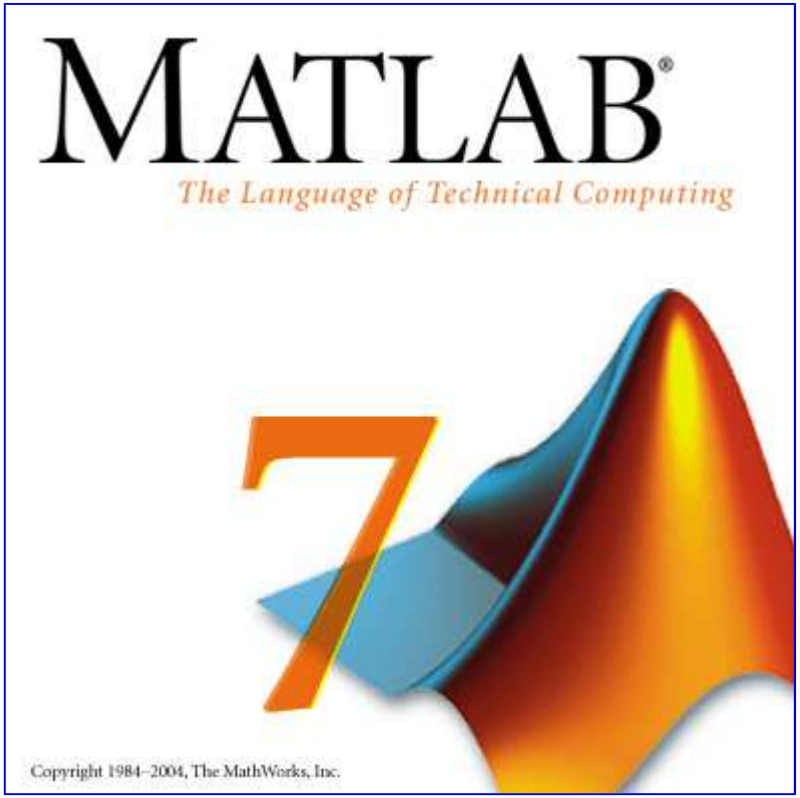
Escuela de Ingeniería de Telecomunicación y Electrónica
Universidad de Las Palmas de Gran Canaria

PDS

Grado en Ingeniería en Tecnologías de la Telecomunicación

Introducción a Matlab



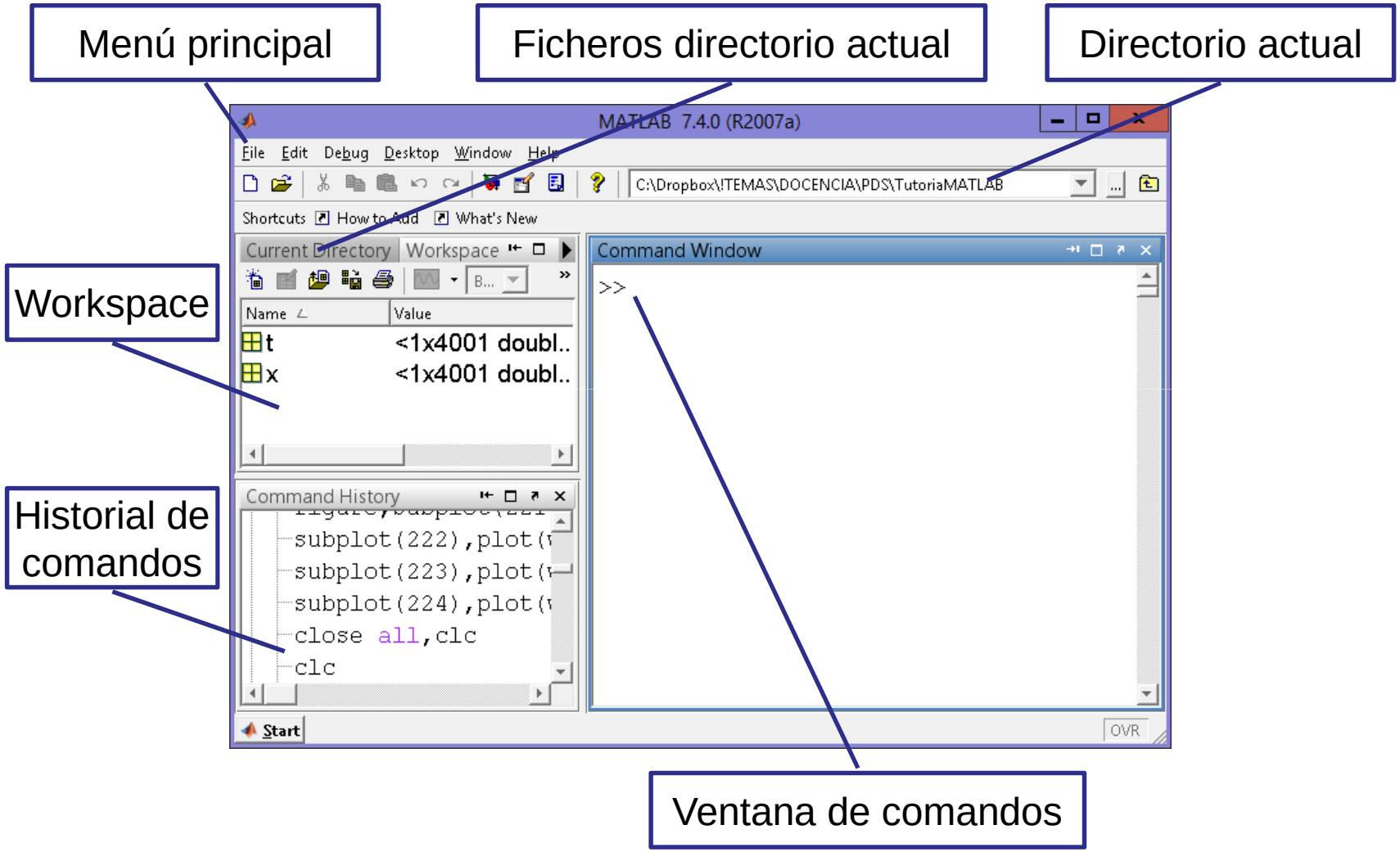
Autor: **José Aurelio Santana Almeida**

¿Qué es Matlab?

Entorno de programación con un lenguaje propio que facilita la ejecución operaciones de cálculo y representación gráfica.

- Permite la implementación de señales y simulación de sistemas con gran facilidad.
- Se centra en el dominio de las matemáticas discretas (incluso números complejos)
- Maneja principalmente datos vectoriales y matriciales.
- Genera diferentes tipos de representación gráfica en 2D y 3D.
- Permite almacenar la secuencia de instrucciones que implementa una operación de cálculo/gráfica como un script o función.
- Incluye una librería de funciones que desarrollan operaciones muy diversas.

Entorno de Trabajo



Variables. Asignación y Referencia

Creación de una variable mediante asignación

```
>>a=-2.5; %Se puede insertar comentarios a la derecha de %
          %La coma decimal se expresa con .
          %El ; final evita eco del resultado
          %pero dicho valor puede ser consultado en el WORKSPACE
```

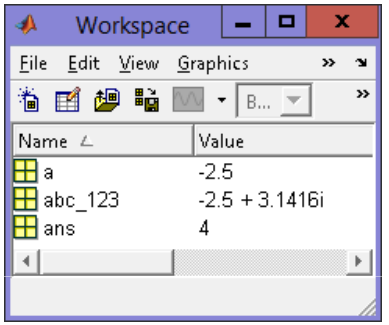
Si la variable aparece a la derecha del = es una referencia de lectura (a la izquierda se considera referencia de escritura)

```
>>abc_123=a+pi*j  %El nombre de variables debe comenzar con texto
                  %Se distingue mayúsculas/minúsculas
                  %También puede incluir números y/o el carácter _
                  %Variables predefinidas:  j,i,pi,Inf,NaN
                  %La ausencia de ; genera eco del resultado
abc_123 =

  -2.5000 + 3.1416i
```

Variables. Asignación y Referencia

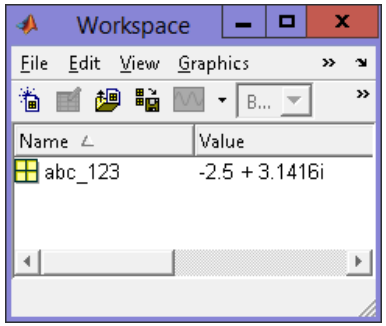
Una operación matemática es considerada asignación sin destino

```
>>2+2  %Por defecto, su resultado será asignado a la variable ans

ans =

     4
```



Para borrar selectivamente variables del WORKSPACE usamos *clear* seguido de la lista de variables separadas por espacios

```
>>clear  a  ans , %Si no indicamos variables, clear las borrará todas
```



Variables. Tipos básicos

1) ESCALARES (estructuras de 1x1): Todos los ejemplos previos

2) VECTORES (estructuras de 1xN ó Mx1): Implementa señales

En Vectores y Matrices podemos usare dos tipos de referencia:

a) Referencia DIRECTA.- Contenido de una posición de la matriz

```
>>v(1)=0;  v(3)=-5.6 %Las posiciones son enteros positivos (1 es la 1ª)
                     %Por defecto se crea un vector fila, pero si el vector
                     %existía previamente, se cambiará su contenido
                     %(manteniendo su formato original)
                     %Las posiciones no asignadas se rellenan con 0
```

b) Referencia EXPLÍCITA.- Contenido de todas las posiciones

```
>>vc=[0, -3.5, 7], vf=[1.2 ; -1] %Las columnas se separan con , o espacio
                                 %Las filas se separan con ; o ENTER
                                 %La , tras asignación separa instrucciones
```

Variables. Tipos básicos

El tipo de vector más utilizado es:

RANGO.- Progresión aritmética, expresada como *v=vi:p:vf*, que comienza en *vi*, obtiene cada elemento sucesivo sumando *p* al elemento anterior, y acaba en (o antes de) *vf*. Si *p*=1 se puede omitir, quedando *v=vi:vf*.

```
>> r=-3.1:-1:-7, % si el paso es negativo el rango es descendente
```

3) MATRICES(estructuras de MxN)

```
>> A=[0,1,2;-3,7,2] %R.expl.- Filas/Columnas deben encajar como puzzle
A =
     0     1     2
    -3     7     2

>> A(1,2)=44; A(2,2)=22 %R.dir.- Si estaba creada, mantiene su formato
A =
     0    44     2
    -3    22     2
```

Variables. Referencia Avanzada

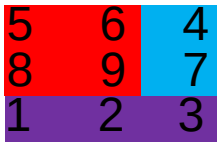
Se puede alcanzar un acceso más eficiente a vectores y matrices, integrando en las referencias otros vectores

```
>> k1=1:2:4; V(k1)=1; k2=2:2:4; V(k2)=2 %usa vectores como posiciones
```

```
>> Vp=[V,0,V,0] %usa vectores como contenido de vectores
```

```
>> M=[1,2,3;4,5,6;7,8,9];
```


```
>> R(1:2,1:2)=M(2:3,2:3); R(1:2,3)=M(2:3,1); R(3,:)=M(1,:) %piezas
R =
```




```
>>R=[M(2:3,2:3),M(2:3,1);M(1,:)]; % Más sencillo y daría igual resultado
```

Operaciones con Variables

Cada tipo de variable tiene restricciones específicas

Las combinaciones presentan especificidades distintas

En los ejemplos usaremos los siguientes datos:

```
>>a=-3; b=5;                          %Escalares
>>v1=[2,6,4]; v2=[1,-1,1];            %Vectores
>>m1=[1,2;3,4], m2=[1,1;1,1]          %Matrices

m1 =
     1     2
     3     4

m2 =
     1     1
     1     1
```

Operaciones: Suma y Resta

Entre Escalares

```
>> escalares=a+b  %entre escalares = escalar
escalares =
     2
```

Entre Vectores o Matrices (con dimensiones iguales)

```
>> vectores=v1+v2 %entre vectores = vector (¡dimensiones idénticas!)
vectores =
    3    5    5
>> matrices=m1+m2  %entre matrices = matriz (¡dimensiones idénticas!)
matrices =
    2    3
    4    5
```

Entre Escalares y Matrices (cada elemento se opera con el escalar)

```
>> esc_y_mat=b+m1 %entre matriz y escalar = matriz
esc_y_mat =
    6    7
    8    9
```

Operaciones: Producto elemento a elto.

Entre Escalares

```
>> escalares=a.*b  %entre escalares = escalar
escalares =
   -15
```

Entre Vectores o Matrices (con dimensiones iguales)

```
>> vectores=v1.*v2 %entre vectores = vector (¡dimensiones idénticas!)
vectores =
     2    -6     4
>> matrices=m1.*m2 %entre matrices = matriz (¡dimensiones idénticas!)
matrices =
     1     2
     3     4
```

Entre Escalares y Matrices (cada elemento se opera con el escalar)

```
>> esc_y_mat=b.*m1 %entre matriz y escalar = matriz
esc_y_mat =
     5    10
    15    20
```

Operaciones: Producto Matricial

Entre Escalares

```
>> escalares=a*b  %entre escalares = escalar
escalares =
   -15
```

Entre Vectores o Matrices (suma de productos fila x columna)

```
>> vectores=v1*[1;-1;1] %Debe ser N1=M2. Dimens. resultado: M1 x N2
vectores =
     0
>> matrices=m1*m2 %Debe ser N1=M2. Dimens. resultado: M1 x N2
matrices =
     3     3
     7     7
```

Entre Escalares y Matrices (cada elemento se opera con el escalar)

```
>> esc_y_mat=b*m1 %entre matriz y escalar = matriz
esc_y_mat =
     5    10
    15    20
```

Operaciones: División elemento a elto.

Entre Escalares

```
>> escalares=a./b  %entre escalares = escalar=a/b=div.mat.=a*inv(b)
escalares =
   -0.6000
```

Entre Vectores o Matrices (con dimensiones iguales)

```
>> vectores=v1./v2 %entre vectores = vector (¡dimensiones idénticas!)
vectores =
     2    -6     4
>> matrices=m1./m2 %entre matrices = matriz (¡dimensiones idénticas!)
matrices =
     1     2
     3     4
```

Entre Escalares y Matrices (cada elemento se opera con el escalar)

```
>> esc_y_mat=b./m1 %entre matriz y escalar = matriz
esc_y_mat =
    5.0000    2.5000
    1.6667    1.2500
```

Operaciones: Potenciación elto. a elto.

Entre Escalares

```
>>escalares=a.^b  %entre escalares = escalar
escalares =
  -243
>>escalares=a^b;  %Potenc. matricial = a^b = a.^b (¡entre escalares!)
```

Entre Matrices (o Vectores)

```
>>vectores=v1.^v2 %entre vectores = vector (¡dimensiones idénticas!)
vectores =
    2.0000    0.1667    4.0000
```

Entre Matrices (o Vectores) y Escalares

```
>>esc_y_mat=b.^m1 %entre matriz y escalar = matriz
esc_y_mat =
     5    25
   125   625
```

Operaciones: Trasposición

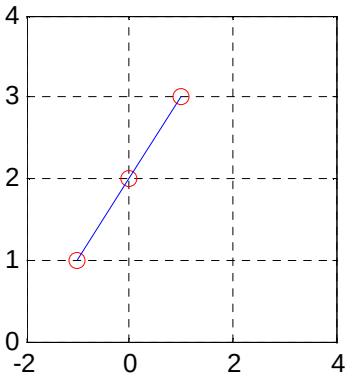
Dada una matriz (o vector) complejo *A*, la operación *B=A'* traspone y conjuga *A*; mientras que *C=A.'* traspone (y no conjuga) *A*.

```
>>vc=j*v2'
vc =
        0 + 1.0000i
        0 - 1.0000i
        0 + 1.0000i

>>vf1=vc'   %cambia filas por columnas ¡y conjuga!
vf1 =
        0 - 1.0000i        0 + 1.0000i        0 - 1.0000i

>>vf2=vc.'  %cambia filas por columnas ¡NO conjuga!
vf2 =
        0 + 1.0000i        0 - 1.0000i        0 + 1.0000i
```

Representación. Señal continua

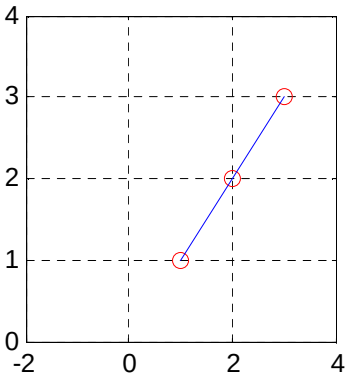
Sintaxis general: plot(x,y1,c1,x,y2,c2,x,y3,c3, …)

>>x=[-1,0,1]; y=[1,2,3]; plot(x,y)



Abscisa= x	Ordenada= y
-1	1
0	2
1	3

>>plot(y)



Abscisa= 1:end	Ordenada= y
1	1
2	2
3	3

semilogx(…) %abscisas en escala logarítmica

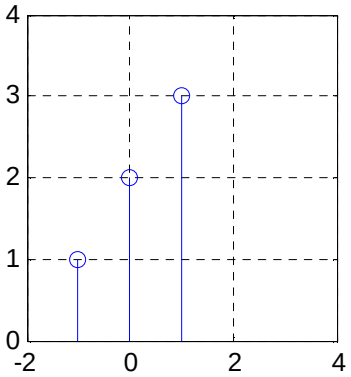
semilogy(…) %ordenadas en escala logarítmica

loglog(…) %ambos ejes en escala logarítmica

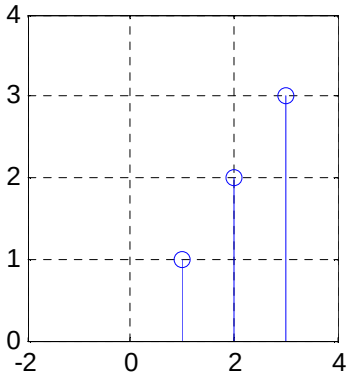
Representación. Señal discreta

stem(x,y)

>>x=[-1,0,1]; y=[1,2,3]; stem(x,y)

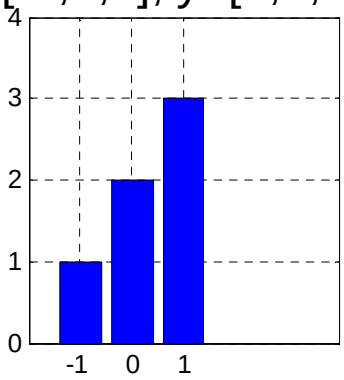


>>stem(y)

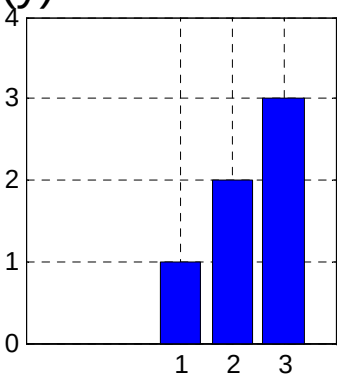


bar(x,y)

>>x=[-1,0,1]; y=[1,2,3]; bar(x,y)



>>bar(y)

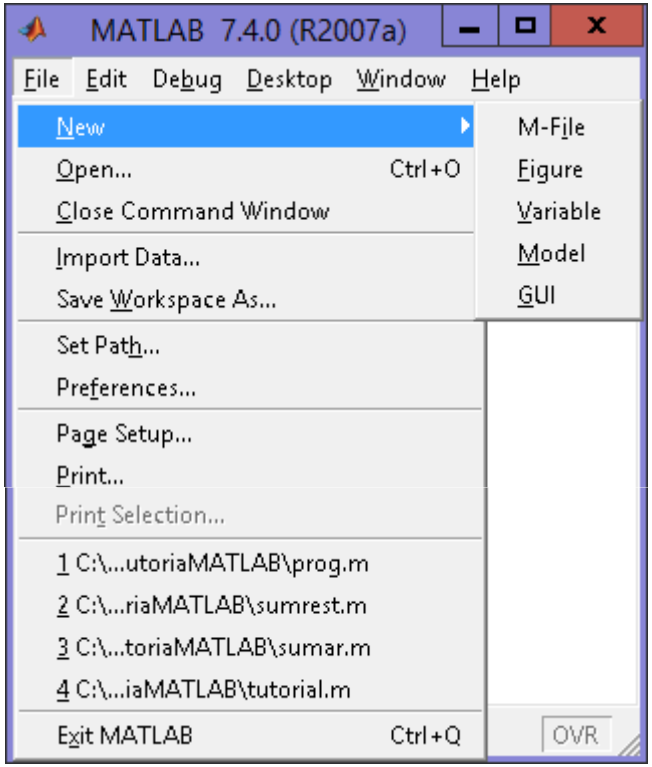


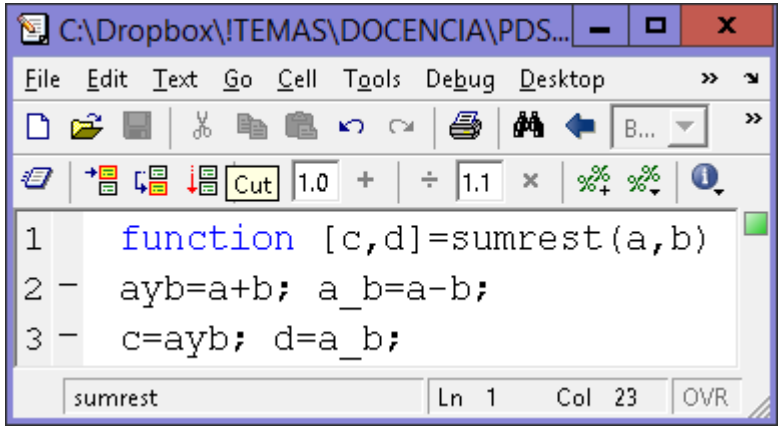
Almacenamiento de Variables

- Permite almacenar y recuperar variables de disco
- Se evita volver a ejecutar el programa que las generó
- Pueden utilizarse como entrada en otros programas

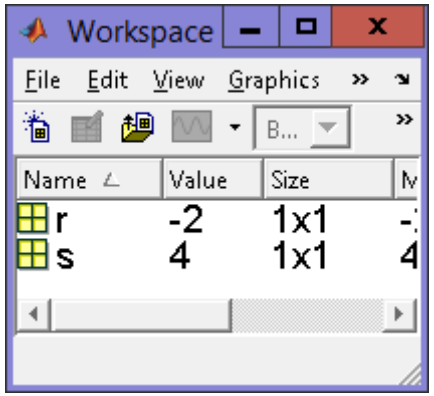
```
>>save datos.mat x y, %Guarda x e y en el fichero datos.mat
                      %si no especificamos variables se guardan todas

>>load datos.mat,     %Cargaría todas las variables de datos.mat en el
                      %workspace
```

Ficheros de Instrucciones. Funciones

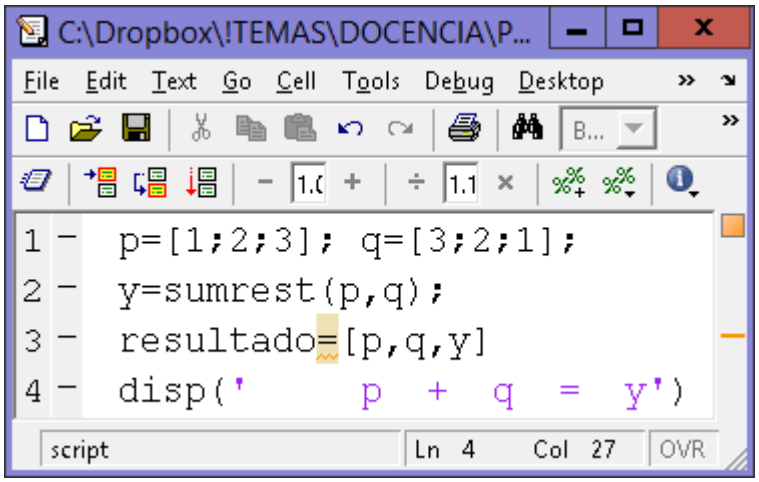




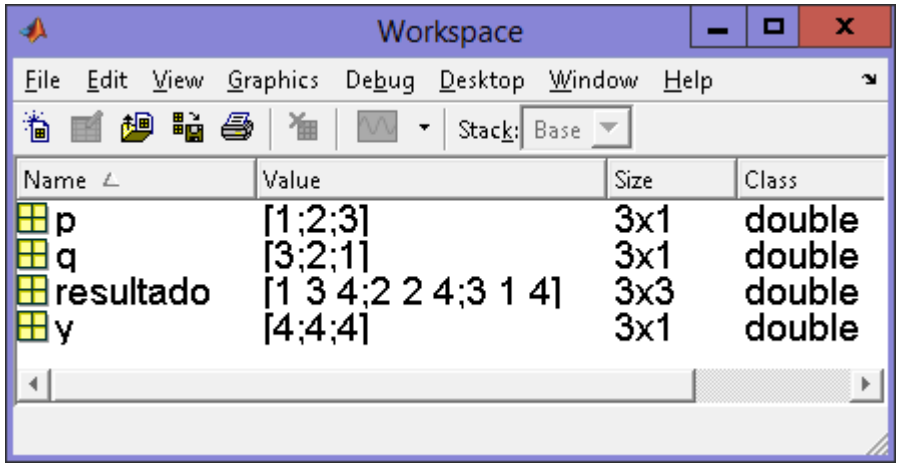

```
>> clear
>> [s,r]=sumar(1,3) %Variables internas
                    %de la función son
                    %"olvidadas"

s =
     4
r =
    -2
```



Ficheros de Instrucciones. Scripts




```
>> clear
>> prog %Workspace maneja todas las
        %variables internas del script

resultado =

     1     3     4
     2     2     4
     3     1     4

  p + q = y
```



Instrucciones de Control de Flujo: *for*

En cada iteración del bucle, el contador toma el siguiente valor del vector al que está asignado

```
>> for contador=0:0.5:1, %el bucle se repite para cada valor del contador
        valor=contador
    end

valor =
     0

valor =
    0.5000

valor =
     1
```

Instrucciones de Control de Flujo: *if*

```
>> valor=randn(1,1) %vector de 1x1 aleatorio (dist. normal media=0, var=1)

>> if valor<0  %condición puede ser una combin. de oper. relacionales
      disp('El valor es negativo')
   else          %si la condición no se cumple
      disp('El valor es positivo o nulo')
   end

valor =

    0.1253

El valor es positivo o nulo
```

Operadores Relacionales

Operadores Comparadores

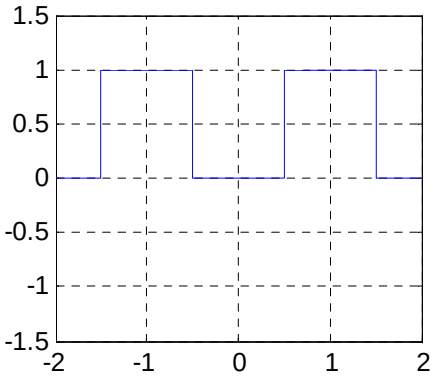
a == b	"a" igual a "b“
a ~= b	"a" distinto a "b“
a < b	"a" menor que "b“
a > b	"a" mayor que "b“
a <= b	"a" menor o igual que "b“
a >= b	"a" mayor o igual que "b"

Operadores Lógicos

P & Q	Son verdaderas las condiciones P y Q,	p.e. (a<b)&(c>d)
P | Q	Es verdadera la condición P o la Q,	p.e. (a<b)|(c>d)
~P	Es verdadera la condición opuesta a P,	p.e. ~(a<b)

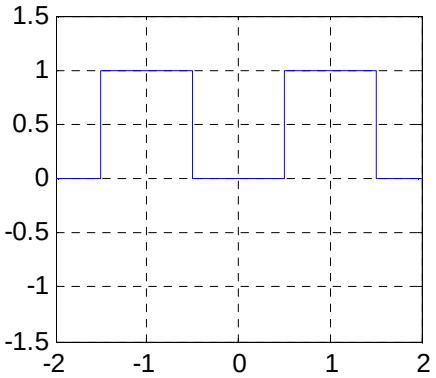
Operadores Relacionales: Ejemplo

```
>> t=-2:0.001:2; N=length(t); x=zeros(1,N);
>> for n=1:N
        if  ( t(n)>=–0.5 & t(n)<0.5 ) | …
           (  t(n)>=0.5 & t(n)<1.5  )

            x(n)=1;
        end
    end
>> plot(t,x), axis([-2,2,-1.5,1.5]), grid
```



O, más eficiente, con vectores lógicos:

```
>> t=-2:0.001:2;
>> x=(t>=–1.5 & t<–0.5)+(t>=0.5 & t<1.5);
>> plot(t,x), axis([-2,2,-1.5,1.5]), grid
```



RESUMEN: Operadores

Operadores

Caracteres Especiales:

= *Asignación*
[;] *Concatenación vertical*
[,] *C. horizontal o Parámetros de Salida*
(,) *Indexación o Parámetros de Entrada*
% *Comentario*
: *Rango a:b:c ⇒ { a, a+b, ... , c }*
; *Supresión de salida*
, *Separación de comandos*
() *Precedencia de operador*
' *Delimitador de texto ⇒* 'texto'

Operadores Aritméticos:

.' *Trasponer*
+ *Suma*
- *Resta*
.* *Producto elemento a elemento*
***** *Producto matricial*
./ *División elemento a elemento*
.^ *Potencia elemento a elemento*

Op. Relacionales:

== *Igual*
~= *Distinto*
> *Mayor*
< *Menor*
>= *Mayor o Igual*
<= *Menor o Igual*

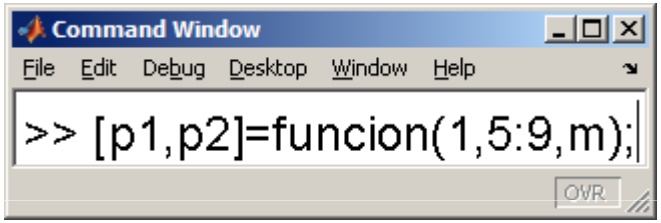
Op. Lógicos:

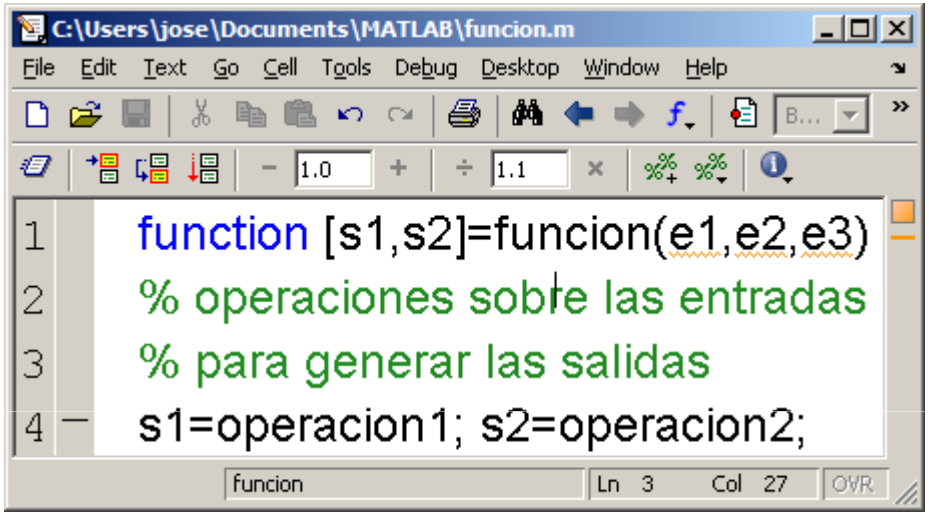
~ *Not*
| *Or*
& *And*

RESUMEN: Funciones

Funciones

Funciones de usuario:





Funciones internas:

Información básica			title	axis	grid	sin	cos	tan
length	size	max	**Operaciones de texto**			sinc	conj	atan
min	find	nargin	disp	num2str	format	real	imag	abs
Operaciones gráficas			**Inicialización de matrices**			exp	log	sqrt
plot	stem	bar	ones	zeros	rand	mean	std	round
figure	subplot	semilogy	randn	load	save	**Ayuda**		
label	xlabel	ylabel	**Operaciones matemáticas**			help	help ops	doc