code>ans =</code> <code>6.9078</code>
<code>log10(x)</code>	Logaritmo decimal	<code>>> log10(1000)</code> <code>ans =</code> <code>3</code>
<code>log2(x)</code>	Logaritmo base 2	<code>>> log2(512)</code> <code>ans =</code> <code>9</code>
<code>factorial(x)</code>	Factorial	<code>>> factorial(10)</code> <code>ans =</code> <code>3628800</code>

1.2 Funciones trigonométricas

Función	Descripción	Ejemplo
$\sin(x)$	Seno	<pre>>> sin(pi/6) ans = 0.5000</pre>
$\cos(x)$	Coseno	<pre>>> cos(pi) ans = -1</pre>
$\tan(x)$	Tangente	<pre>>> tan(pi/6) ans = 0.5774</pre>
$\cot(x)$	Cotangente	<pre>>> cot(pi/6) ans = 1.7321</pre>
$\text{asin}(x)$	Arco-seno	<pre>>> asin(0.8660) ans = 1.0471</pre>
$\text{acos}(x)$	Arco-coseno	<pre>>> acos(0.866) ans = 0.5236</pre>
$\text{atan}(x)$	Arco-tangente	<pre>>> atan(1) ans = 0.7854</pre>
$\text{acot}(x)$	Arco-cotangente	<pre>>> acot(1) ans = 0.7854</pre>
$\sinh(x)$	Seno hiperbólico	<pre>>> sinh(0) ans = 0</pre>
$\cosh(x)$	Coseno hiperbólico	<pre>>> cosh(0) ans = 1</pre>
$\tanh(x)$	Tangente hiperbólica	<pre>>> tanh(1) ans = 0.7616</pre>

1.3 Funciones de redondeo

Función	Descripción	Ejemplo
round(x)	Redondea al entero más cercano	>> round(3.14) ans = 3
fix(x)	Redondea al entero más cercano a cero	>> fix(1.98) ans = 1
ceil(x)	Redondea al entero mayor más cercano	>> ceil(10.1) ans = 11
floor(x)	Redondea al entero menor más cercano	>> floor(-3.28) ans = -4
rem(x,y)	Resto de dividir en forma entera x por y	>> rem(11,3) ans = 2
sign(x)	Devuelve el signo de x	>> sign(-1034) ans = -1
max(x,y)	Devuelve el maximo entre x e y	>> max(-6,9) ans = 9
min(x,y)	Devuelve el minimo entre x e y	>> min(-6,9) ans = -6

2 Funciones y vectores especiales

En muchas ocasiones es necesario evaluar una misma función en muchos puntos. Esto se puede hacer en Matlab definiendo el argumento x como un vector fila o columna. Veamos un ejemplo:

```
>> x=[1 2 3 4 5 6 7 8 9]
```

```
x =
```

```
1      2      3      4      5      6      7      8      9
```

```
>> sqrt(x)
```

```
ans =
```

```
1.0000    1.4142    1.7321    2.0000    2.2361    2.4495    2.6458    2.8284    3.0000
```

También se pueden evaluar las funciones típicas en matrices.

2.1 Paso dado

En muchas ocasiones necesitaremos que un vector comience en un determinado número, y en cada posición del vector sea igual a la posición anterior más una constante llamada paso. En Matlab se pueden crear estos vectores de la siguiente forma:

$$\text{nombre_variable} = [m : q : n]$$

m : El primer término del vector.

q : El paso entre dos términos.

n : El último término del vector.

Observaciones

- i) Los corchetes son opcionales.
- ii) El paso **q** puede ser un real positivo o negativo.
- iii) Si el paso **q** es omitido, por defecto vale 1, es decir $[m : n] \iff [m : 1 : n]$
- iv) Si $m = n$, entonces el paso no es considerado y guarda en la variable la constante **m**.
- v) Si los valores de **m**, **q** y **n** no calzan, entonces:
 - (a) Si **q** es positivo, el último término del vector es el mayor número menor que **n**.
 - (b) Si **q** es negativo, el último término del vector es el menor número mayor que **n**.

Ejemplos

```
>> x=[0:-3:-10]
```

```
x =  
0   -3   -6   -9
```

```
>> y=[0:3:10]
```

```
y =  
0  3  6  9
```

```
>> z=[0:0.25:1]
```

```
z =  
0  0.2500  0.5000  0.7500  1.0000
```

2.2 Numero de puntos dados

En otros casos necesitaremos un vector de tamaño fijo, donde el primer y último término sean los extremos de un intervalo, y los otros términos sean puntos intermedios equiespaciados. Para esto, en **Matlab** existe una funcion predefinida:

$$\text{nombre_variable} = \text{linspace}(xi, xf, n)$$

xi : Primer término del vector
xf : Último término del vector
n : Dimensión del vector

Observaciones

- i) **n** debe ser un número natural.
- ii) Si **n** es omitido, por defecto se concidera **n**=100.
- iii) Si **xi** es menor que **xf**, entonces el vector es creciente.
- iv) Si **xi** es mayor que **xf**, entonces el vector es decreciente.

Ejemplos

```
>> x=linspace(0,8,6)
```

x =

```
0    1.6000    3.2000    4.8000    6.4000    8.0000
```

```
>> x=linspace(30,10,11)
```

x =

```
30    28    26    24    22    20    18    16    14    12    10
```

```
>> x=linspace(-1,-10,7)
```

x =

```
-1.0000   -2.5000   -4.0000   -5.5000   -7.0000   -8.5000  -10.0000
```

3 Graficar funciones

El comando plot crea un gráfico bidimensional a partir de dos vectores. La forma de usar este comando es:

$$\text{plot}(x, y)$$

x: Vector que contiene los puntos de la abcisa.

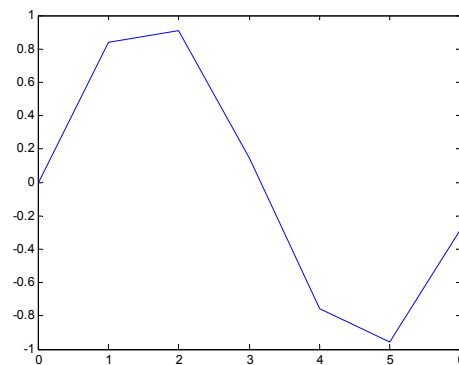
y: Vector que contiene los puntos de la ordenada.

Observaciones

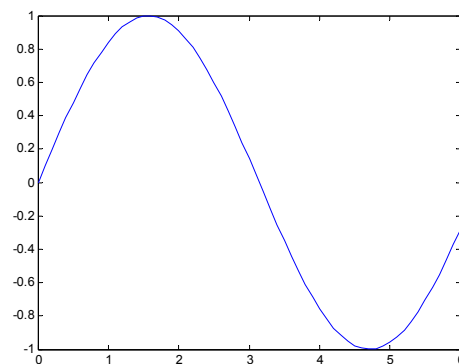
- i) Los vectores **x** e **y** deben tener la misma cantidad de elementos.
- ii) La función plot une con una recta el punto (x_{j-1}, y_{j-1}) con el punto (x_j, y_j) , $\forall j = 2 \dots n$, y luego grafica estas n-1 rectas.
- iii) La calidad del gráfico depende proporcionalmente a la cantidad de puntos tomados.

Veamos dos ejemplos:

```
>> x=[0:1:6];  
>> y=sin(x);  
>> plot(x,y)
```



```
>> x=[0:0.1:6];  
>> y=sin(x);  
>> plot(x,y)
```



4 Archivos Script

A veces es necesario reutilizar variables o líneas de código. Estas se pueden guardar en **Matlab** en archivos **.m** (al igual que las funciones). La forma de programar estos archivos es análogo a crear funciones en **Matlab**, con la diferencia que no son funciones. Veamos un ejemplo:

4.1 Para crear un archivo **.m**

File→New→M-File

4.2 Programar

```
x=[1:10];  
y=sqrt(x)
```

4.3 Guardar la función

File→Save as

// Se escoge la carpeta donde guardaremos la función y el nombre que tendrá. En este ejemplo se llamará **raices.m**

4.4 Ejecutar el archivo Script

Primero hay que asegurarse que el archivo **raices.m** aparezca en la ventana Current Directory.

```
>> raices
```

```
y =
```

```
Columns 1 through 9
```

```
1.0000    1.4142    1.7321    2.0000    2.2361    2.4495    2.6458    2.8284    3.0000
```

```
Column 10
```

```
3.1623
```