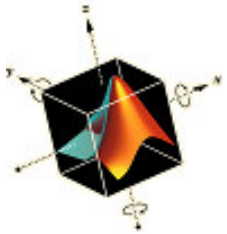
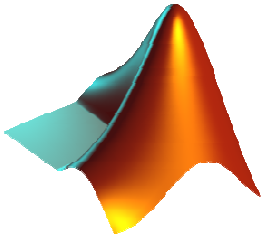
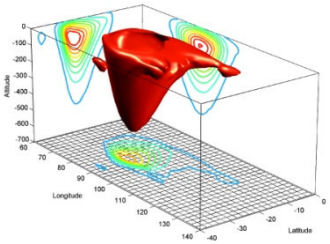






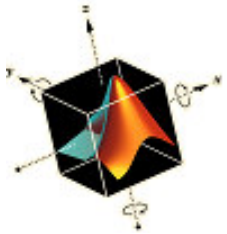
Gráficos en MATLAB

Pedro Corcuera

Dpto. Matemática Aplicada y
Ciencias de la Computación

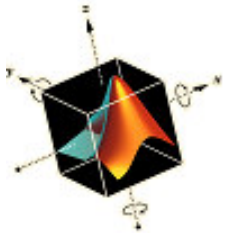
Universidad de Cantabria

corcuerp@unican.es



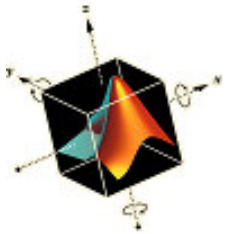
Objetivos

- Presentar la implementación de una amplia selección de capacidades de gráficas de dos dimensiones



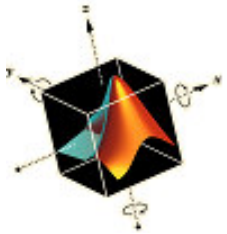
Indice

- Introducción a la gestión de gráficos
- Comandos Básicos 2D Plotting
- Notas y mejoras en gráficos



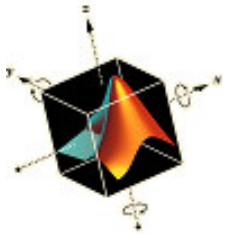
Introducción

- Matlab proporciona una amplia selección de capacidades muy flexibles y fáciles de implementar de gráficos en dos y tres dimensiones
- Las funciones gráficas se puede agrupar en tres categorías:
  - Gestión gráfica
  - Generación de curvas y superficie
  - Anotación y características del gráfico
- La mayor parte de las funciones gráficas tienen una sintaxis similar



Resumen de funciones gráficas principales

Management	Generation		Annotation and Characteristics
figure hold subplot zoom **3D** rotate3d view	**2D** axes bar convhull delauney fill image loglog movie patch pie plot plotyy polar semilogx semilogy stairs stem voronoi	**3D** contour, contour3, contourf cylinder mesh, meshc, meshz pie3 plot3 surf, surfc waterfall	**2D and 3D** axis, axis equal, axis off, axis image box clabel grid legend set text title xlabel xlim ylabel ylim **3D** colorbar colormap shading text3 zlabel

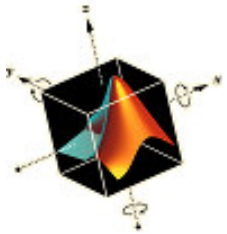


Generalidades

- Un gráfico es creado en una ventana de figura, que es una ventana creada por Matlab en tiempo de ejecución, cuando cualquier función de gestión, generación o anotación y características se invoca
- Para retener cada nuevo gráfico en la ventana de figura se debe usar

  `figure`(n)

  donde n es un entero



Generalidades

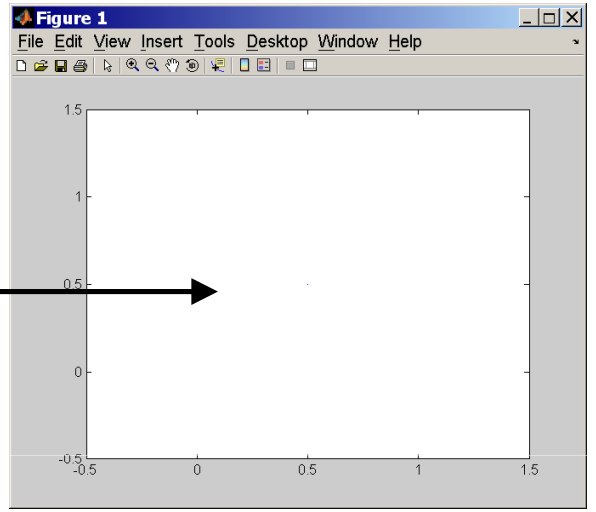
- Se puede colocar varios gráficos creados independientemente en una ventana de figura con `subplot`(i, j, k)

  donde
  - i,j dividen la ventana en sectores (filas y columnas)
  - k indica el sector donde se coloca el gráfico
- Dentro de cada sector, se puede usar cualquier conjunto compatible de funciones de generación de gráficos 2d o 3d
- Se puede guardar el gráfico en muchos formatos

Ejemplos

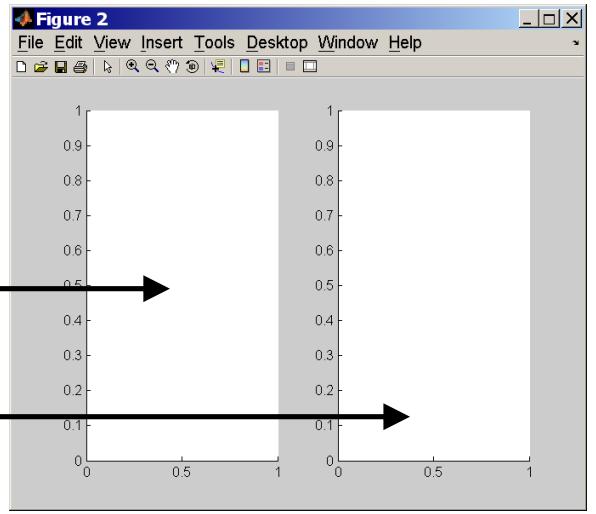
Script o Función

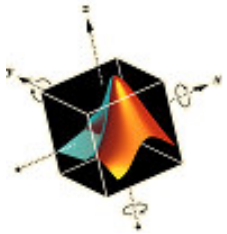
```
figure(1)
expresiones plotting
```



Script or Función

```
figure(2)
subplot(1, 2, 1)
expresiones plotting
subplot(1, 2, 2)
expresiones plotting
```





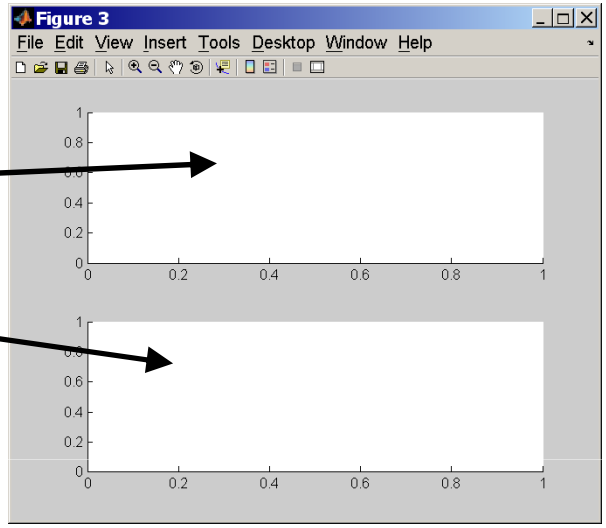
Ejemplos

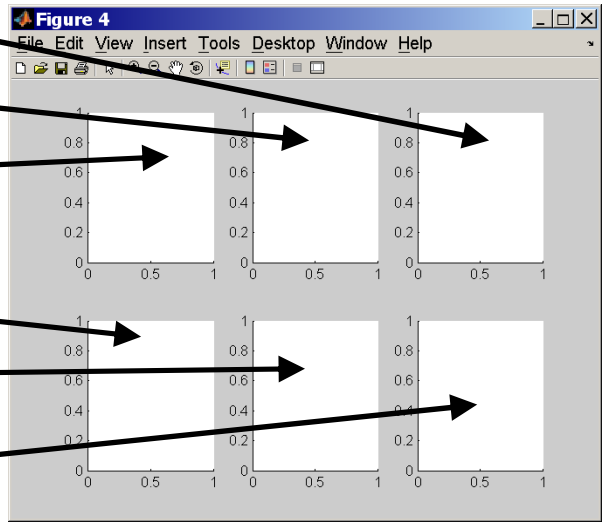
Script or Función

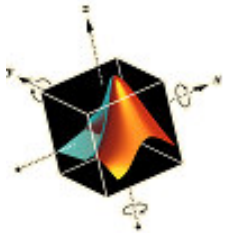
```
figure(3)
subplot(2, 1, 1)
plotting expressions
subplot(2, 1, 2)
plotting expressions
```

Script or Función

```
figure(4)
subplot(2, 3, 3)
plotting expressions
subplot(2, 3, 2)
plotting expressions
subplot(2, 3, 1)
plotting expressions
subplot(2, 3, 4)
plotting expressions
subplot(2, 3, 5)
plotting expressions
subplot(2, 3, 6)
plotting expressions
```





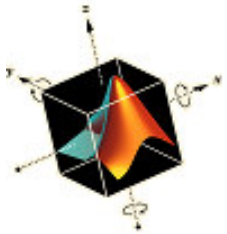


Generalidades

- Como cada función de generación de gráfico crea una nueva ventana de figura, para dibujar más de una curva, superficie o línea (o combinación de éstos) en un mismo gráfico, se debe usar

```
hold on
```

- Todas las figuras creadas se pueden copiar al portapapeles seleccionando *Copy Figure* en el menú *Edit* dentro de cada ventana de figura.
  - La figura se puede pegar en cualquier documento que acepte el formato Windows metafile



Copy Figure

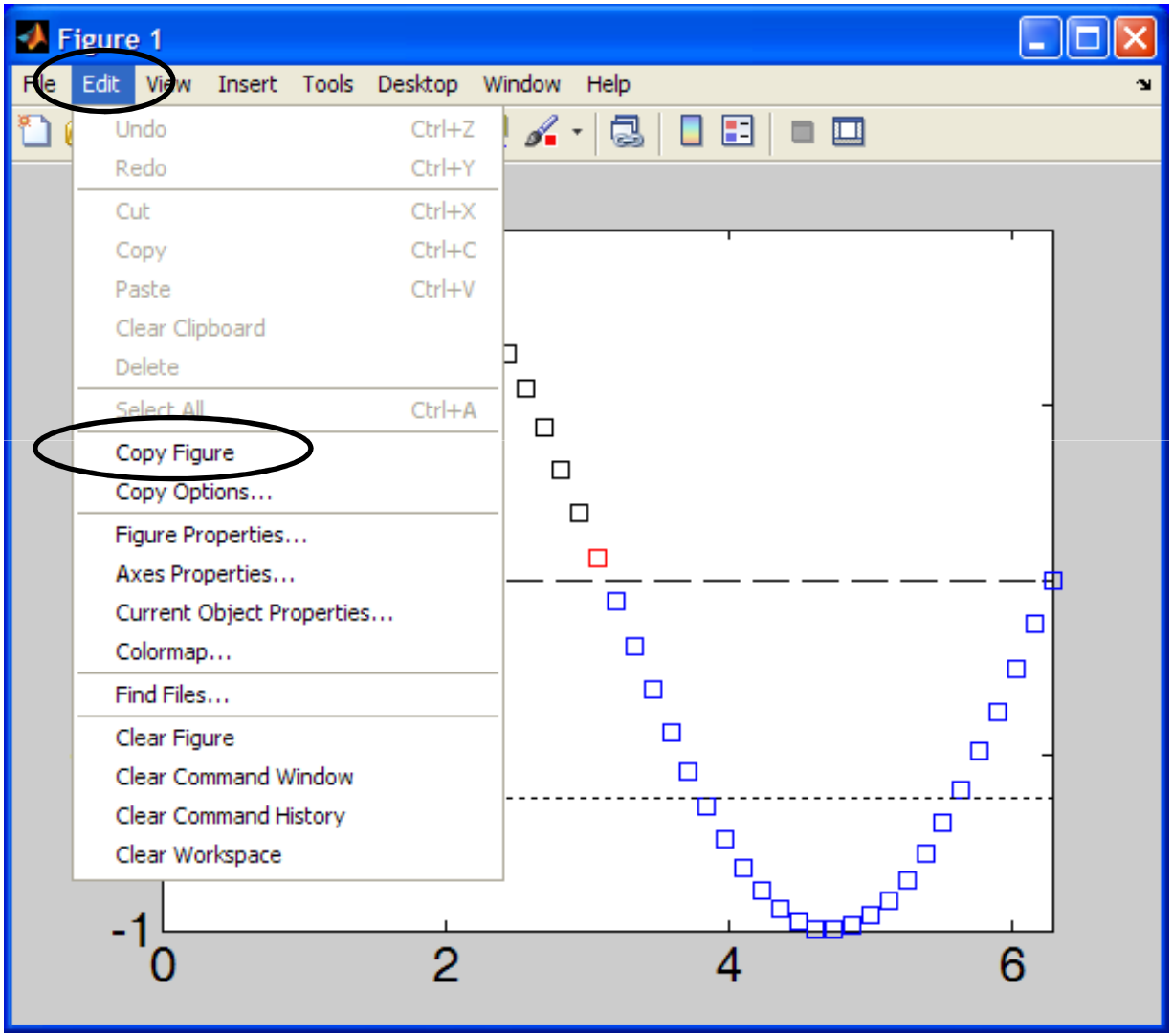
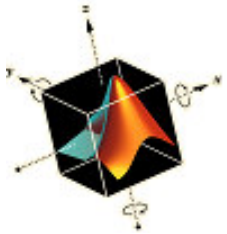


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
Undo Ctrl+Z
Redo Ctrl+Y
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Clear Clipboard
Delete
Select All Ctrl+A
Copy Figure
Copy Options...
Figure Properties...
Axes Properties...
Current Object Properties...
Colormap...
Find Files...
Clear Figure
Clear Command Window
Clear Command History
Clear Workspace
-1
0
2
4
6



Comandos básicos de gráficos 2d

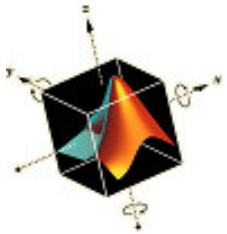
- El comando básico para gráficos 2d es

  `plot`(u, v, c)

  donde

  u y *v* son las coordenadas *x* e *y*, respectivamente, de un punto o series de puntos. Los puntos pueden ser un par de números, vectores, matrices o expresiones que los producen

  c es una cadena opcional de caracteres para especificar el color de la línea/punto, tipo de punto o características de la línea



Comandos básicos de gráficos 2d

- Cuando se grafican puntos y líneas, pero los puntos de la línea (*u*1, *v*1) son diferentes de los puntos (*u*2, *v*2) se usa

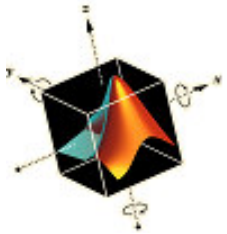
```
plot(u1, v1, c1, u2, v2, c2)
```

o

```
plot(u1, v1, c1)
hold on
plot(u2, v2, c2)
```

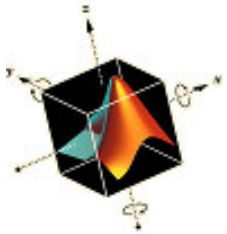
donde

c1 y *c2* contienen los símbolos para los tipos y colores



Características de línea y punto

Line type		Line or point color		Point type	
Symbol	Description	Symbol	Description	Symbol	Description
-	Solid	r	Red	+	Plus sign
--	Dashed	g	Green	o	Circle
:	Dotted	b	Blue	*	Asterisk
-.	Dashed-dot	c	Cyan	.	Point
		m	Magenta	x	Cross
		y	Yellow	s	Square
		k	Black	d	Diamond
		w	White	^	Upward-pointing triangle
				v	Downward-pointing
				>	triangle
				<	Right-pointing triangle
				p	Left-pointing triangle
				h	Pentagram
					Hexagram



Cambio de atributos de línea y punto

- Se pueden cambiar los atributos de la lineas y puntos que se grafican de dos formas
- Primera opción: usar

  `plot`(u1, v1, c1,'KeyWord', KeyWordValue, …)

  donde

  'KeyWord' es una expresión string expression del keyword para uno de los atributos de la línea y punto

  KeyWordValue es un valor numérico o un string dependiente de 'KeyWord'.

  Se puede usar tantos pares de keywords y valores como sea necesario

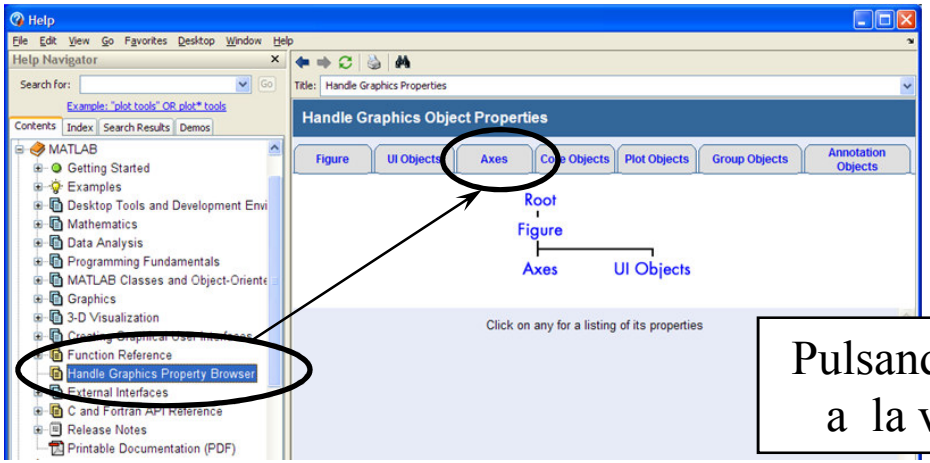
Cambio de atributos de línea y punto

- Segunda opción: obtener un handle de la entidad y cambiar su atributo usando `set`

  hdl = `plot`(u1, v1, c1);

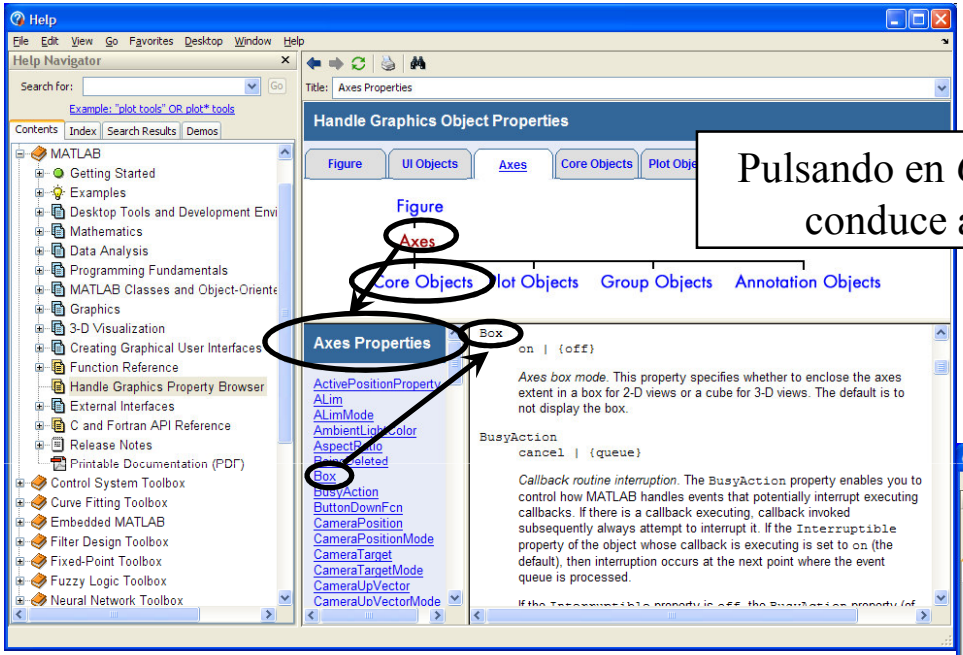
  `set`(hdl, 'KeyWord', KeyWordValue, …)

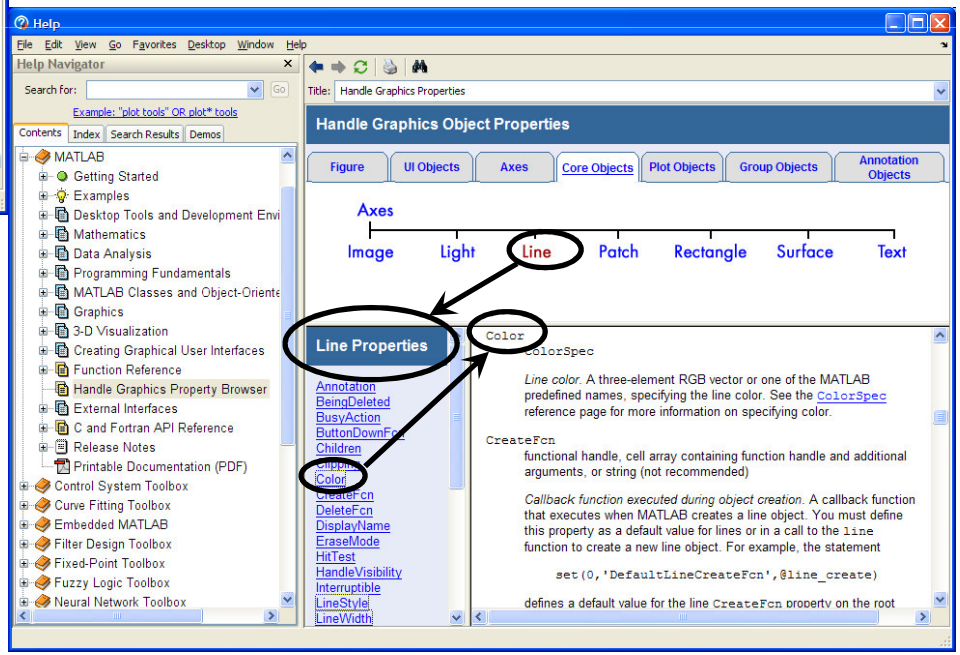
- Los keywords y sus valores apropiados se pueden determinar usando el fichero Help

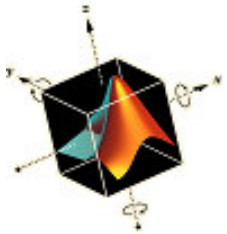


Pulsando en *Axes* conduce a la ventana de detalle

Cambio de atributos de línea y punto







Ejemplos de gráficos

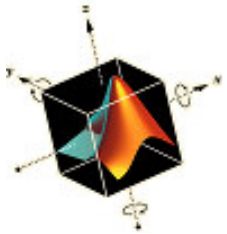
- Puntos

```
plot(2, 4, 'r*')
```

- Líneas

```
plot([0, 1], [0, 2])
```

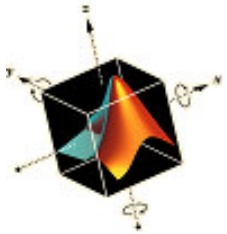
```
x = 2:2:8;
y = [zeros(1, length(x)); cos(pi*x/20)];
plot([x; x],y, 'k') | plot([x; x], y, 'k', x, cos(pi*x/20), 'rs')
axis([1, 9, 0, 1])
```



Ejemplos de gráficos

- Líneas y puntos

```
x = 2:2:8;
y = [zeros(1, length(x)); cos(pi*x/20)];
plot([x; x], y, 'k')
hold on
plot(x, cos(pi*x/20), 's', 'MarkerEdgeColor', 'b', ...
    'MarkerFaceColor', 'r', 'MarkerSize', 14)
axis([1, 9, 0, 1])
```



Ejemplos de gráficos

- Círculos

  Para dibujar un círculo de radio r con centro en (*a*, *b*) se usa

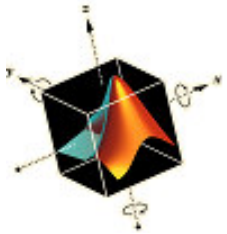
$$x = a + r\cos(\theta)$$

$$y = b + r\sin(\theta)$$

  donde $0 \le \theta \le \theta_1 \le 2\pi$

  Script para dibujar círculo $\theta_1 = 2\pi$, $a = 1$, $b = 2$, and $r = 0.5$

```
theta = linspace(0, 2*pi);
plot(1+0.5*cos(theta), 2+0.5*sin(theta))
axis equal
```



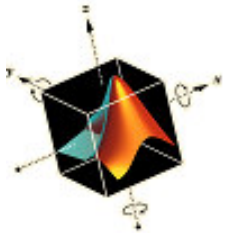
Ejemplos de gráficos

- Círculos

  Para dibujar una familia de seis círculos concéntricos con radio inicial 0.5 e incremento de 0.25 y centros indicados por un signo +

  Script:

```
theta = linspace(0, 2*pi, 50);      % (1×50)
rad = 0.5:0.25:1.75;                % (1×6)
x = 1+cos(theta)'*rad;              % (50×6)
y = 2+sin(theta)'*rad;              % (50×6)
plot(x, y, 'k', 1, 2, 'k+')
axis equal
```



Ejemplos de gráficos

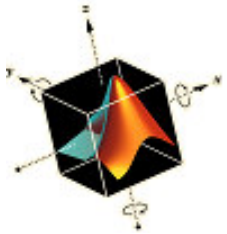
- Familia de Curvas

  Matlab permite representar el eje *x* por un vector y el eje *y* por una matriz. Dibujará las curvas según el vector y las columnas o filas de la matriz, dependiendo de cual coincide con la longitud del vector

  Script: Dibujo de familia de parábolas $y = a^2 - x^2$

  donde $-5 \leq x \leq 5$ e incrementos 0.2 y $a = 1, 2, \ldots, 5$

```
x = –5:0.2:5;                          % (1×51)
a = 1:5;                               % (1×5)
[xx, aa] = meshgrid(x.^2, a.^2);       % (5×51)
plot(x, aa–xx, 'k')
```



Ejemplos de gráficos

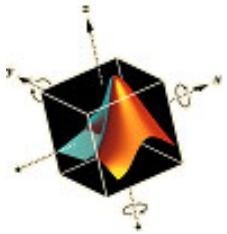
- Familia de Curvas

  Script: Visualización de la convergencia de series

$$S_N = \sum_{j=1}^{N} \frac{1}{(a+j)^2}$$

  para N = 1, 2, …, 10 y a = 1, 2, y 3

```
aa = 1:3;                          % (1×3)
N = 1:10;                          % (1×10)
[a, k] = meshgrid(aa, N);          % (10×3)
S = cumsum(1./(a+k).^2);           % (10×3)
plot(N, S, 'ks-')
```



Gráficas de múltiples funciones en una figura

- Considerando las tres funciones

$$g_1(x) = 0.1x^2$$

$$g_2(y) = \cos^2 y$$

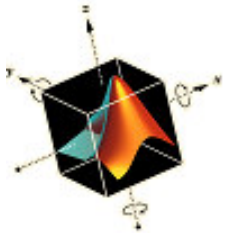
$$g_3(z) = e^{-0.3z}$$

donde $0 \leq x, y, z \leq 3.5$

Se puede graficar las tres funciones en una figura de tres maneras:

Modo 1:

```
x = linspace(0, 3.5);
plot(x, [0.1*x.^2; cos(x).^2; exp(-0.3*x)], 'k')
```



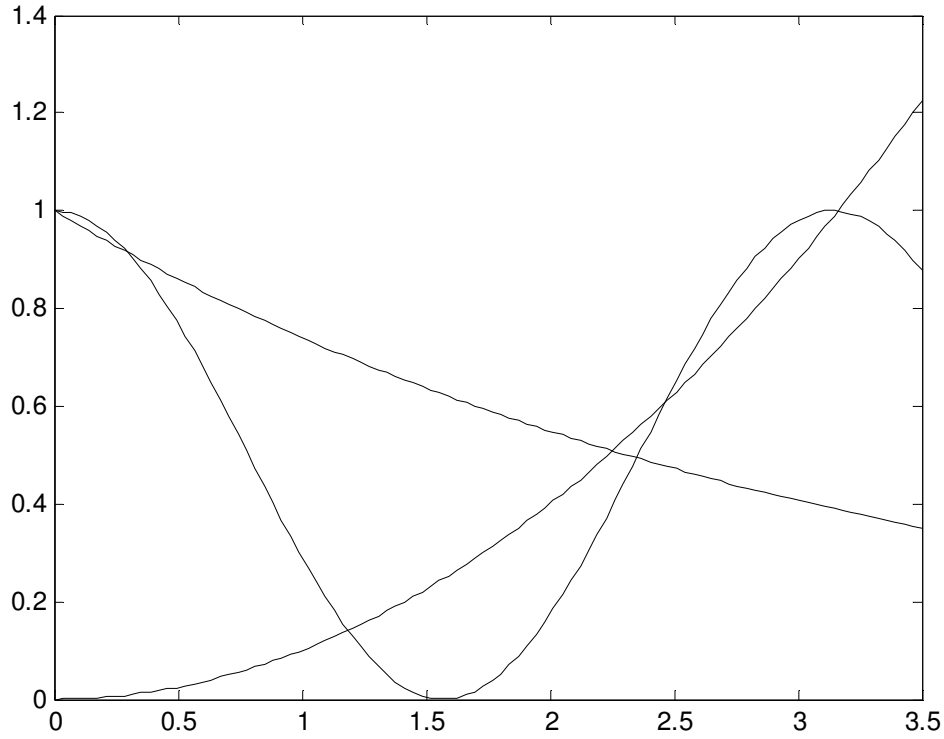
Gráficas de múltiples funciones en una figura

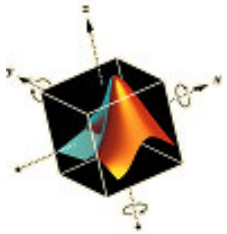
Modo 2:

```
x = linspace(0, 3.5);
plot(x, 0.1*x.^2, 'k', x, cos(x).^2, 'k', x, exp(-0.3*x), 'k')
```

Modo 3:

```
x = linspace(0, 3.5);
plot(x, 0.1*x.^2, 'k')
hold on
plot(x, cos(x).^2, 'k')
plot(x, exp(-0.3*x), 'k')
```



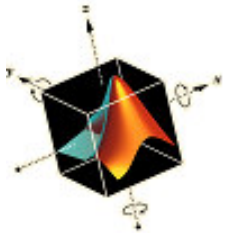


Gráficas de múltiples funciones en una figura

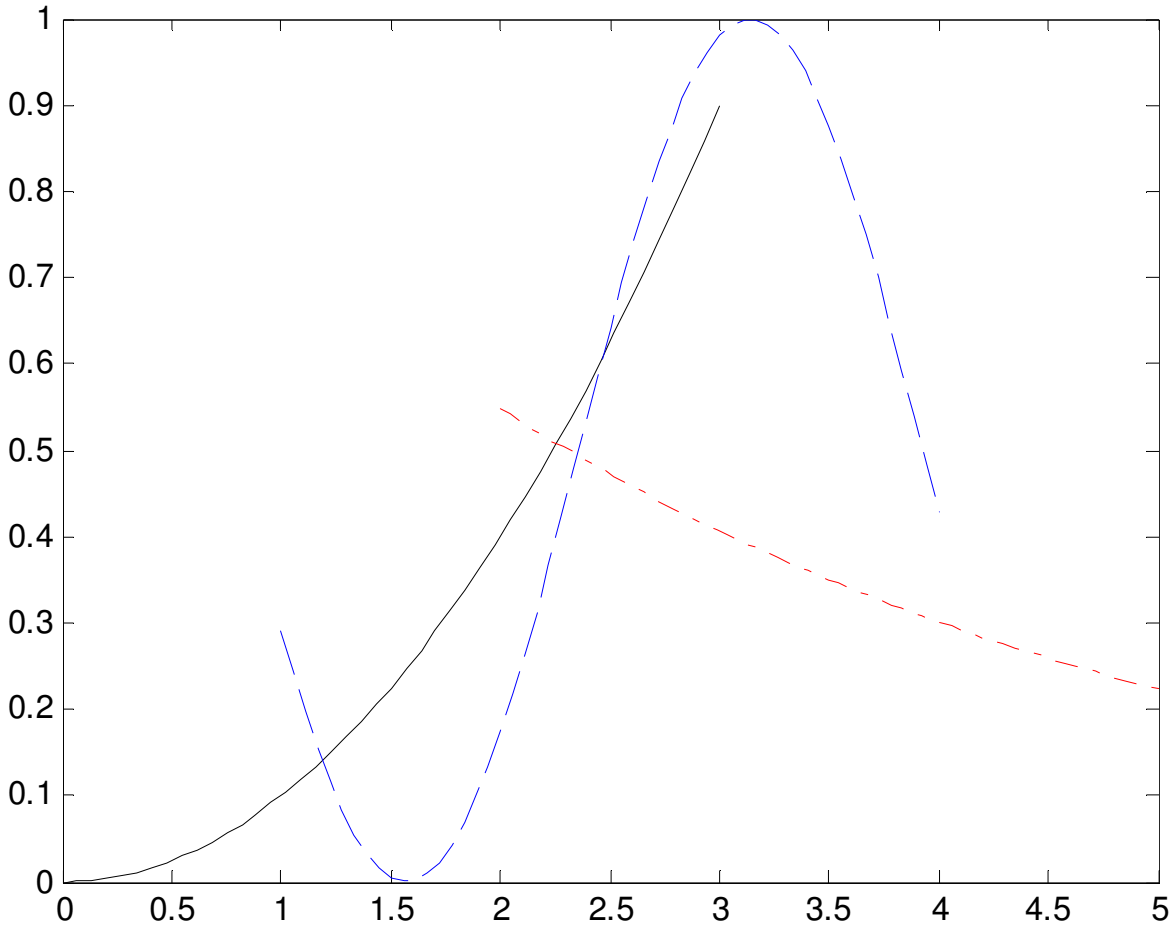
- Si el rango de las variables independientes de cada de las funciones es diferente, sólo se pueden usar los modos 2 y 3

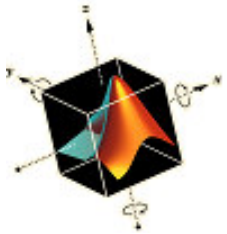
  pe si $0 \le x \le 3$, $1 \le y \le 4$, y $2 \le z \le 5$, el script es:

```
x = linspace(0, 3, 45);
y = linspace(1, 4, 55);
z = linspace(2, 5, 65);
plot(x, 0.1*x.^2, 'k-', y, cos(y).^2, 'b--', z, exp(-0.3*z), 'r-.')
```



Gráficas de múltiples funciones en una figura





Cambio de apariencia de las gráficas

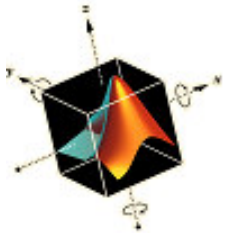
- box, grid, and axis
  - Se pueden usar varias funciones para cambiar la apariencia de un gráfico

`axis on` or `axis off` [default – on]

`box on` or `box off` [default – on]

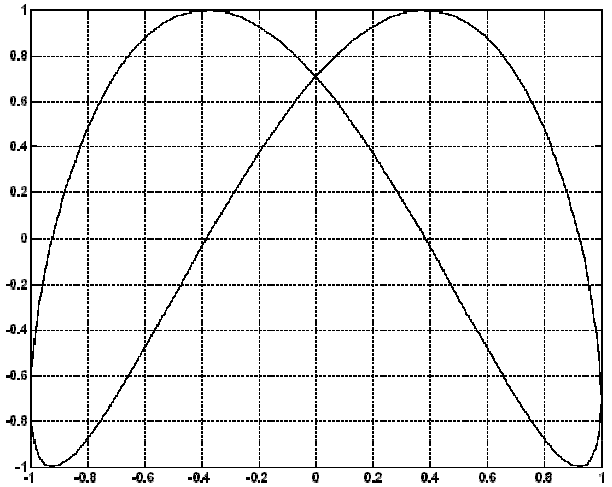
`grid on` or `grid off` [default – off]

La función `box on` sólo funciona cuando se ha seleccionado `axis on`

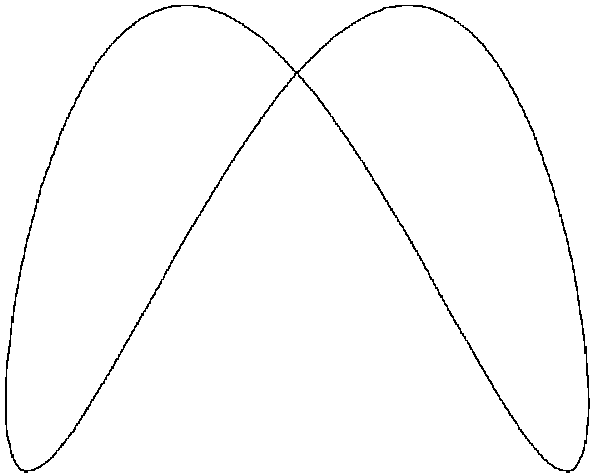


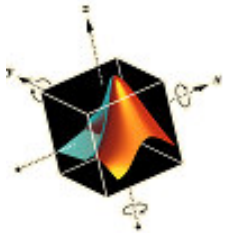
Cambio de apariencia de las gráficas

```
th = linspace(0, 2*pi, 101);
x = sin(th);
y = sin(2*th+pi/4);
plot(x, y, 'k-')
box on
grid on
```



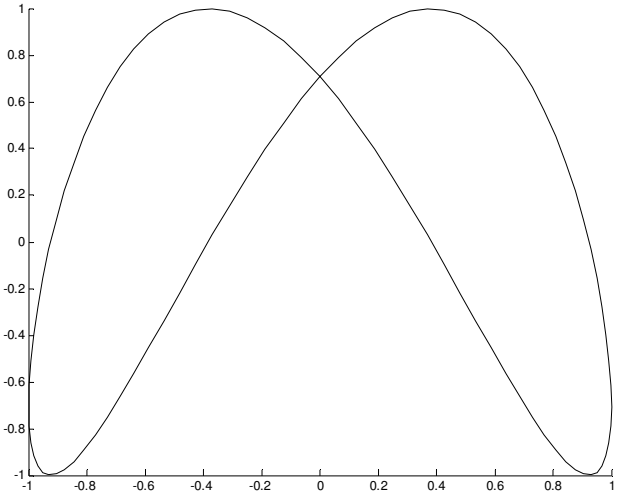

```
th = linspace(0, 2*pi, 101);
x = sin(th);
y = sin(2*th+pi/4);
plot(x, y, 'k-')
box off
grid off
axis off
```

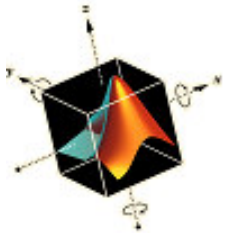




Cambio de apariencia de las gráficas

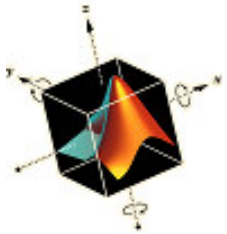
```
th = linspace(0, 2*pi, 101);
x = sin(th);
y = sin(2*th+pi/4);
plot(x, y, 'k-')
box off
grid off
```





Gráficos de propósito especial

- semilogx, semilogy, y loglog
  - `semilogx` grafica el eje-*x* en escala log base 10
  - `semilogy` grafica el eje-*y* en escala base 10
  - `loglog` grafica ambos ejes en escala log base 10
- Stairs, stem y bar
  - `stairs` gráfica en escalera de los valores *y*
  - `stem` dibuja líneas desde el eje-*x* al valor y
  - `bar` dibuja barras para cada elemento y



Ejemplos de gráficos de propósito especial

- Se aplica los comandos anteriores a alguna parte de la siguiente expresión $F(\Omega) = H(\Omega)e^{j\theta(\Omega)} \quad \Omega \geq 0$

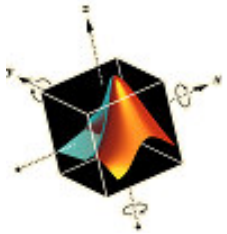
  donde $\zeta < 1$ y $H(\Omega) = \dfrac{1}{\sqrt{\left(1-\Omega^2\right)^2 + \left(2\zeta\Omega\right)^2}}$

$$\theta(\Omega) = \tan^{-1}\frac{2\zeta\Omega}{1-\Omega^2}$$

Se crea la función

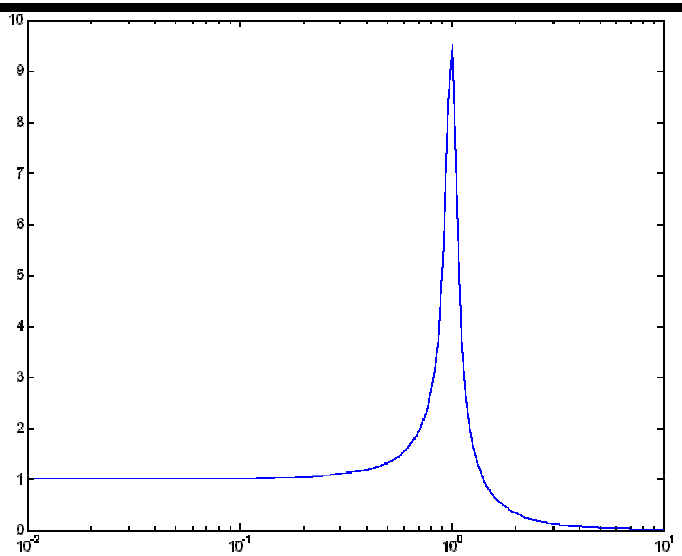
```
function [H, T] = FOm(Om, z)
T = atan2(2*z*Om, 1-Om.^2)*180/pi;
H = 1./sqrt((1-Om.^2).^2+(2*z*Om).^2);
```

donde $T = \theta(\Omega)$ se expresa en grados y $z = \zeta$

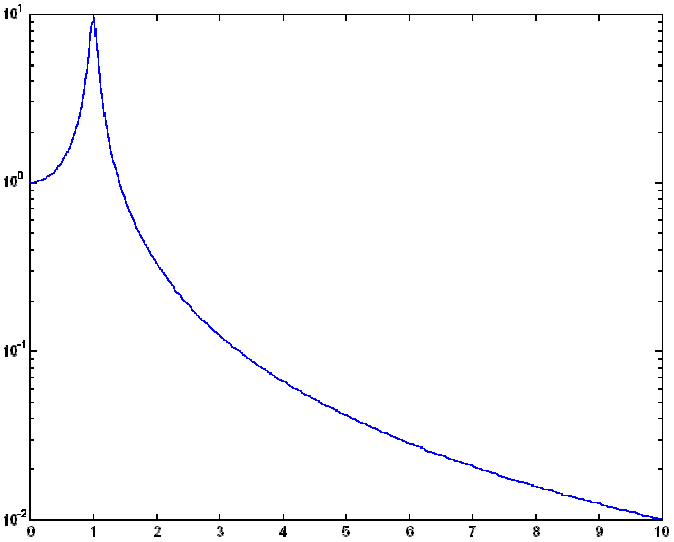


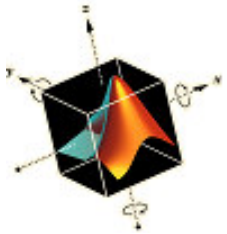
Ejemplos de gráficos de propósito especial

```
Om = linspace(0.01, 10, 200);
[H, T] = FOm(Om, 0.05);
semilogx(Om, H)
```



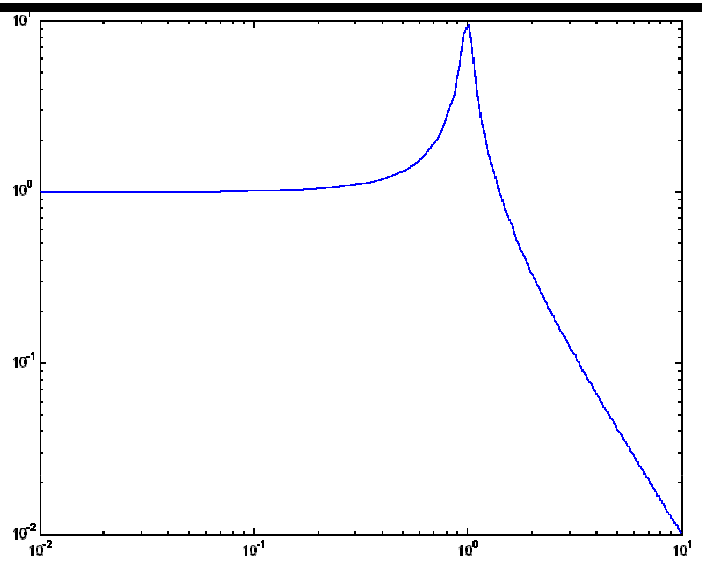
```
Om = linspace(0.01, 10, 200);
[H, T] = FOm(Om, 0.05);
semilogy(Om, H)
```



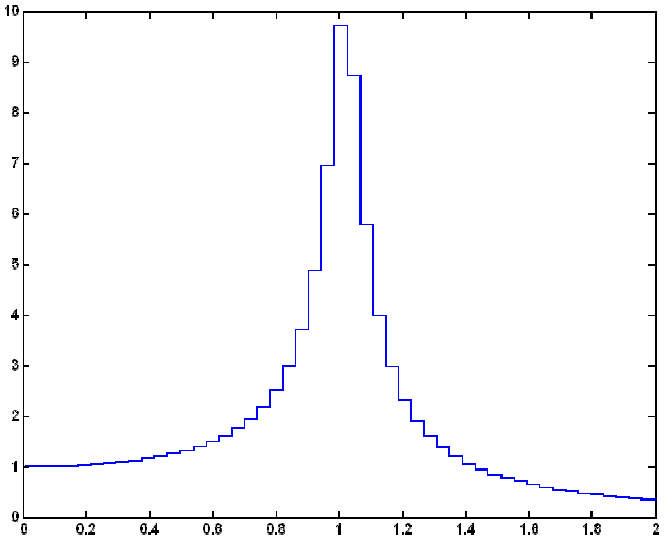


Ejemplos de gráficos de propósito especial

```
Om = linspace(0.01, 10, 200);
[H, T] = FOm(Om, 0.05);
loglog(Om, H)
```

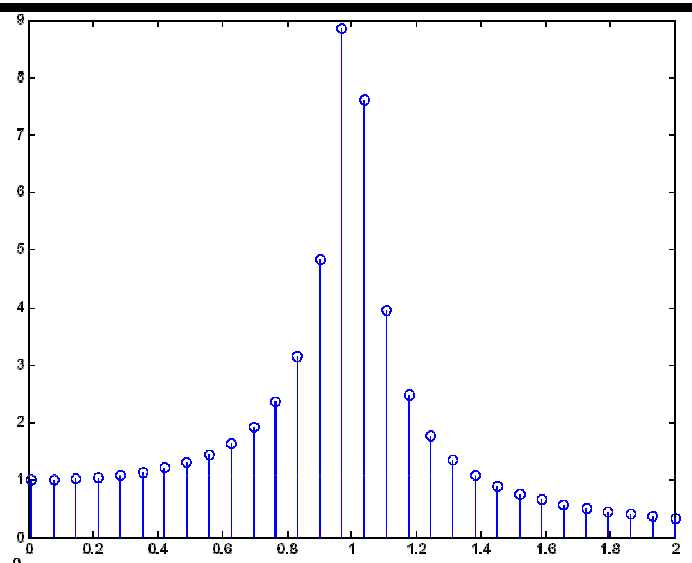


```
Om = linspace(0.01, 2, 50);
[H, T] = FOm(Om, 0.05);
stairs(Om, H)
```



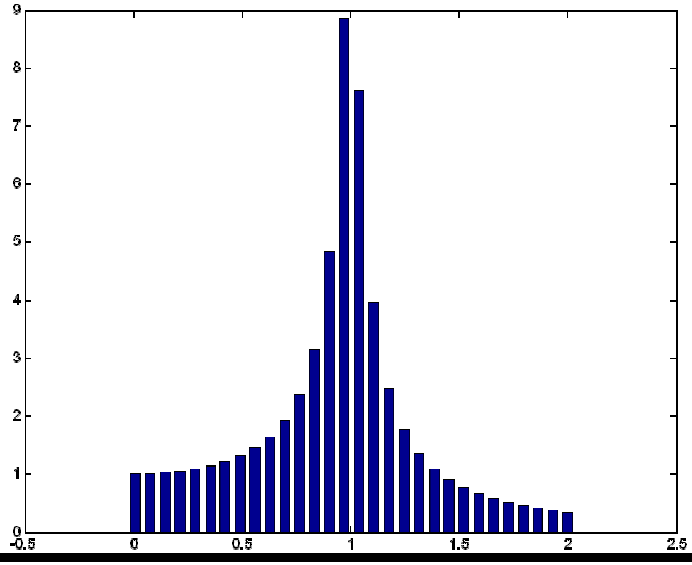
Ejemplos de gráficos de propósito especial

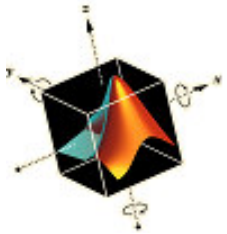
```
Om = linspace(0.01, 2, 30);
[H, T] = FOm(Om, 0.05);
stem(Om, H)
```



```
Om = linspace(0.01, 2, 30);
[H, T] = FOm(Om, 0.05);
bar(Om, H, 0.6)
```

ancho de bar – defecto = 0.8
(20% de ancho de bar es espacio blanco)





plotyy

- Crea un gráfico que consiste de dos funciones diferentes cada uno con dos rangos diferentes de valores de *x* e *y*. La función es

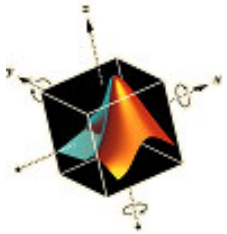
  plotyy(x1, y1, x2, y2, 'function_1', 'function_2')

  donde 'function_1' and 'function_2' pueden ser

  `plot`, `semilogx`, `semilogy`, `loglog`, o `stem`

  Es equivalente a

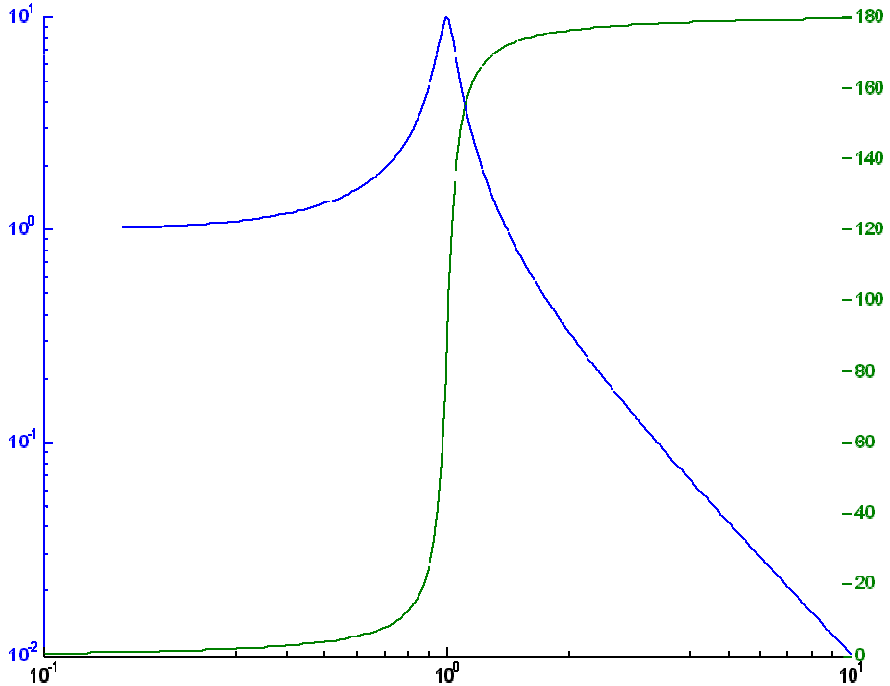
  function_1(x1, y1)
  `hold on`
  function_2(x2, y2)

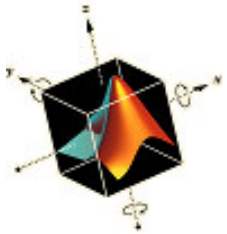


Ejemplos plotyy

- Si se quiere obtener $H(\Omega)$ y $\theta(\Omega)$ en la misma grafica

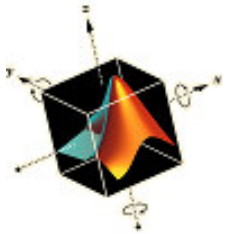
```
Om = logspace(-1, 1, 200);
[H, T] = FOm(Om, 0.05);
plotyy(Om, H, Om, T, 'loglog', 'semilogx')
```





Gráficos de propósito especial

- convhull, delauney, y voronoi aplicados a un conjunto de puntos P
  - `convexhull` dibuja la envolvente convexa (menor polígono que encierra un conjunto de puntos en un plano)
  - `delauney` crea un conjunto de triángulos tal que ningún punto se encuentra en un círculo circunscrito de un triángulo. La salida de `delauney` se grafica con `triplot` (función para dibujar triángulos en un plano)
  - `voronoi` dibuja un polígono convexo alrededor de cada punto P tal que cada segmento de línea del polígono es el bisector perpendicular entre P y sus vecinos

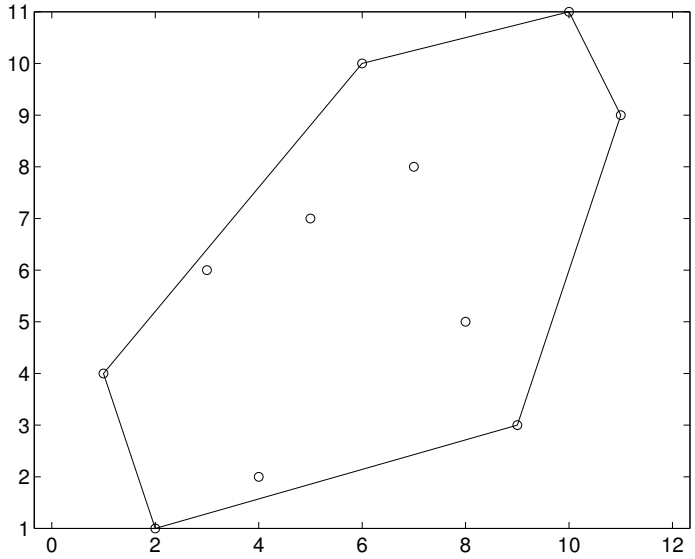


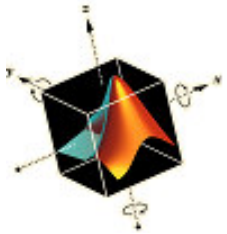
Ejemplos convhull, delauney, y voronoi

- Se crea una función fichero M que contiene un conjunto de pares de coordenadas *x-y*

```
function [x, y] = PointSet
x = [1, 3, 5, 2, 4, 6, 7, 9, 10, 8, 11];
y = [ 4, 6, 7, 1, 2, 10, 8, 3, 11, 5, 9];
```

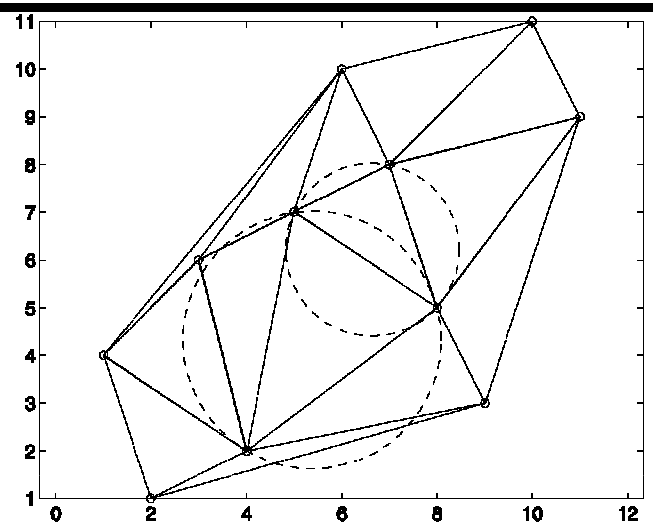
```
[x, y] = PointSet;
n = convhull(x, y);
plot(x(n),y(n), 'k-', x, y, 'ok')
axis equal
```



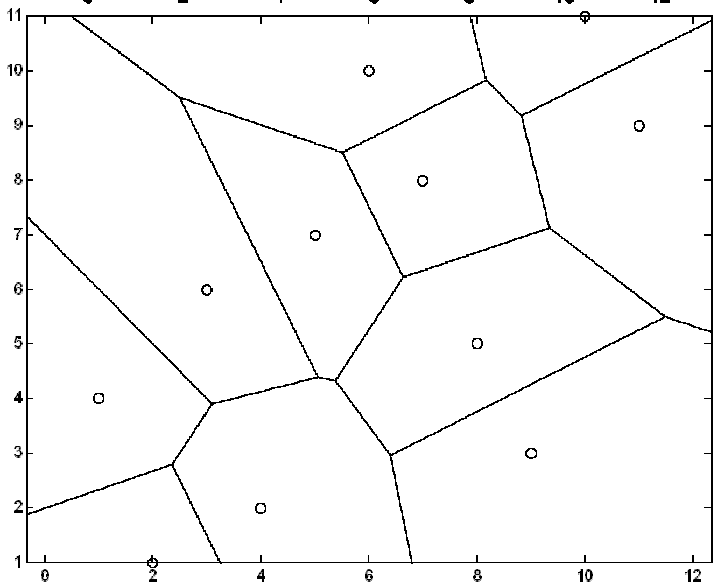


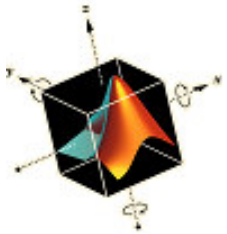
Ejemplos convhull, delauney, y voronoi

```
[x, y] = PointSet;
tr = delaunay(x,y);
triplot(tr,x,y,'k')
hold on
axis equal
plot(x, y, 'ok')
```




```
[x, y] = PointSet;
voronoi(x, y, 'ko')
axis equal
```



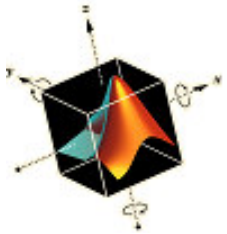


Gráficos de propósito especial

- `pie` y `pie3`
  - Permiten crear gráficos tipo tarta. La sintaxis es:

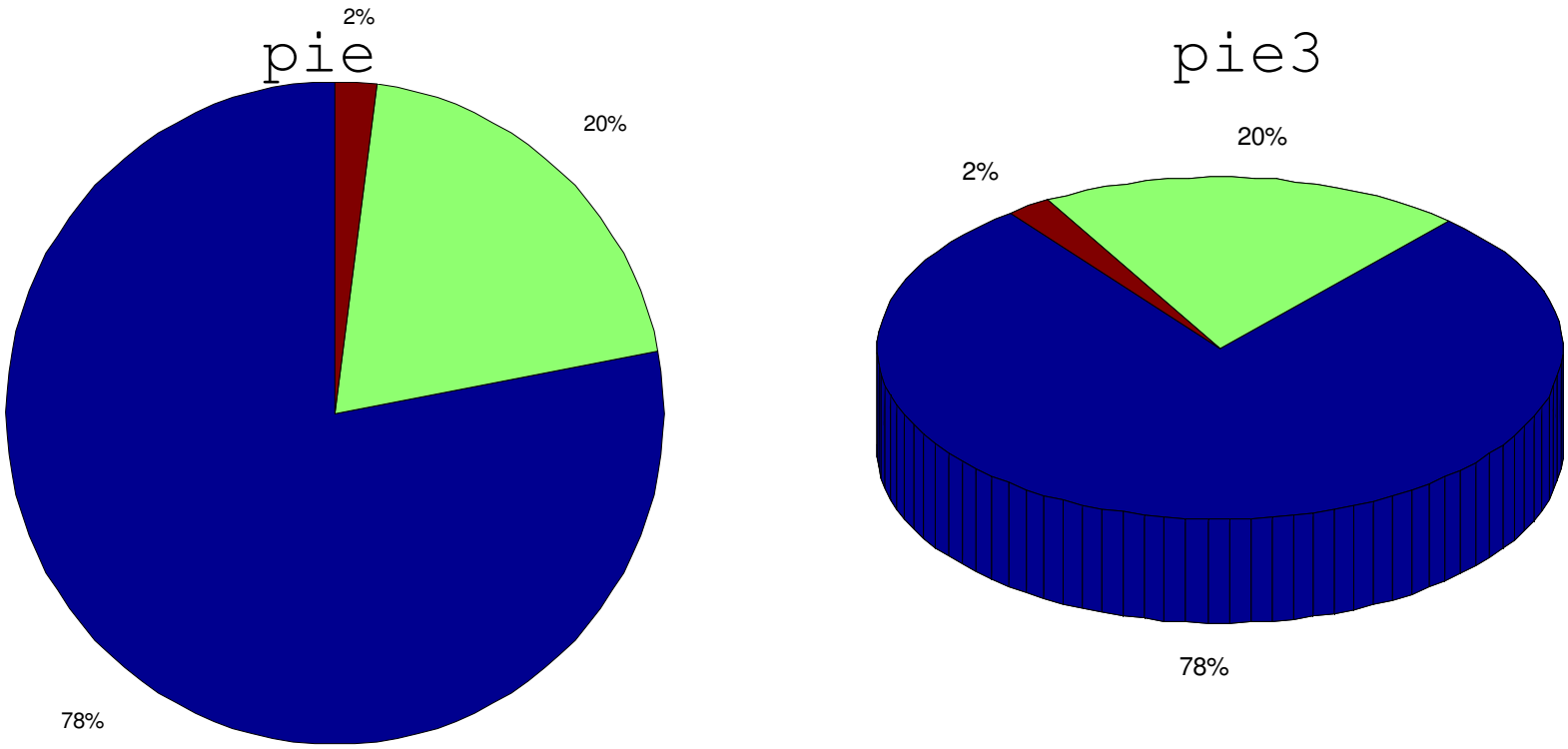
    `pie`(d, expl, label) y `pie3`(d, expl, label)

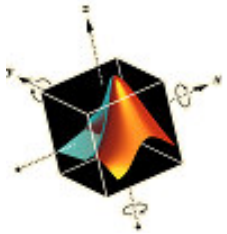
    donde
    d es un vector de longitud *n* a partir del cual se construirá la tarta
    expl es un vector opcional de longitud *n* consistentes de 1's y 0's para indicar los sectores del pie deben ser separados
    label es una celda opcional de longitud *n* con etiquetas para cada sector del pie



Ejemplo de pie y pie3

```
dat = [39, 10, 1];
pie(dat)   % o pie3(...)
```

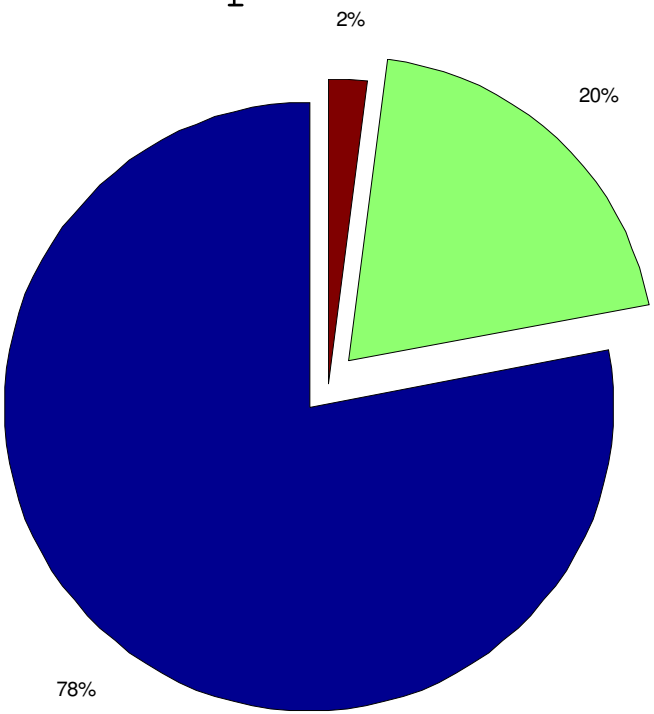




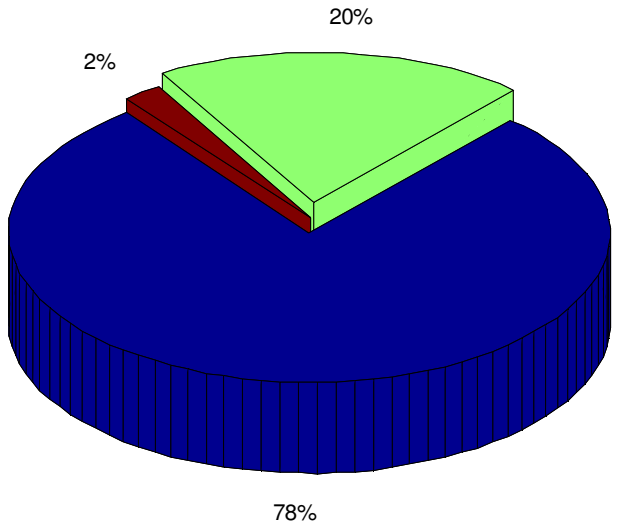
Ejemplo de pie y pie3

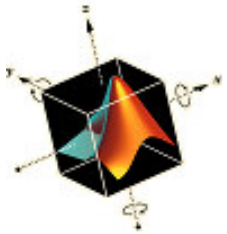
```
dat = [39, 10, 1];
pie(dat, [1, 1, 0] )   % o pie3(...)
```

pie



pie3

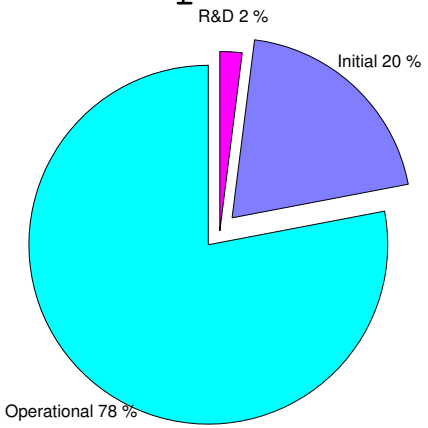




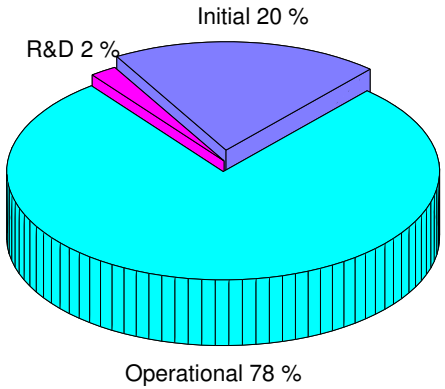
Ejemplo de pie y pie3

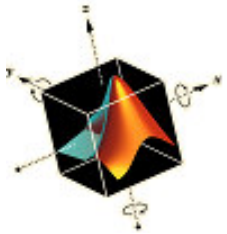
```
dat = [39, 10, 1];
dat = 100*dat/sum(dat);
A = ['Operational ' num2str(dat(1)) ' %'];
B = ['Initial ' num2str(dat(2)) ' %'];
C = ['R&D ' num2str(dat(3)) ' %'];
colormap('cool')
pie(dat, [1, 1, 0],{A, B, C})     % o pie3(...)
```

pie



pie3





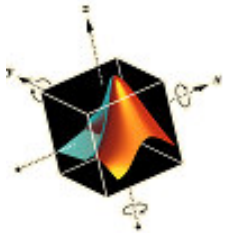
Lectura, visualización y manipulación de imágenes digitales

- Matlab puede leer 15 formatos de imágenes digitales diferentes, algunos de los más comunes son:
  - jpeg (joint photographic experts group)
  - bmp (Windows bit map)
  - tiff (tagged image file format)
  - gif (graphics interchange format)

- La función para leer una imagen es:

  A = `imread`('FileName', 'fmt')

  donde *FileName* es el nombre del fichero que contiene la imagen digital en el formato especificado por *fmt*

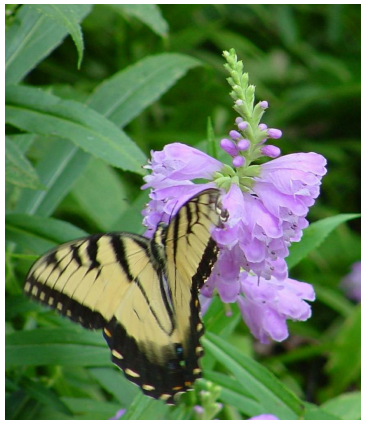


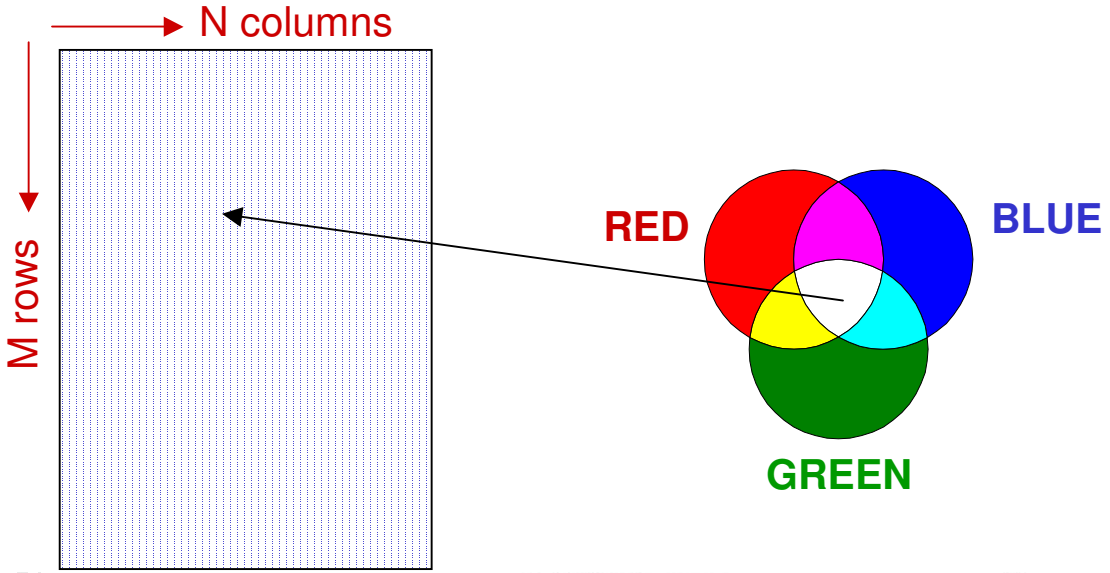
Lectura, visualización y manipulación de imágenes digitales

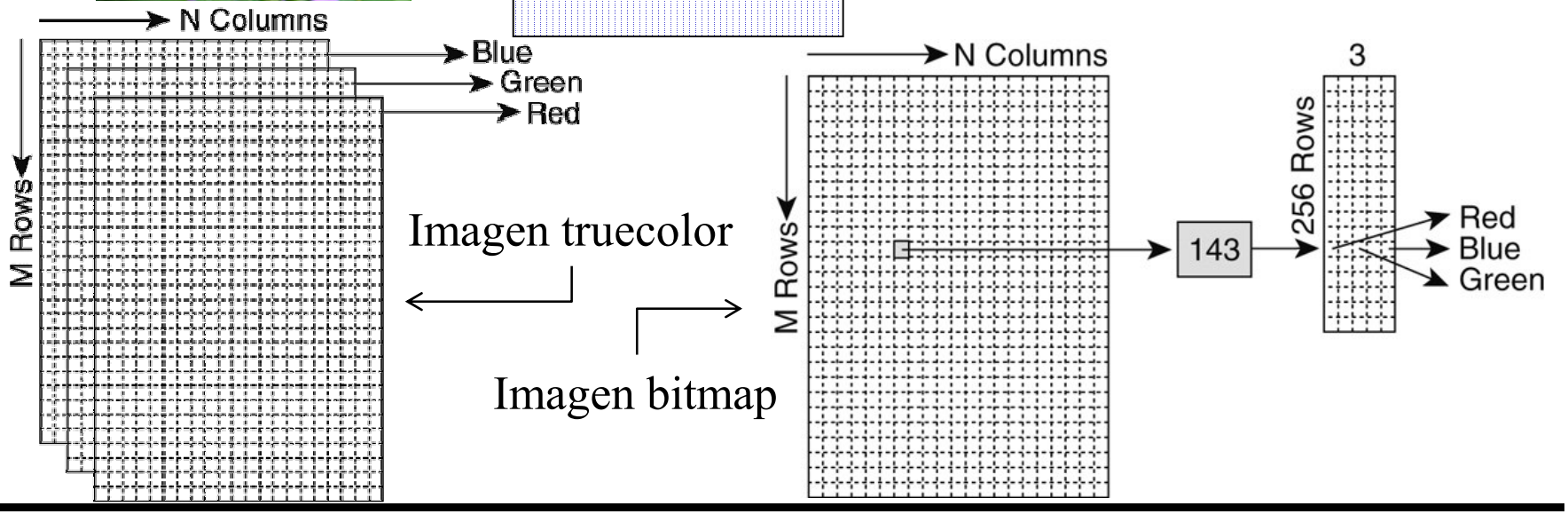
- El array *A* es una array (*N*×*M*×3) donde
  - (*n*×*m*) es la posición del pixel en el array.
  - *A*(*n*, *m*, 1), *A*(*n*, *m*, 2), y *A*(*n*, *m*, 3) son componentes del triplete red-green-blue (RGB , rojo-verde-azul) para cada pixel de una imagen en color.
- Los valores de cada componente del triplete varáin de 0 to 255.
- Por ejemplo, el color amarillo se representa como

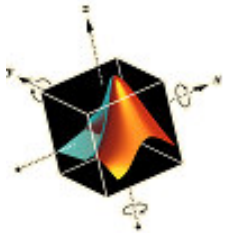
  A(*n*, *m*, 1) = 255
  A(*n*, *m*, 2) = 255
  A(*n*, *m*, 3) = 0

Lectura, visualización y manipulación de imágenes digitales







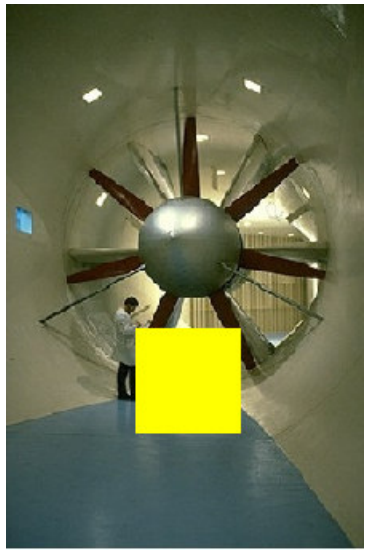


Lectura, visualización y manipulación de imágenes digitales

```
A = imread('WindTunnel.jpg', 'jpeg');
image(A)
axis image off
```

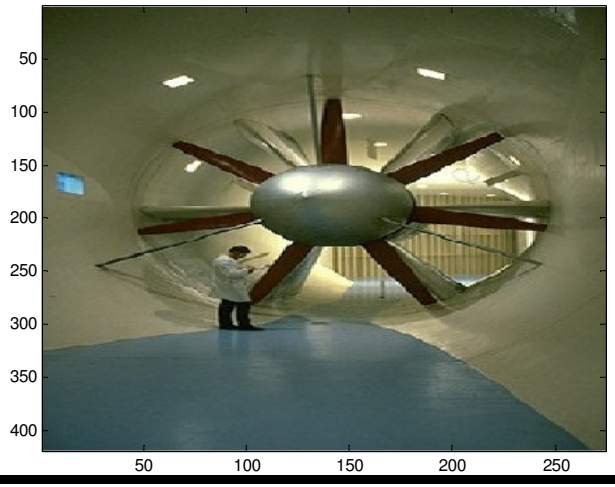
El tamaño de *A* es (419×274×3)

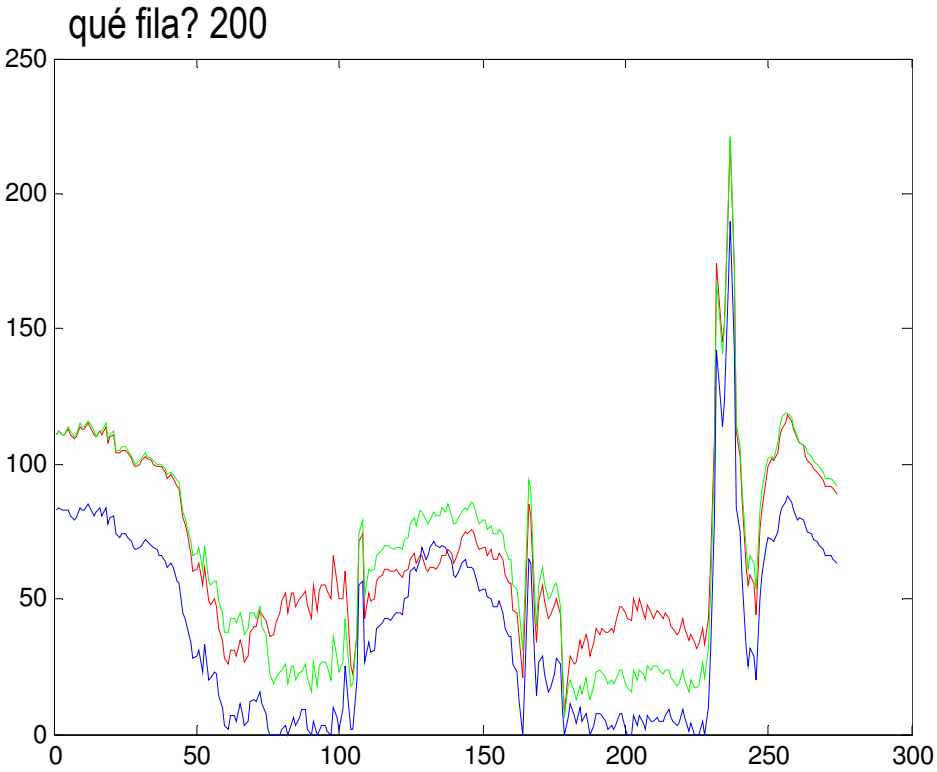
```
A = imread('WindTunnel.jpg', 'jpeg');
A(250:330, 100:180, 1) = 255;
A(250:330, 100:180, 2) = 255;
A(250:330, 100:180, 3) = 0;
image(A)
axis image off
```

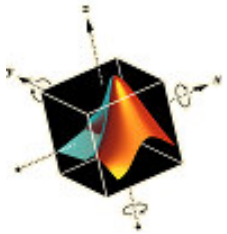


Lectura, visualización y manipulación de imágenes digitales

```
A = imread('WindTunnel.jpg', 'jpeg');
image(A)
figure
row=input('qué fila? ');
red = v(row, :, 1);
gr  = v(row, :, 2);
bl  = v(row, :, 3);
plot(red, 'r');
hold on
plot(gr, 'g');
plot(bl, 'b');
```

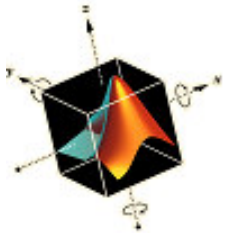






Anotaciones en gráficos y mejoras visuales

- Se puede mejorar un gráfico mediante
  - Añadir etiquetas a los ejes, títulos a las figuras, gráficas etiquetadas, leyendas, áreas de relleno y texto
  - Alterar los atributos de los ejes, líneas de las curvas, y texto. Uso de letras griegas, símbolos matemáticas, subindices y superindices
  - Colocando una figura dentro de otra.
  - Uso de las herramientas interactivas para gráficas
  - Uso de animación



Etiquetas de ejes y curvas, títulos y texto

- Para colocar etiquetas en los ejes *x* e *y* y título al gráfico se usan los comandos

  `xlabel`(s)

  `ylabel`(s)

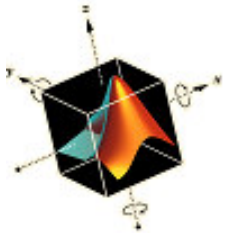
  `title`(s)

  donde *s* es un string

- Para colocar texto en cualquier posición se usa

  `text`(x, y, s)

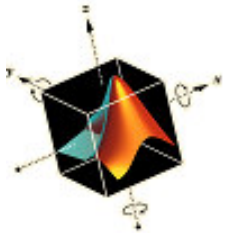
  donde *x* e *y* son las coordenadas del texto y *s* es el string



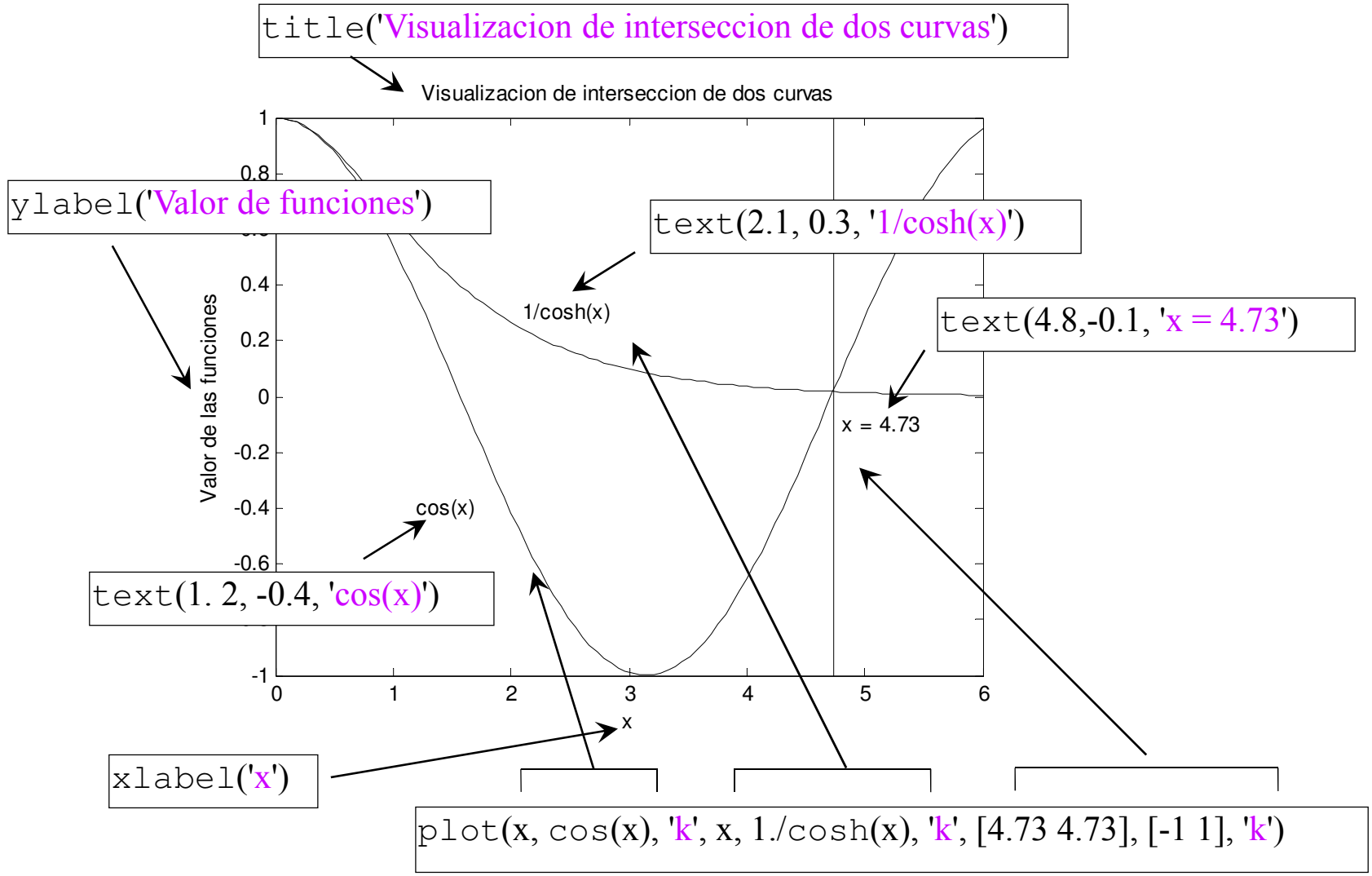
Ejemplo: Etiquetas de ejes y curvas, títulos y texto

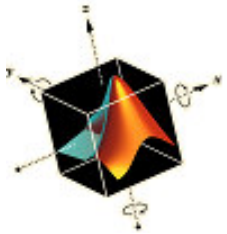
- Dibujo con etiquetas, título y anotación de la intersección de dos curvas: cos(*x*) y 1/cosh(*x*), en el rango $0 \leq x \leq 6$. Se dibuja una línea vertical en $x = 4.73$ y se indica el valor de x cerca a la intersección

```
x = linspace(0, 6, 100);
plot(x, cos(x), 'k', x, 1./cosh(x), 'k', [4.73, 4.73], [-1, 1], 'k')
xlabel('x')
ylabel('Valor de las funciones')
title('Visualizacion de interseccion de dos curvas')
text(4.8, -0.1, 'x = 4.73')
text(2.1, 0.3, '1/cosh(x)')
text(1.2, -0.4, 'cos(x)')
```



Ejemplo: Etiquetas de ejes y curvas, títulos y texto





Leyendas

- Para identificar las curvas se puede usar

  `legend`(s1, s2, …, sn, 'Location', 'Keyword')

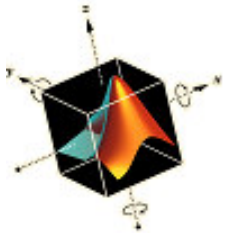
  donde

  *s*1,etc, son strings que contiene el identificador para cada línea en el recuadro de leyenda en orden de aparición

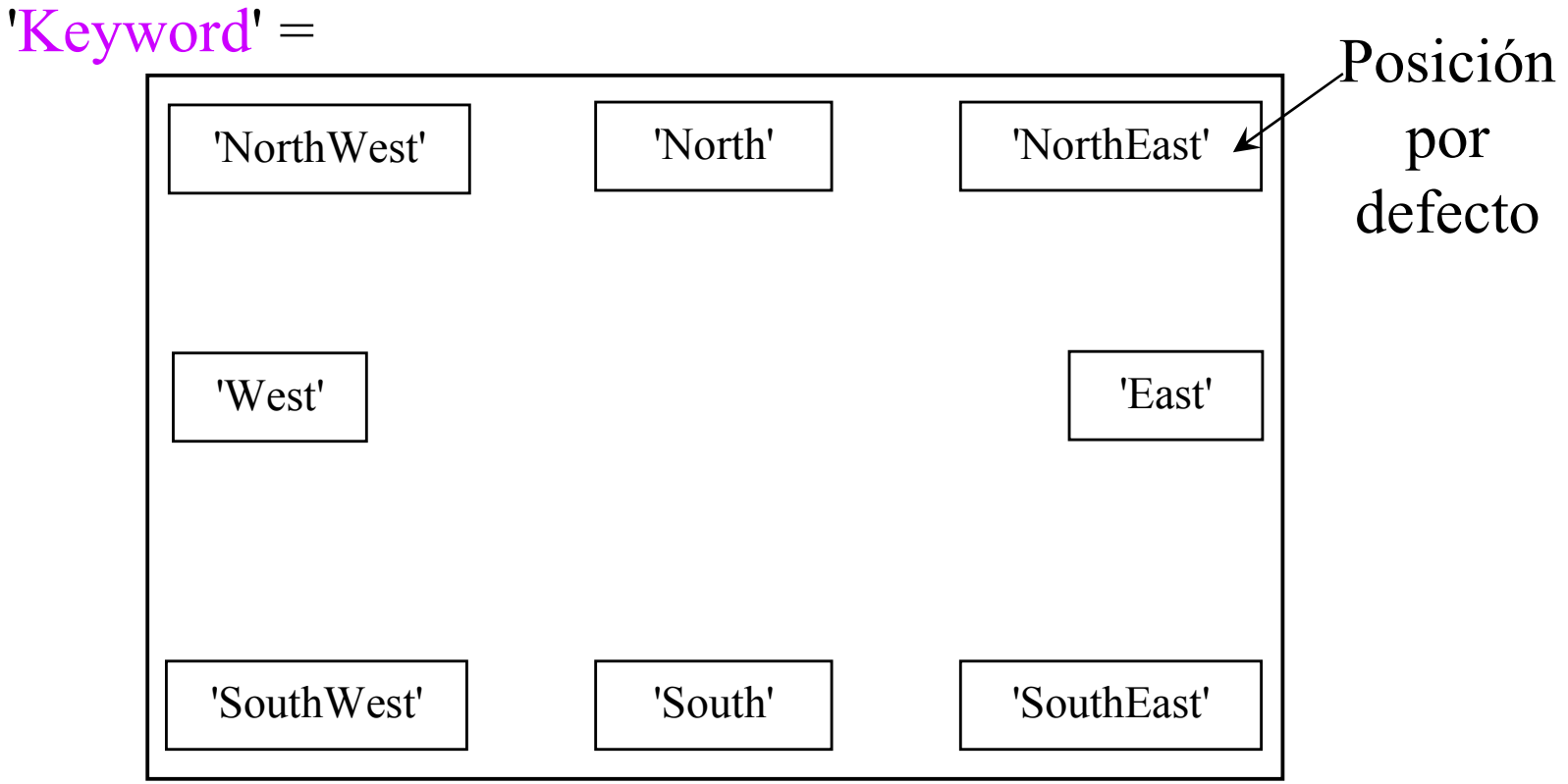
  Keyword es un string opcional que coloca la leyenda en uno de 8 lugares predeterminados en el gráfico

  El número de argumentos de leyenda es menor o igual al número de curvas correspondientes a las funciones

  La leyenda se coloca después de todas las funciones a graficar

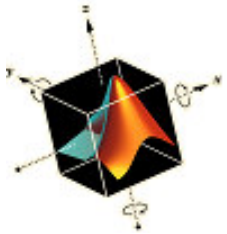


Posición de las leyendas



- El rectángulo de la leyenda se puede ocultar con

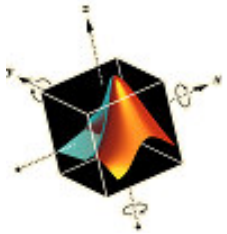
  `legend('boxoff')` el valor por defecto es `legend('boxon')`



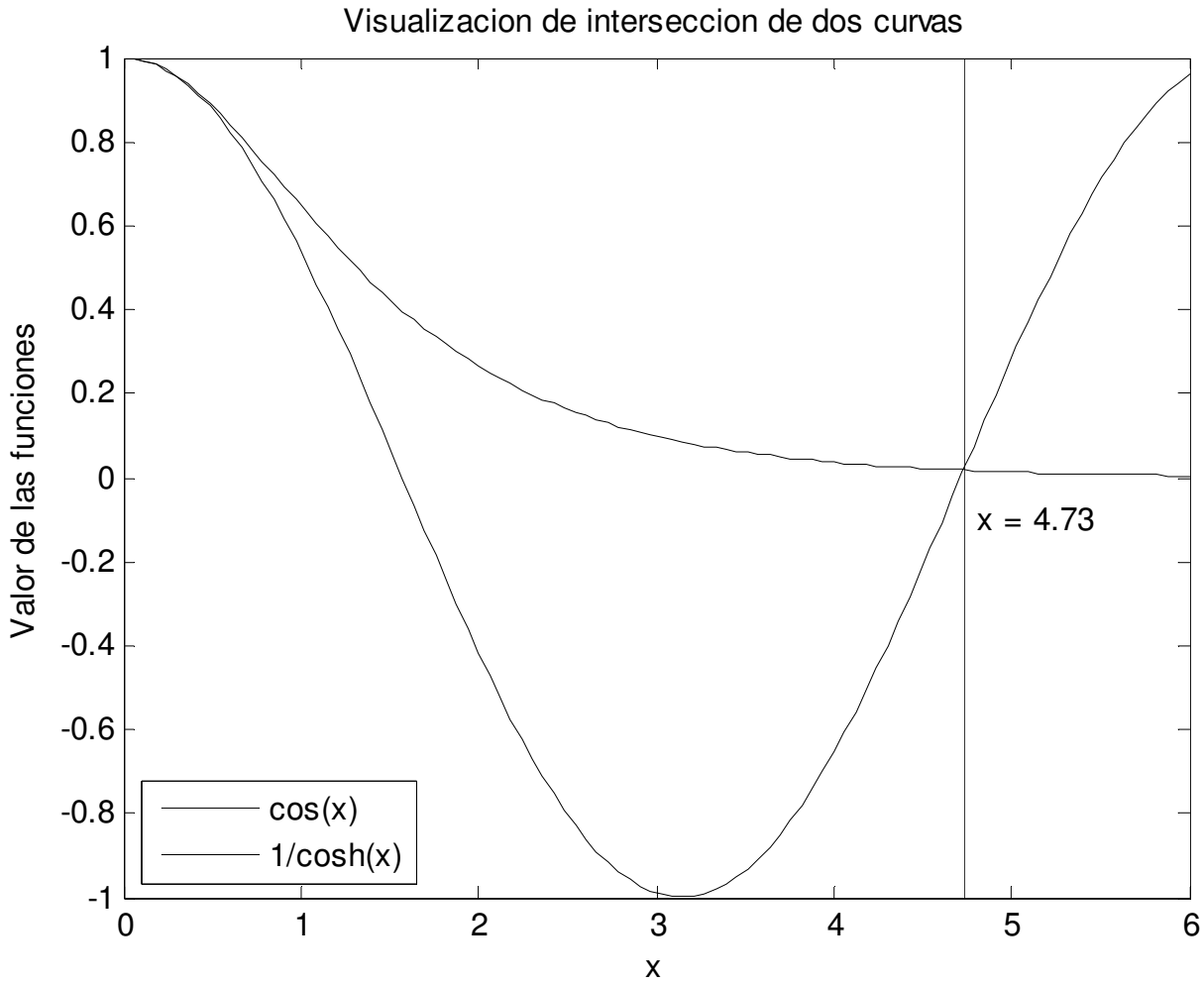
Ejemplo: leyendas

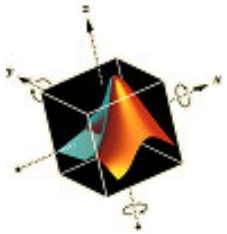
- `legend` difiere de `text` en que `text` se puede usar muchas veces mientras que `legend` sólo se puede usar una vez

```
x = linspace(0, 6, 100);
plot(x, cos(x), 'k', x, 1./cosh(x), 'k', [4.73, 4.73], [-1, 1], 'k')
xlabel('x')
ylabel('Valor de las funciones')
title('Visualizacion de interseccion de dos curvas')
text(4.8, -0.1, 'x = 4.73')
legend('cos(x) ', '1/cosh(x) ', 'Location', 'Southwest')
```



Ejemplo: Leyendas





Relleno de regiones

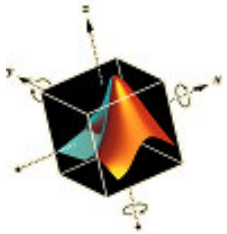
- Una región de un gráfico se puede resaltar mediante su coloreado o mediante el uso de algún elemento geométrico para identificarlo
- Para rellenar un área contenida dentro de una region poligonal se usa:

  `fill`(x, y, c)

  donde:

  x e *y* son arrays de la misma longitud que representan los vértices de las aristas del polígono cerrado

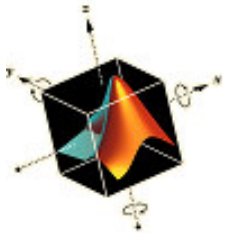
  c es un string que indica el color de relleno



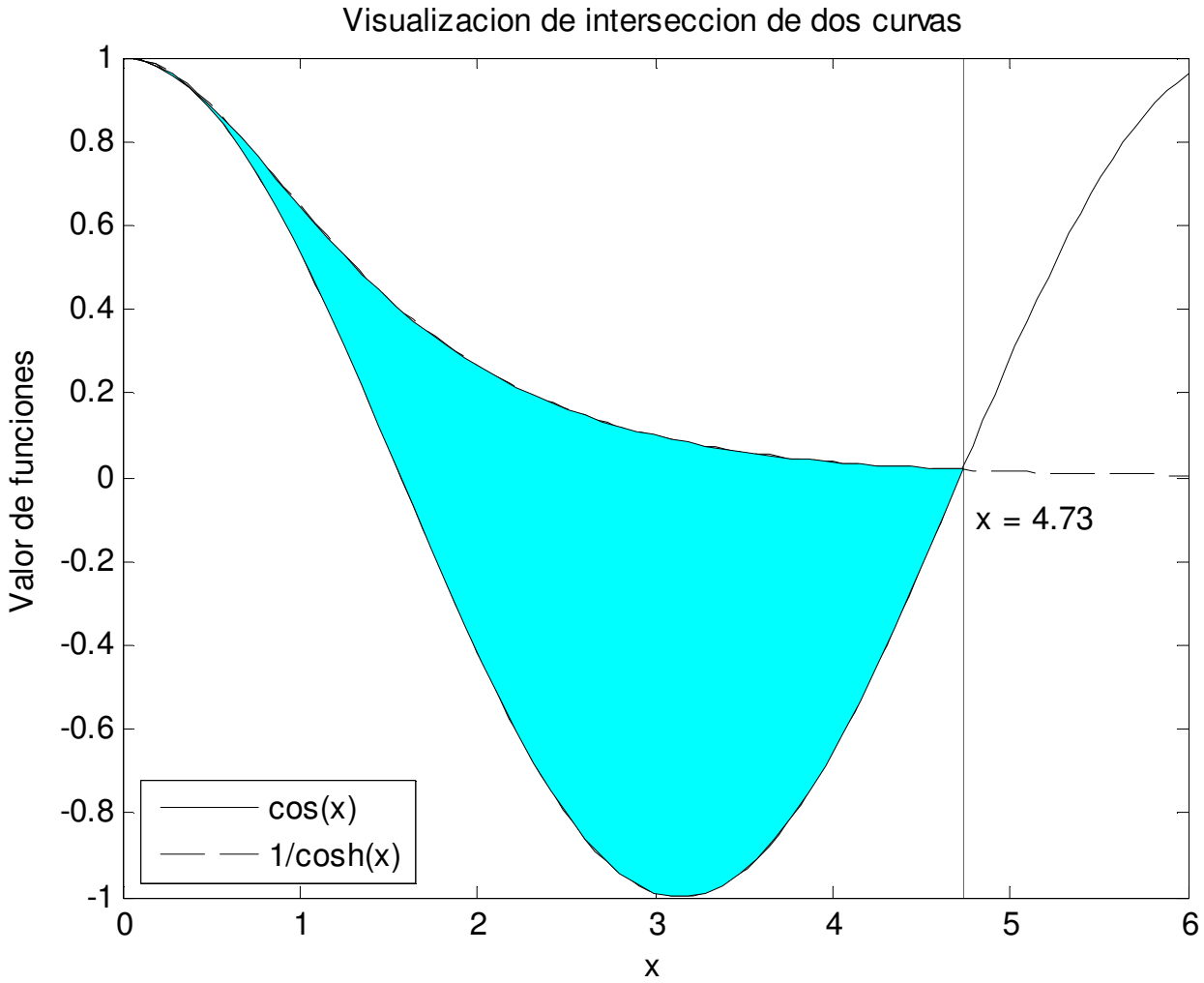
Ejemplo: relleno de regiones

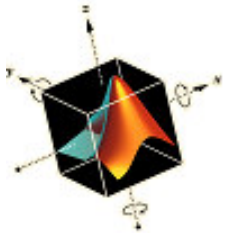
- Modificación del script previo para rellenar el área de las dos curvas en el rango $0 \leq x \leq 4.73$ con color cyan

```
x = linspace(0,6, 100);
plot(x, cos(x), 'k-', x, 1./cosh(x), 'k--', [4.73, 4.73], [-1, 1], 'k')
xlabel('x')
ylabel('Valor de funciones')
title('Visualizacion de interseccion de dos curvas')
text(4.8, -0.1, 'x = 4.73')
legend('cos(x) ', '1/cosh(x) ', 'Location', 'Southwest')
xn = linspace(0, 4.73, 50);
hold on
fill([xn, fliplr(xn)], [1./cosh(xn), fliplr(cos(xn))], 'c');
```



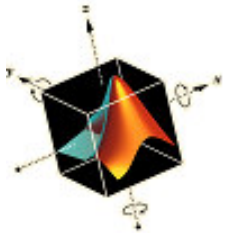
Ejemplo: relleno de regiones





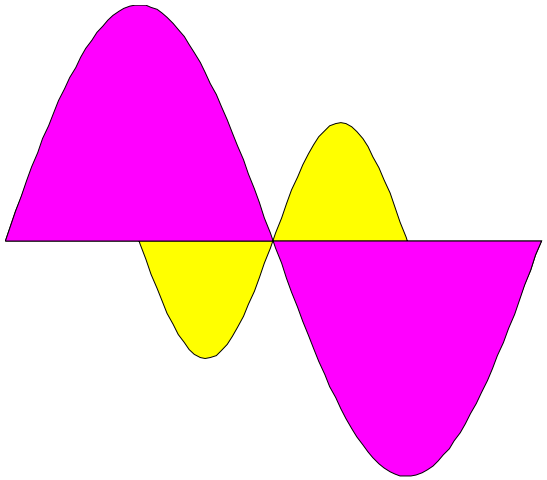
Relleno de regiones: transparencia

- El uso de fill para dibujar dos o más areas solapadas es dependiente en el orden de fill en el script
- Adicionalmente `fill` tiene un ajuste de transparencia que permite que dos areas solapadas se muestren según una cantidad que varía de 0 (invisible) a 1 (opaco)
- Para obtener transparencia se requiere dos keywords
  - *FaceVertexAlphaData*, número entre 0 y 1
  - *FaceAlpha*, que se debe asignar a 'flat' cuando *FaceVertexAlphaData* es igual a un número simple

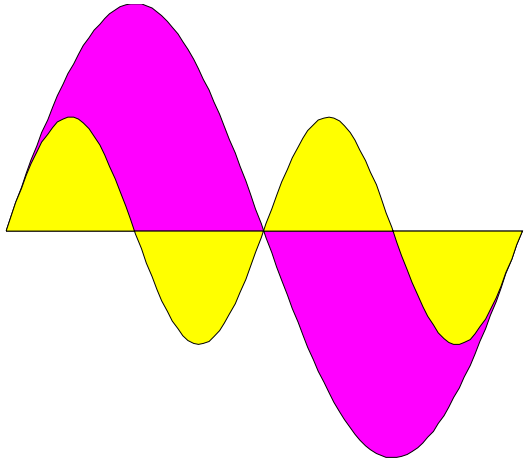


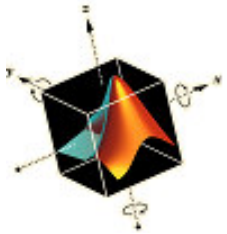
Ejemplo: orden de relleno

```
t = linspace(0, 2*pi);
fill(t, 0.5*sin(2*t), 'y')
hold on
fill(t, sin(t), 'm')
axis off
```



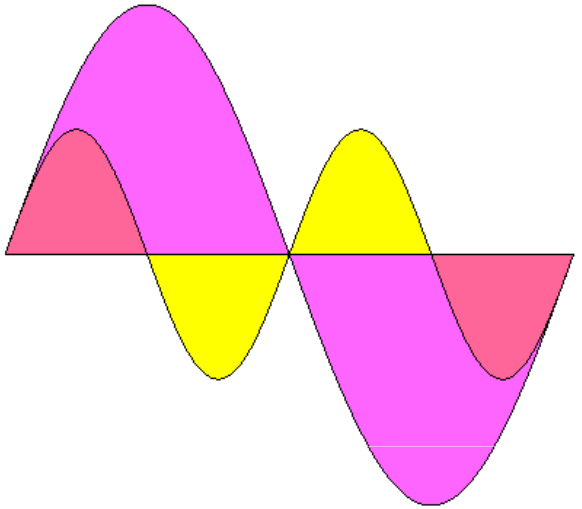
```
t = linspace(0, 2*pi);
fill(t, sin(t), 'm')
hold on
fill(t, 0.5*sin(2*t), 'y')
axis off
```





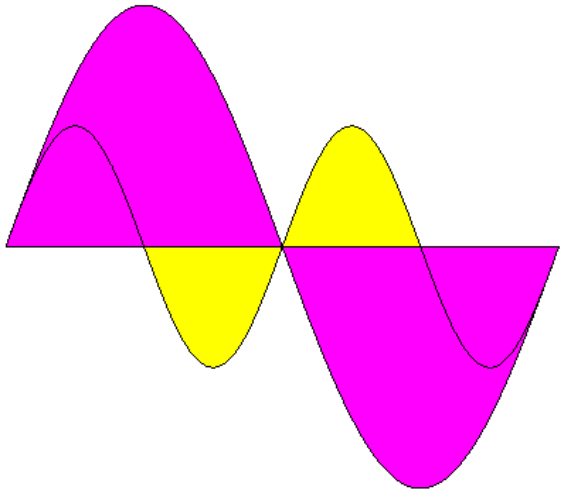
Ejemplo: transparencia

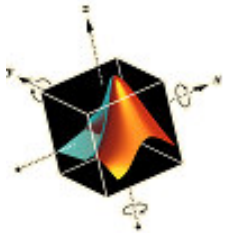
```
t = linspace(0, 2*pi);
fill(t, 0.5*sin(2*t), 'y')
hold on
fill(t, sin(t), ...
  'm', 'FaceVertexAlphaData', 0.6, ...
  'FaceAlpha', 'Flat')
axis off
```



```
t = linspace(0, 2*pi);
fill(t, 0.5*sin(2*t), 'y')
hold on
fill(t, sin(t), ...
  'm', 'FaceVertexAlphaData', 1, ...
  'FaceAlpha', 'Flat')
axis off
```

Opaco

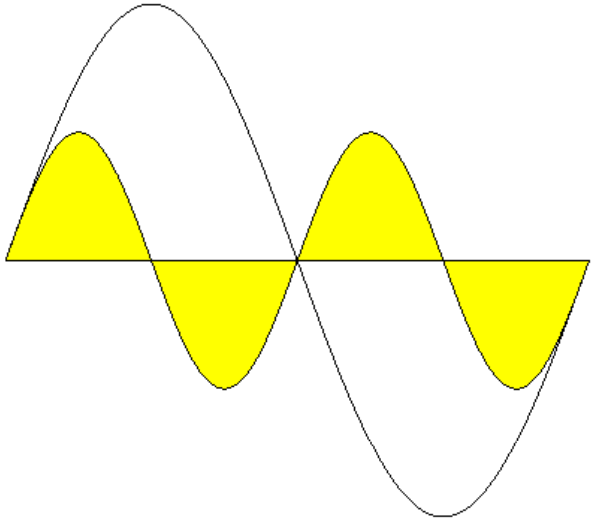


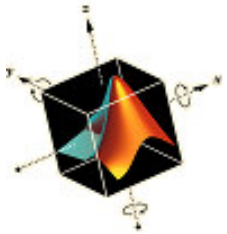


Ejemplo: transparencia

```
t = linspace(0, 2*pi);
fill(t, 0.5*sin(2*t), 'y')
hold on
fill(t, sin(t), …
  'm', 'FaceVertexAlphaData', 0.0, …
  'FaceAlpha', 'Flat')
axis off
```

Invisible





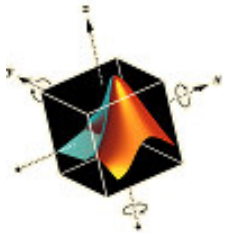
Control del contorno: patch

- `patch` permite mayor control sobre los atributos de las aristas de contorno especificado por

  `patch`(x, y, c)

  donde los argumentos son iguales que `fill`

- Esta función también puede ajustar su propiedad de transparencia para parches solapados de la misma forma que `fill`



Ejemplo: control del contorno

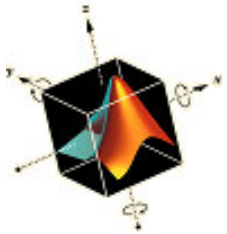
- Se dibujan dos cuadrados que se generan con la función:

```
function [x y] = ptc(a, b, e)
 x = a + [0 0 e e 0];
 y = b + [0 e e 0 0];
```

donde

– *a* y *b* son las distancias desde el origen a lo largo de los ejes *x* e *y*, respectivamente

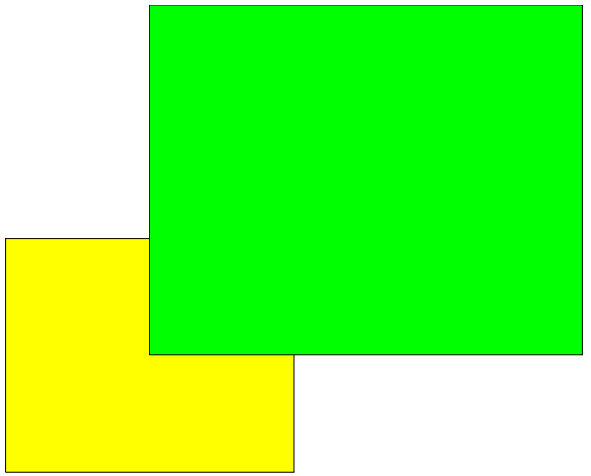
– *e* es la longitud de las aristas del cuadrado



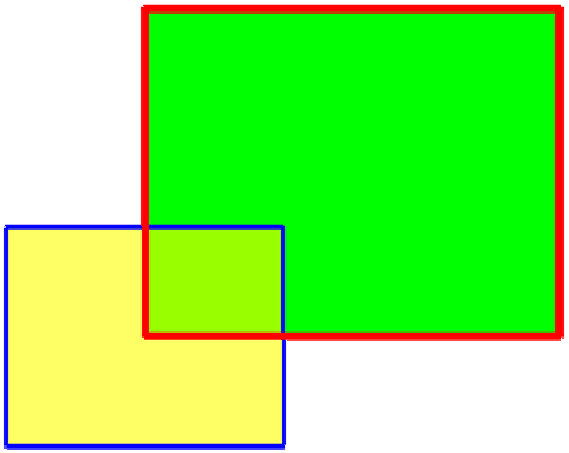
Ejemplo: control del contorno

```
[x1, y1] = ptc(0, 0, 1);
[x2, y2] = ptc(0.5, 0.5, 1.5);
patch(x1, y1, 'y')
patch(x2, y2, 'g')
axis off
```

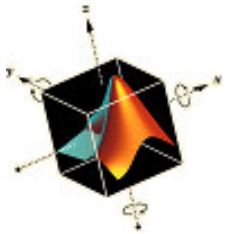
```
[x1, y1] = ptc(0, 0, 1);
[x2, y2] = ptc(0.5, 0.5, 1.5);
patch(x1, y1, 'y', 'EdgeColor', 'b',...
    'LineWidth', 2.5, …
    'FaceVertexAlphaData', 0.6, …
    'FaceAlpha', 'Flat')
patch(x2, y2, 'g', 'EdgeColor', 'r', …
    'LineWidth', 3.5)
axis off
```



Sin transparencia

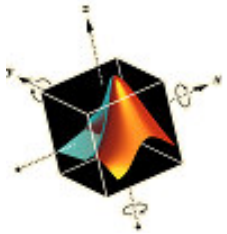


Transparencia y atributos de arista alterados



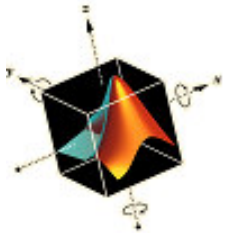
Letras griegas, símbolos matemáticos, subíndices y superíndices

- Matlab proporciona la capacidad de anotar un gráfico con letas griegas, superíndices y subíndices, y símbolos matemáticos
- Tales anotaciones de pueden hacer dentro de `xlabel`, `ylabel`, `text`, `legend`, and `title`
- Las instrucciones de formato siguen el lenguaje LaTeX
- Todas los símbolos requieren estar entre apóstrofes



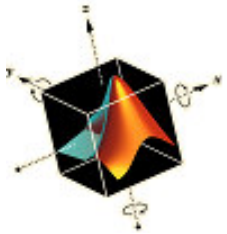
Letras griegas y símbolos matemáticos

Lower case				Upper case		Mathematical			
Symbol	Syntax	Symbol	Syntax	Symbol	Syntax	Symbol	Syntax	Symbol	Syntax
α	\alpha	ν	\nu	Γ	\Gamma	$\leq$	\leq	°	\circ
β	\beta	ξ	\xi	Δ	\Delta	$\geq$	\geq	<<	\ll
γ	\gamma	o	o	Θ	\Theta	$\neq$	\neq	>>	\gg
δ	\delta	π	\pi	Λ	\Lambda	$\pm$	\pm	'	\prime
ε	\epsilon	ρ	\rho	Ξ	\Xi	$\times$	\times	$\Leftarrow$	\Leftarrow
ζ	\zeta	σ	\sigma	Π	\Pi	∞	\infty	$\angle$	\angle
η	\eta	τ	\tau	Σ	\Sigma	$\sum$	\sum	$\surd$	\surd
θ	\theta	υ	\upsilon	Υ	\Upsilon	$\int$	\int	#	\#
ι	\iota	ϕ	\phi	Φ	\Phi	$\div$	\div	$	\$
κ	\kappa	χ	\chi	Ψ	\Psi	~	\sim	%	\%
λ	\lambda	ψ	\psi	Ω	\Omega	$\leftarrow$	\leftarrow	&	\&
μ	\mu	ω	\omega			$\uparrow$	\uparrow	{	\{



Letras griegas, símbolos matemáticos, subíndices y superíndices

- Subíndices se crean con guión bajo (_)
- Superíndices con el operador exponenciación (^)
- Las letras griegas se obtienen anteponiendo a la letra griega una barra invertida (\)
  - La letra mayúscula griega se obtiene poniendo en mayúscula la primera letra
- Cuando se agrupan símbolos se colocan entre llaves ({}). Negrita \bf. Italica \it. Retorno al modo normal \rm
- Lo indicado no funciona con `disp` o `fprint`

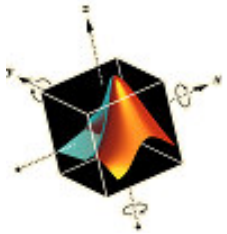


Ejemplo: Letras griegas, símbolos matemáticos, subíndices y superíndices

- Se calcula y grafica la función $g_2 = \cos(4\pi\Omega_1)e^{-(1+\Omega_1^{\beta})}$

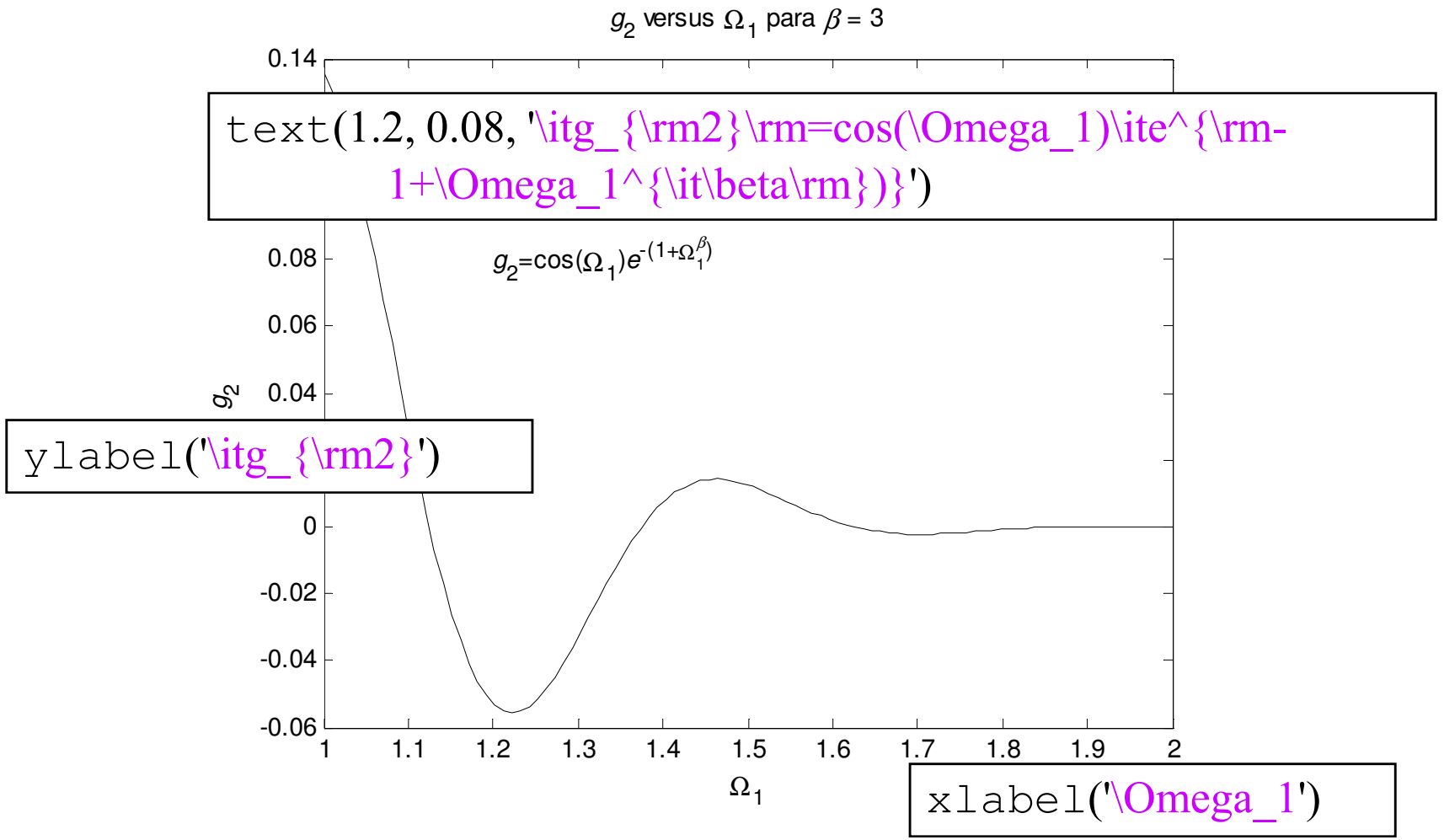
  para $\beta = 3$ y $1 \leq x \leq 2$ y se etiqueta adecuadamente

```
Om1 = linspace(1, 2);  beta = 3;
plot(Om1, cos(4*pi*Om1).*exp(-(1+Om1.^beta)), 'k')
title('\itg_{\rm2} \rmversus \Omega_1 para
        \it\beta \rm= 3')
ylabel('\itg_{\rm2}')
xlabel('\Omega_1')
text(1.2, 0.08, '\itg_{\rm2}\rm=cos(\Omega_1)\ite^{\rm-
                (1+\Omega_1^{\it\beta\rm})}')
```



Ejemplo: Letras griegas, símbolos matemáticos, subíndices y superíndices

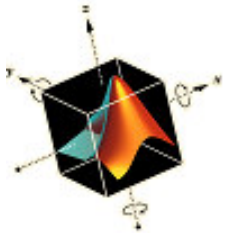
```
title('\itg_{\rm2} \rmversus \Omega_1 para \it\beta \rm= 3')
```




```
text(1.2, 0.08, '\itg_{\rm2}\rm=cos(\Omega_1)\ite^{\rm-1+\Omega_1^{\it\beta\rm})}')
```

```
ylabel('\itg_{\rm2}')
```

```
xlabel('\Omega_1')
```



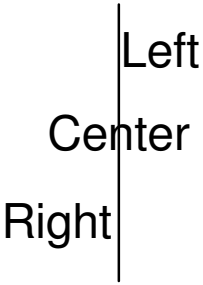
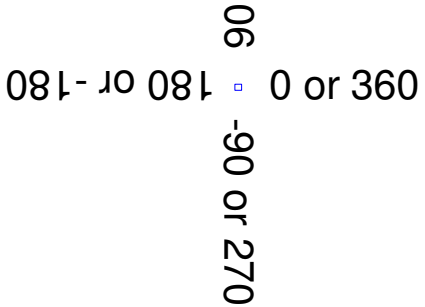
Modificación de atributos de ejes, curvas y texto

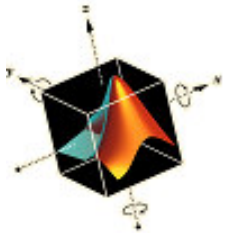
- Es posible modificar todas las características de los elementos que componen un gráfico
- Los atributos para `xlabel`, `ylabel`, `title`, y `text` se cambian según la sintaxis:

```
xlabel(s, 'Keyword', ValorKeyWord, ...)
ylabel(s, 'Keyword', ValorKeyWord, ...)
title(s, 'Keyword', ValorKeyWord, ...)
text(x, y, s, 'Keyword', ValorKeyWord, ...)
```

donde: *s* es el string a mostrar. 'KeyWord' es un string que contiene el key word. *ValorKeyWord* es el valor del key word (string o número). *x*, *y* son las coordenadas de pos. del texto

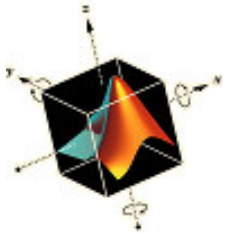
Keywords y atributos para la posición del texto

Keyword	Keyword value	Example
'HorizontalAlignment'	'Left' 'Center' 'Right'	
'VerticalAlignment'	'Top' 'Middle' 'Bottom'	
'Rotation'	0 to 360° or –180° to +180°	



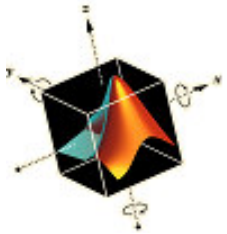
Keywords y atributos para texto

Key word	Key word value
'Linewidth'	number > 0 (default: 0.5)
'FontSize'	number > 0 (default: 10)
'FontName'	'Courier' 'Helvetica' (default) 'Times' (Times roman)
'Color'	'Letter
'FontWeight'	'Normal' (default) 'Bold'



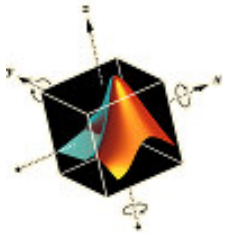
Keywords y atributos del handle legend

Handle name	Keyword	Keyword value	Attribute affected
a(1)	'LineWidth'	Number > 0 (default: 0.5)	Thickness of legend box edges
a(1)	'Color'	'Letter	Background color of legend box
b(1), b(2)	'FontSize'	Number > 0 (default: 10)	Font size of legend text
b(1), b(2)	'FontName'	'Courier' 'Helvetica' (default) 'Times' (Times roman)	Font type of legend text
b(1), b(2)	'Color'	'Letter	Color of legend text



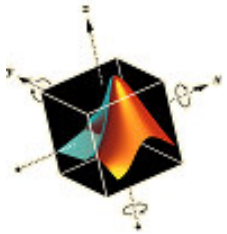
Keywords y atributos para líneas

Key word	Keyword value
'Linewidth'	Number > 0 (default: 0.5)
'Color'	'Letter



Modificación de atributos de ejes, curvas y texto

- Para asignar los atributos de curvas creadas por `plot` y el texto que aparece en `legend` se requiere los function handles
- Para modificar el ancho de los ejes y las marcas se usan `set` y `gca` que asignan el handle para el eje en curso
- La función `set` asigna una propiedad al handle
- Para obtener los handles para `legend` and `plot` se usa: hdl = `plot`(...); [lh, lo] = `legend`(...); donde hdl y lo(1) son los handle de curva y texto



Modificación de atributos de ejes, curvas y texto

- La función `set` es

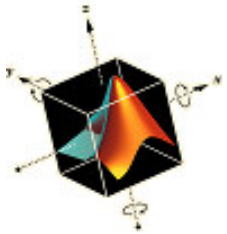
  `set`(hdl, 'Keyword', KeyWordValue, …)

  donde:

  hdl es el handle.

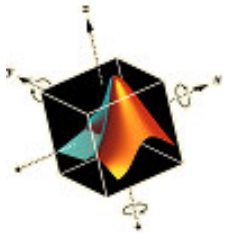
  'KeyWord' es un string conteniendo el key word.

  KeyWordValue es el string o valor numérico que corresponde al key word.



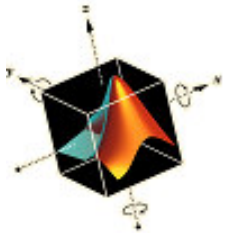
Ejemplo: modificación de atributos de ejes, curvas y texto

- Se requiere modificar
  - color de fondo del recuadro de la leyenda a amarillo
  - ancho del recuadro a 2 puntos
  - tamaño del texto a 14 puntos
  - texto de la curva sólida en azul y que para la curva en trozos en rojo
  - variable independiente *x* en itálica

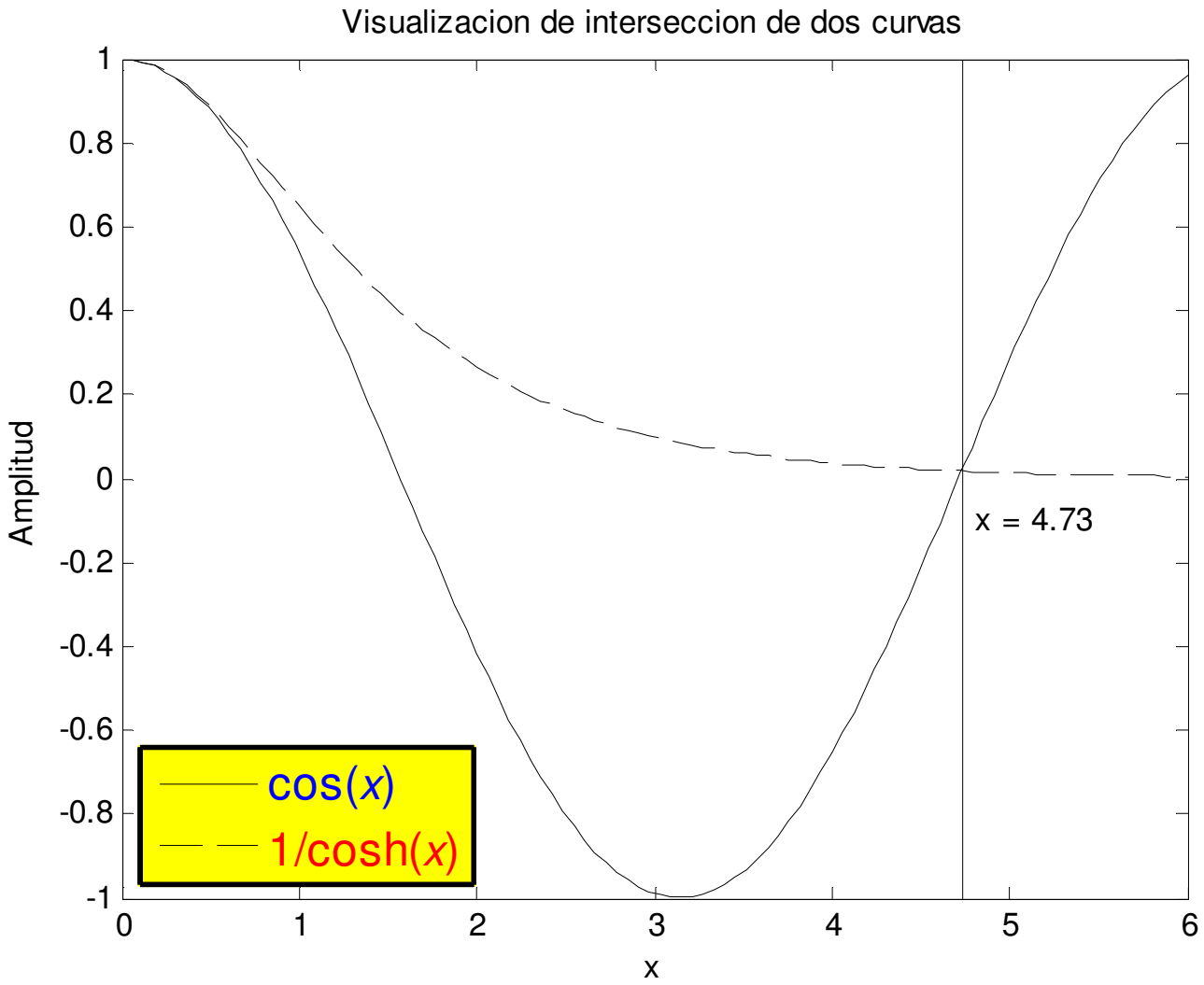


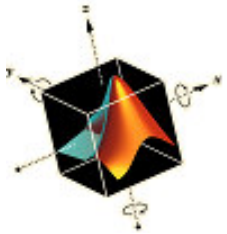
Ejemplo: modificación de atributos de ejes, curvas y texto

```
x = linspace(0, 6, 100);
plot(x, cos(x), 'k-', x, 1./cosh(x), 'k--', [4.73, 4.73], [-1, 1], 'k')
xlabel('x')
ylabel('Amplitud')
title('Visualizacion de interseccion de dos curvas')
text(4.8, -0.1, 'x = 4.73')
[a, b] = legend('cos(\itx\rm)', '1/cosh(\itx\rm)', 'Location',
        'SouthWest');
set(a(1), 'LineWidth', 2, 'Color', 'y')
set(b(1), 'fontsize', 14, 'Color', 'b')
set(b(2), 'fontsize', 14, 'Color', 'r')
```



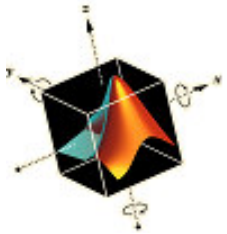
Ejemplo: modificación de atributos de ejes, curvas y texto





Ejemplo: modificación de atributos de ejes, curvas y texto

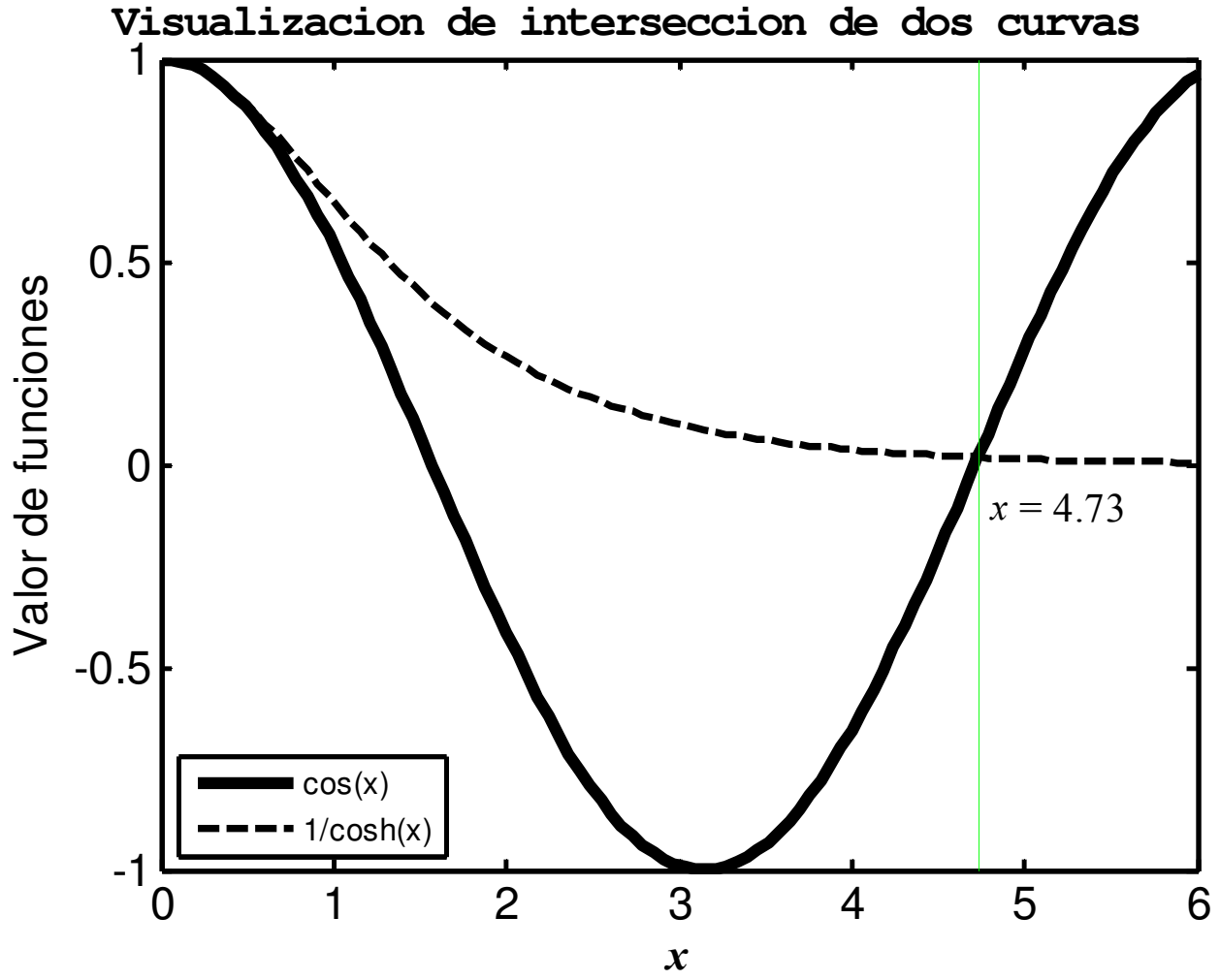
- Se requiere modificar adicionalmente
  - Título: 14 puntos courier, negrita
  - Etiqueta eje-x: 14 puntos roman, italic y negrita
  - Etiqueta eje-y: 14 puntos helvetica
  - Posición de texto: 12 ptos standard notación matemática
  - Líneas ejes: ancho 1.5 ptos
  - Texto de ejes: 14 pts helvetica
  - Curva para cos(x): 4 pt ancho línea
  - Curva para cosh(x): 2.5 pt ancho línea
  - Línea vertical en x = 4.73: 0.25 ancho línea, verde

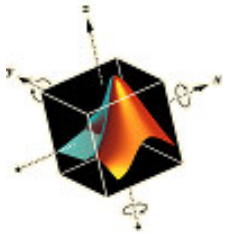


Ejemplo: modificación de atributos de ejes, curvas y texto

```
x = linspace(0, 6,100);
hc = plot(x, cos(x), 'k-');
hold on
hch = plot(x, 1./cosh(x), 'k--');
hsl = plot( [4.73, 4.73], [-1, 1], 'k');
[a, b] = legend('cos(x) ', '1/cosh(x) ', 'Location', 'SouthWest');
xlabel('\it\bfx', 'FontSize', 14, 'FontName', 'Times')
ylabel('Valor de funciones', 'FontSize', 14)
title('\bfVisualizacion de interseccion de dos curvas', 'FontName',
        'Courier', 'FontSize', 14)
text(4.8, -0.1, '\itx \rm= 4.73','FontName', 'Times', 'FontSize', 12)
set(hc, 'LineWidth', 4)
set(hch, 'LineWidth', 2.5)
set(hsl, 'LineWidth', 0.25, 'Color', 'g')
set(gca, 'FontSize', 14, 'LineWidth', 1.5)
set(b(1), 'FontSize', 10)
```

Ejemplo: modificación de atributos de ejes, curvas y texto





Inserción de un gráfico dentro de otro

- Para insertar una función dentro de una figura se usa `axes`('Position', [left, bottom, width, height])

  donde:

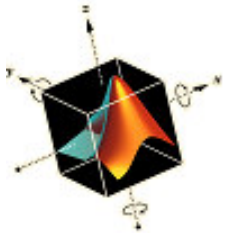
  Position - keyword

  left - coordenada-x de la figura de origen (0<left<1)

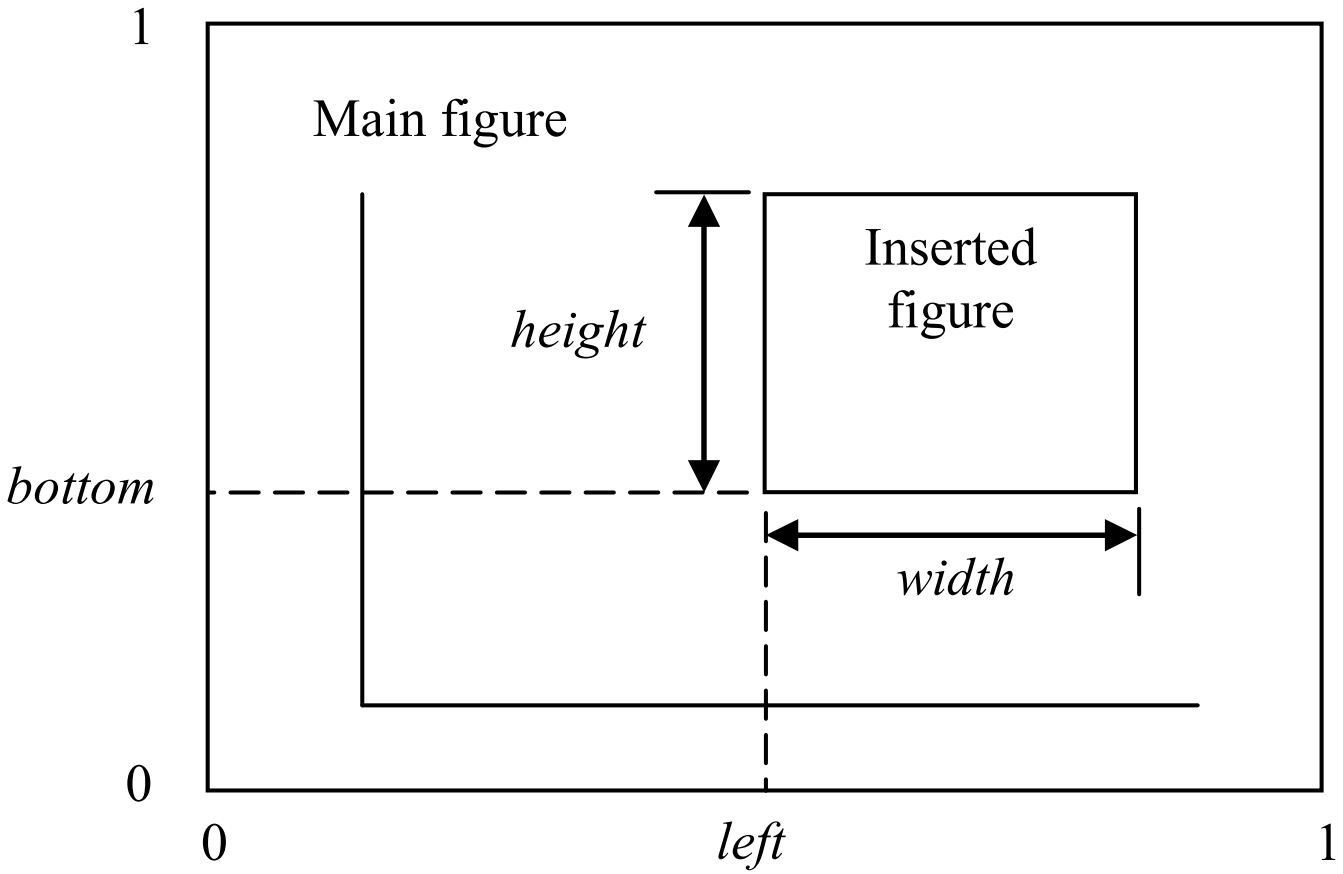
  bottom – coordenada-y de la figura de origen (0<bottom<1)

  width – ancho de la figura (0<width<1)

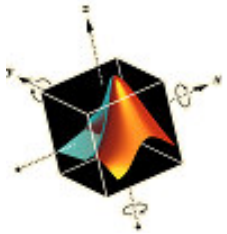
  height – altura de la figura (0<height<1)



Inserción de un gráfico dentro de otro



`axes('Position', [left, bottom, width, height])`

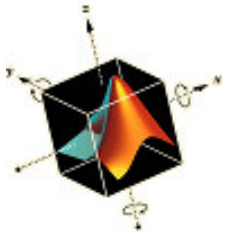


Ejemplo: inserción de un gráfico dentro de otro

- Se muestra como figura principal la respuesta del desplazamiento $x(\tau)$ de un sistema de un grado de libertad sujeto a un pulso periódico de período T y duración de pulso t_d y como figura insertada la amplitud de la función de la repuesta en frecuencia $H(\Omega)$ del sistema. Considerando que la frecuencia natural del sistema es ω_n, las relaciones son

$$x(\tau) = \alpha\left[1 + 2\sum_{k=1}^{\infty} H(\Omega_k)\left|\frac{\sin(k\pi\alpha)}{k\pi\alpha}\right|\sin\left(\Omega_k\tau - \theta(\Omega_k) + \psi_k\right)\right]$$

donde $\alpha = t_d/T < 1$, $\Omega_k = k\Omega_o$, $\Omega_o = \omega_o/\omega_n$, $\omega_o = 2\pi/T$



Ejemplo: inserción de un gráfico dentro de otro

adicionalmente,

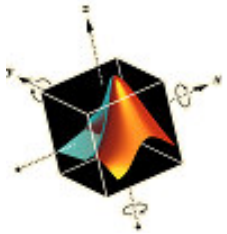
$$H(\Omega_k) = \frac{1}{\sqrt{\left(1-\Omega_k^2\right)^2 + \left(2\zeta\Omega_k\right)^2}}$$

$$\theta(\Omega_k) = \tan^{-1}\frac{2\zeta\Omega_k}{1-\Omega_k^2}$$

$$\psi_k = \tan^{-1}\frac{\sin(k\pi\alpha)/k\pi\alpha}{0}$$

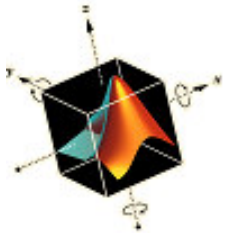
y $\zeta < 1$ es el factor de amortiguamiento

Si se toma 200 términos de las series, y se asume que $\zeta = 0.1$, $-50 \le \tau \le 120$, y $\alpha = 0.4$, el script es

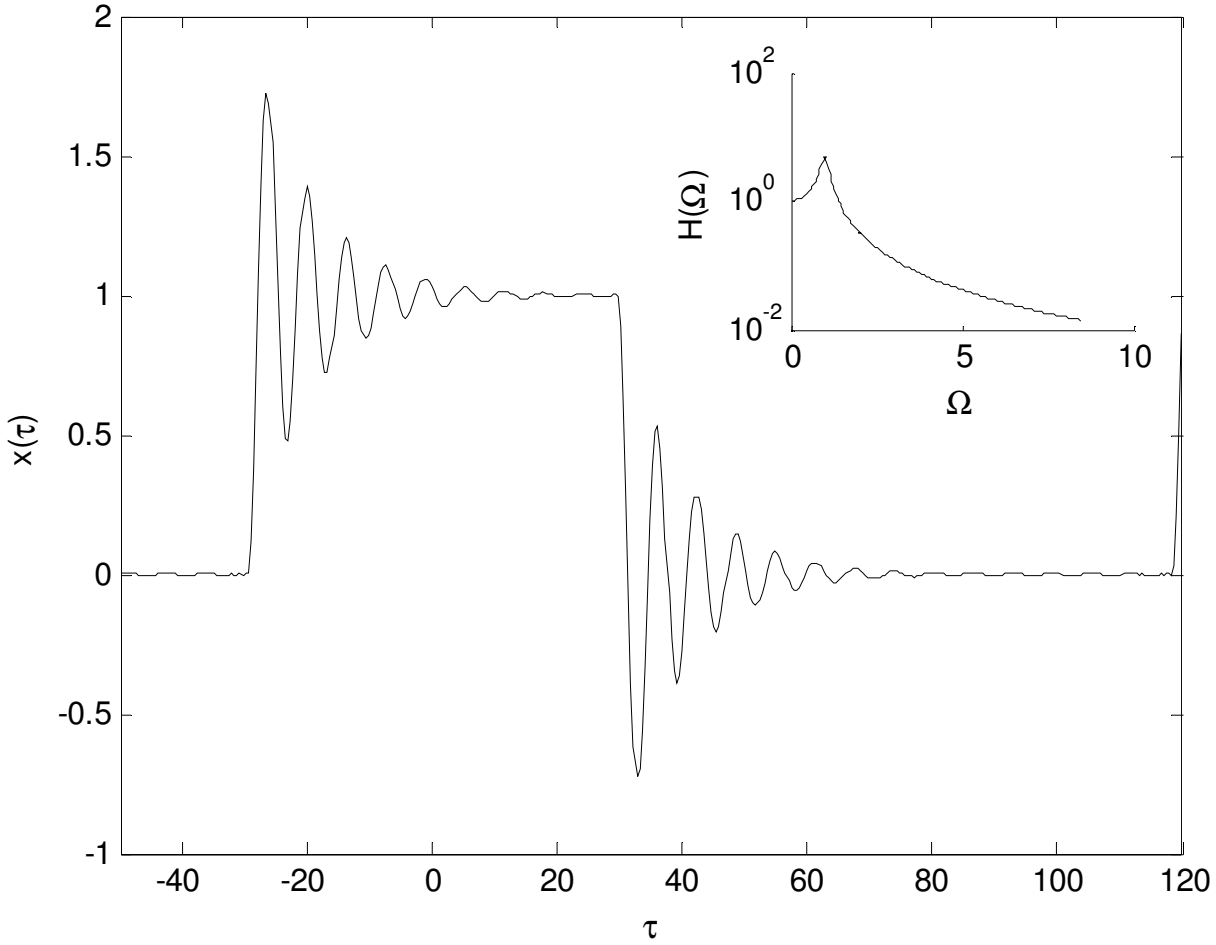


Ejemplo: inserción de un gráfico dentro de otro

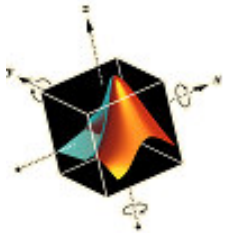
```
k = 1:200;   alph = 0.4;   xi = 0.1;  Omo = 0.03*sqrt(2);  N = 400;
HOm = inline('1./sqrt((1-(Om*k).^2).^2+(2*xi*Om*k).^2)', 'k', 'Om', 'xi');
tau = linspace(-50, 120, N);
sn = sin(pi*k*alph)./(pi*k*alph);
thn = atan2(2*xi*Omo*k, (1-(Omo*k).^2)); psi = atan2(sn, 0);
cnt = sin(Omo*k'*tau-repmat(thn', 1, N)+repmat(psi', 1, N));
z = alph*(1+2*abs(sn).*HOm(k, Omo, xi)*cnt);
plot(tau, z, 'k-')
a = axis;  a(1) = -50;  a(2) = 120;
axis(a)
xlabel('\tau')
ylabel('x(\tau)')
axes('Position', [0.62, 0.62, 0.25, 0.25])
semilogy(k*Omo, HOm(k, Omo, xi), 'k-')
ylabel('H(\Omega)')
xlabel('\Omega')
box off
```



Ejemplo: inserción de un gráfico dentro de otro



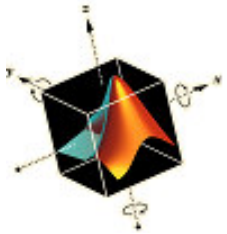

```
axes('Position', [0.62, 0.62, 0.25, 0.25])
axes('Position', [left, bottom, width, height])
```



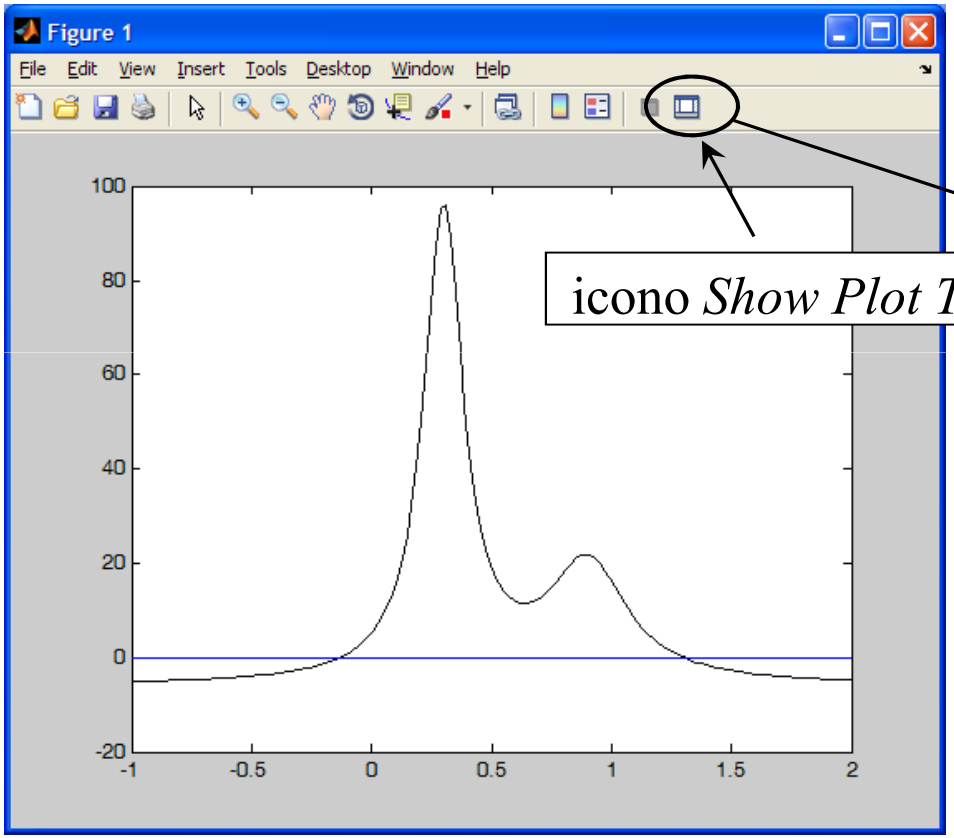
Herramientas de graficación interactivas

- Las modificaciones de los atributos de texto, curvas y ejes se pueden realizar interactivamente en la ventana de la figura seleccionando la operación apropiada en el menú Tools
- Después de realizar los cambios, la figura se puede guardar como un fichero función m
- Como ejemplo de uso de estas herramientas se usa el gráfico producido por

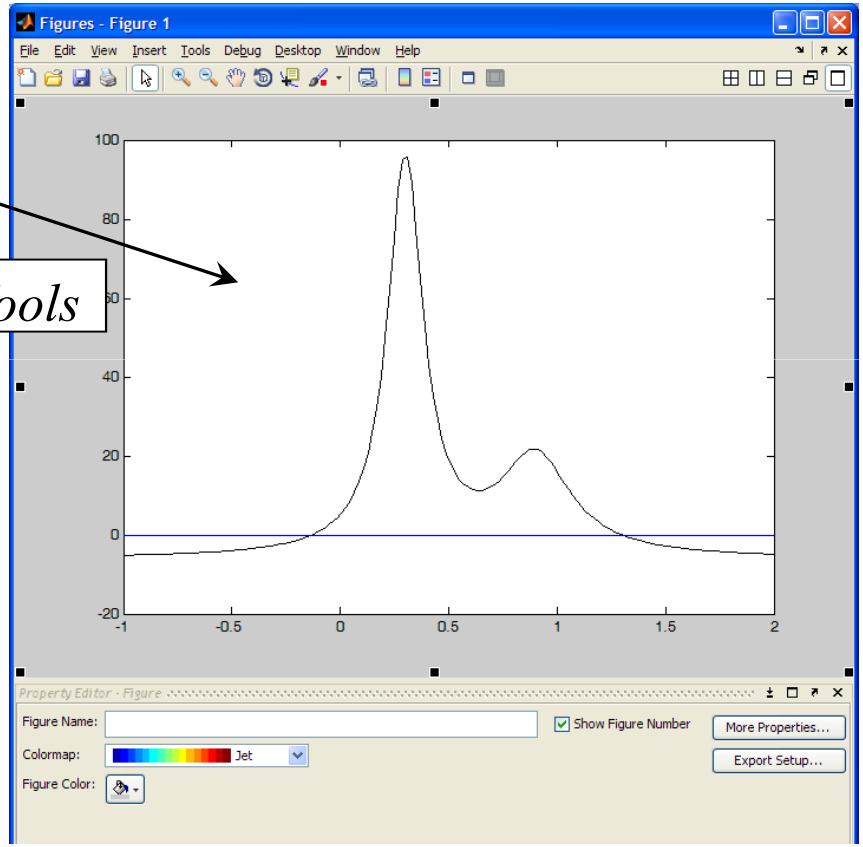
```
x = linspace(-1, 2, 150);
plot(x, humps(x), 'k-', [-1, 2], [0, 0], 'b-')
```

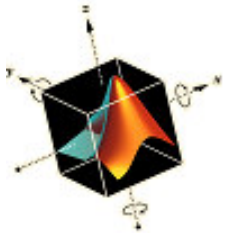


Ejemplo: herramientas de graficación interactivas

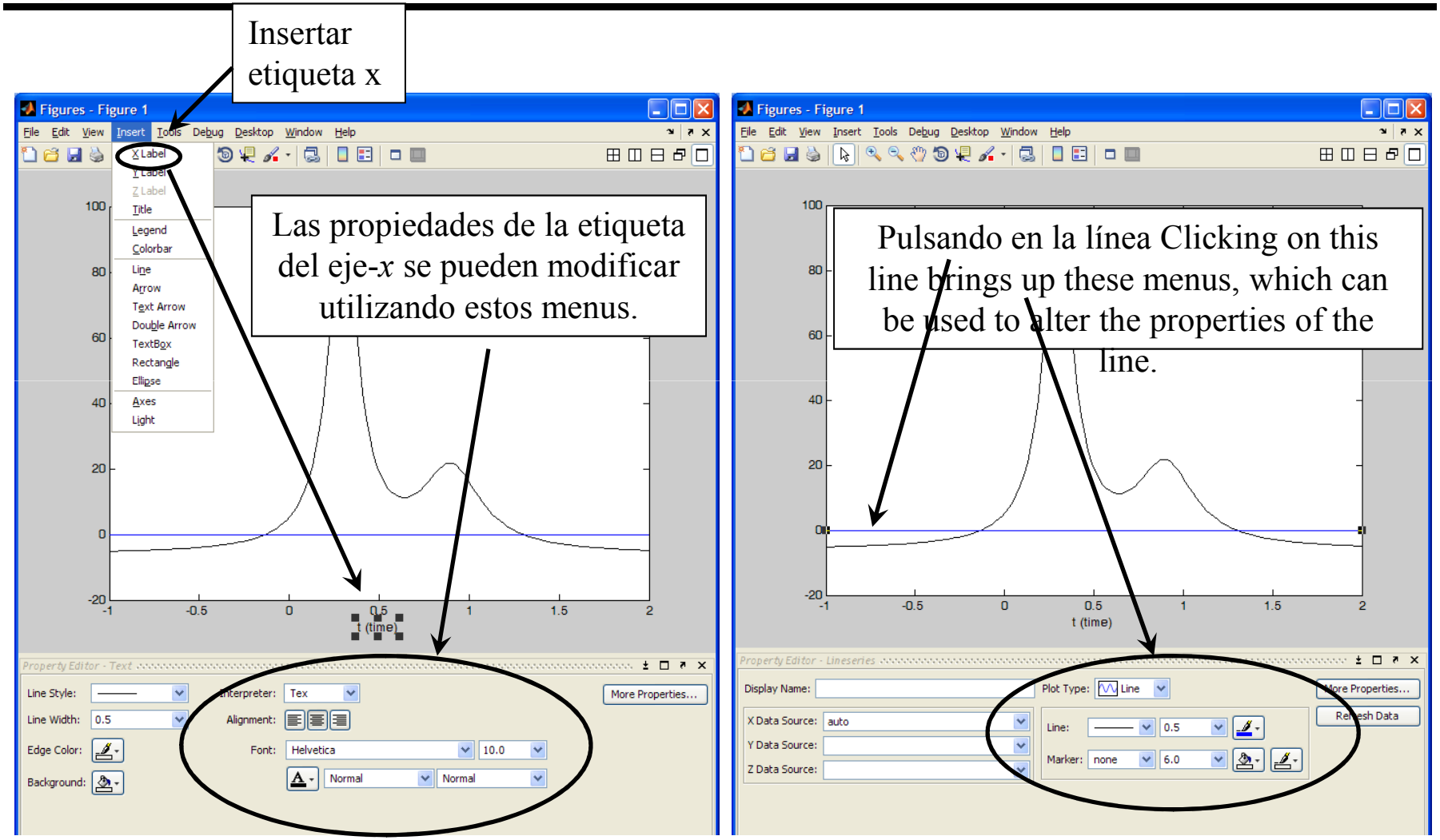


icono *Show Plot Tools*

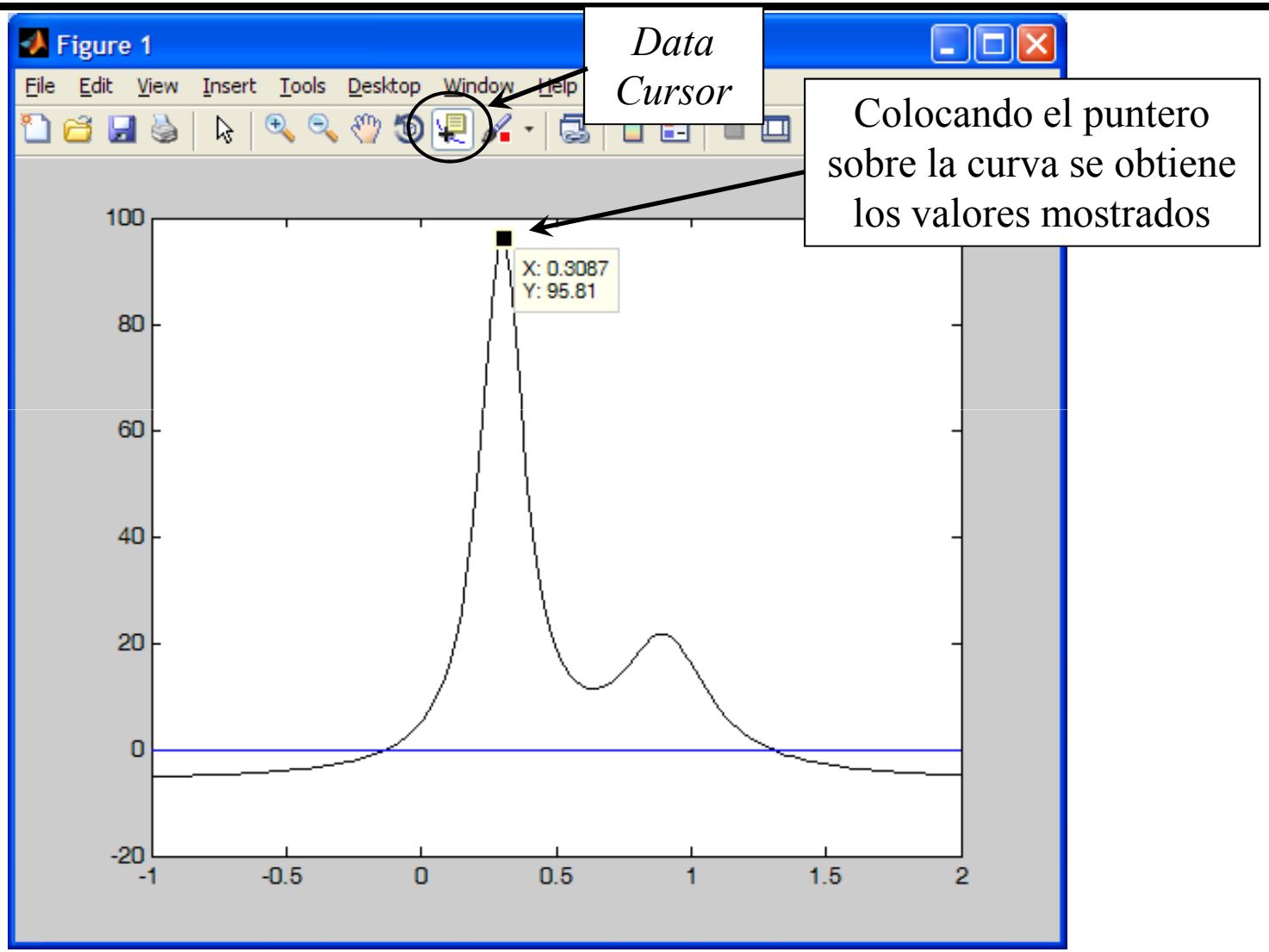




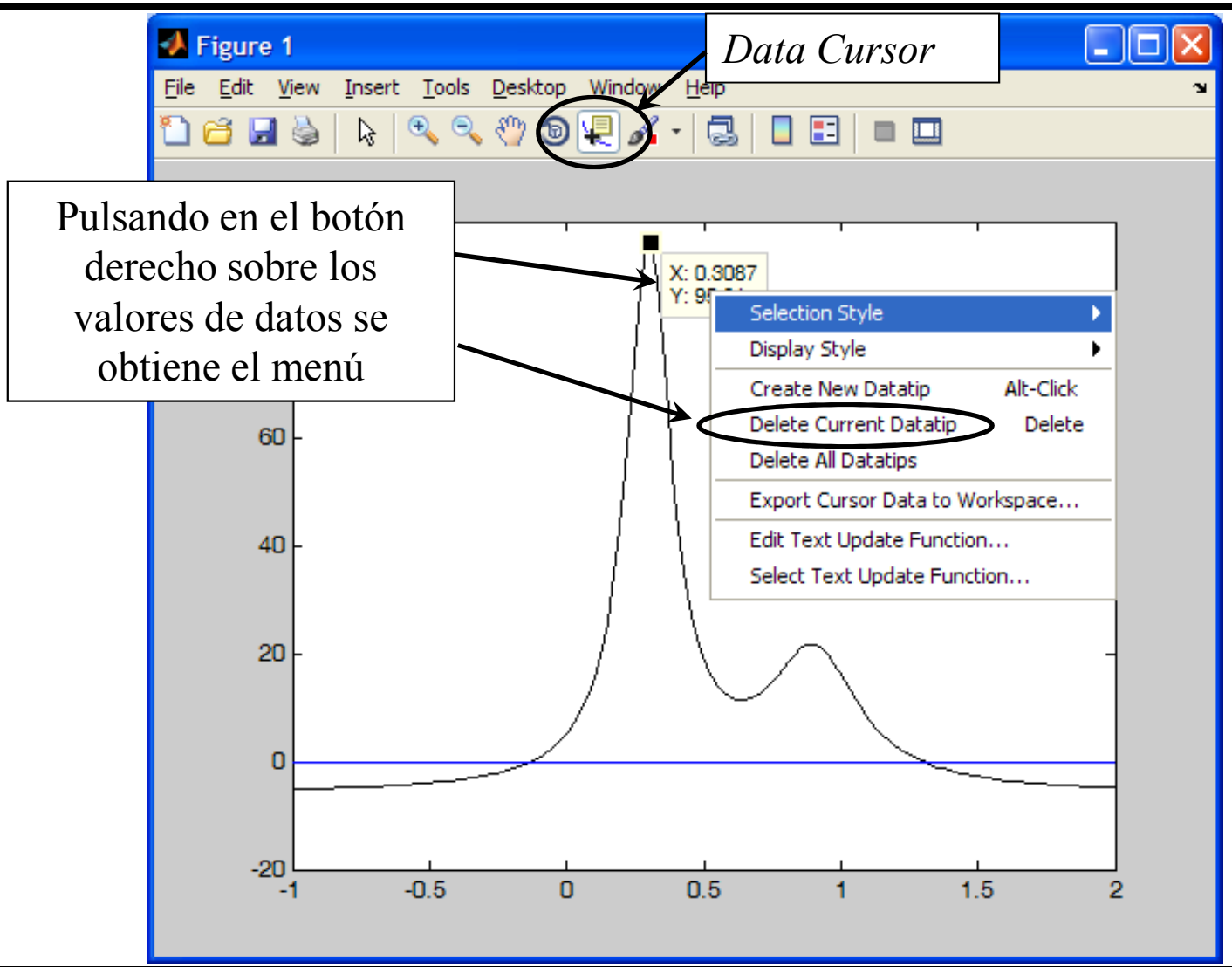
Ejemplo: herramientas de graficación interactivas

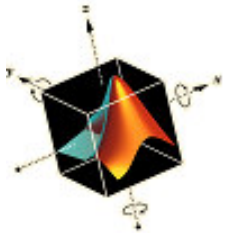


Ejemplo: herramientas de graficación interactivas



Ejemplo: herramientas de graficación interactivas





Animación

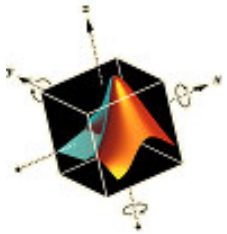
- Las funciones que se usan para crear una película son

  A(k) = `getframe`

  que captura el *k*-ésimo frame de un total de N frames

  `movie`(A, nF, pbs)

  que se usa para reproducir *nF* veces los *N* frames capturados en la matriz *A* por getframe. Esta función reproduce la secuencia a una velocidad *pbs*, que si se omite se usa el valor por defecto 12 frames por segundo



Animación

- Para crear una película en el formato *avi* (audio/video interleaved) a partir de la película creada con `movie`, se usa

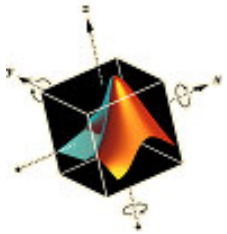
```
movie2avi(A, 'FileName.avi', 'KeyWord', 'KeyWordValue');
```

  La variable *A* es la variable usada en `movie`

- Para crear películas que pueden ser mostrados en Microsoft Power Point, se asigna

  'KeyWord' = 'compression'

  'KeyWordValue' = 'none'



Ejemplo de Animación

- Se requiere crear una animación de un mecanismo de de manivela. La distancia horizontal del movimiento del deslizador como función del ángulo de rotación φ de la manivela es

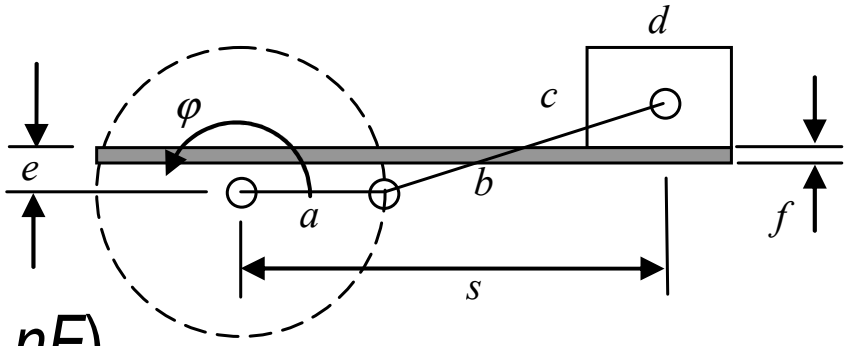
$$s = a\cos\varphi + \sqrt{b^2 - (a\sin\varphi - e)^2}$$

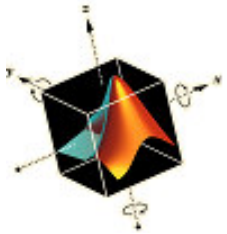
$0 \le \varphi \le 2\pi$ en posiciones igualmente espaciadas $n = 40$

$a = 1 \quad b = 2.5 \quad e = 0.25$

$c = 0.5 \quad d = 1 \quad f = 0.06$

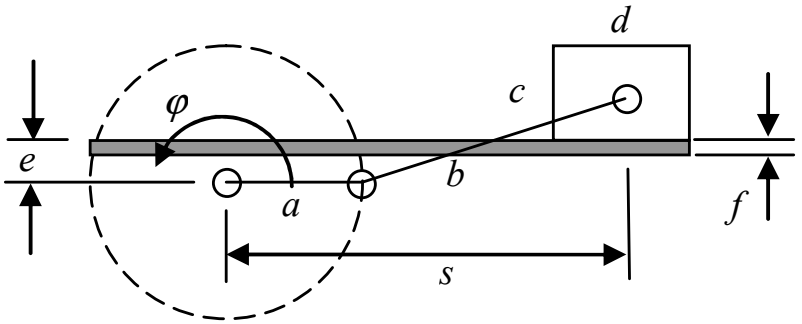
El número de repetición es 5 (= nF)



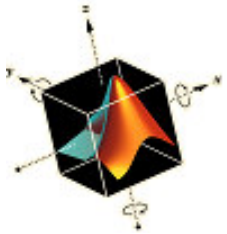


Ejemplo de Animación

```
n = 40;  phi = linspace(0, 2*pi, n);
a = 1;   b = 2.5;   e = 0.25;   nF = 5;
c = 0.5;   d = 1;   f = 0.06;
ax = a*cos(phi);   ay = a*sin(phi);
s = real(ax+sqrt(b^2-(ay-e).^2));
v = [1.1*min(ax), 1.1*(max(s)+d/2) 1.1*min(ay), 1.1*max(ay)];
xgnd = [min(ax), max (s)+d/2, max(s)+d/2, min(ax), min(ax)];
ygnd = [e, e, e-f, e-f, e];
slidery = [e, e+c, e+c, e, e];  % Componente vertical del deslizador
                                es constante
```

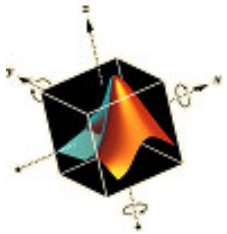


$$s = a\cos\varphi + \sqrt{b^2 - (a\sin\varphi - e)^2}$$

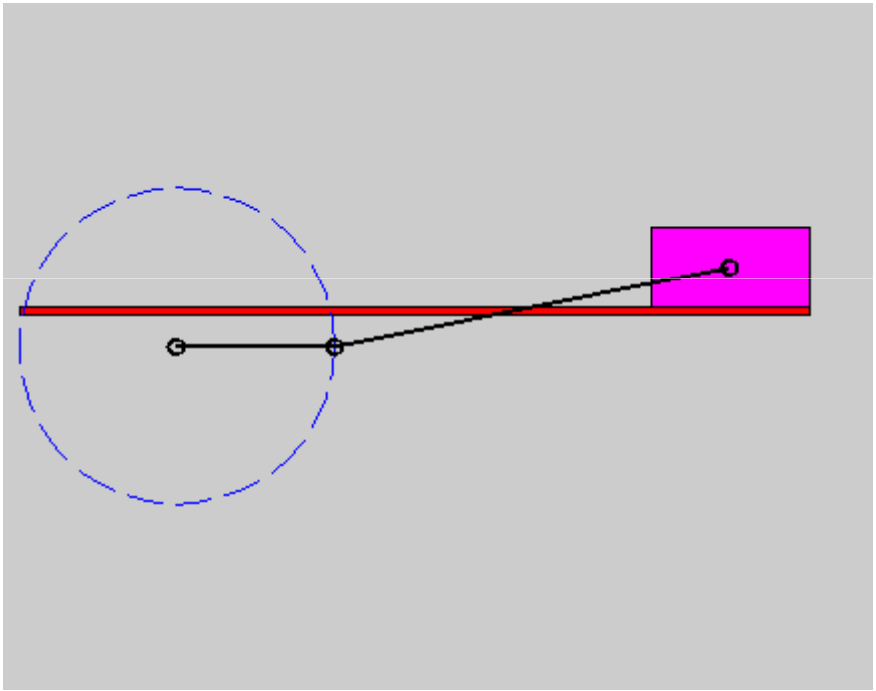


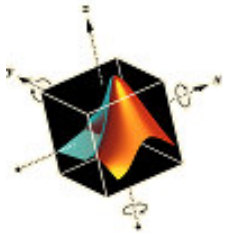
Ejemplo de Animación

```
for k=1:n
   fill(xgnd, ygnd, 'r')  % Barra horizontal
   hold on
   plot(ax, ay, 'b--', 0, 0, 'ko');  % circulo punteado y centro de circulo
   sliderx = [s(k)-d/2, s(k)-d/2, s(k)+d/2, s(k)+d/2, s(k)-d/2];
   fill(sliderx, slidery, 'm');  % posicion  slider
   plot([0 ax(k)], [0;ay(k)], 'ko-', 'LineWidth', 2);
   plot([ax(k), s(k)], [ay(k), e+c/2], 'ko-', 'LineWidth', 2);
   axis(v)
   axis off equal
   ManivelaFrame(k) = getframe;
   hold off
end
movie(ManivelaFrame, nF, 30)
movie2avi(ManivelaFrame, 'Manivela.avi', 'compression', 'none')
```



Ejemplo de Animación





Ejemplo de gráfico en coordenadas polares

- La presión normalizada del sonido a una distancia grande del centro de un pistón circular en una pantalla acústica infinita que vibra a una frecuencia *f* es

$$p(r,\theta) = \left| \frac{J_1(ka\theta)}{ka\theta} \right| \qquad ka^2 << r \quad \text{and} \quad a << r$$

donde

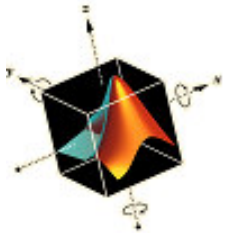
r es la distancia radial desde el centro del pistón

θ es el ángulo de *r* con respecto al plano de la pantalla

k es el número de onda

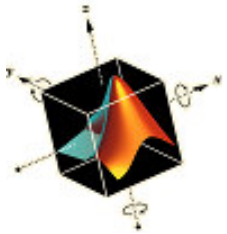
a es el radio del pistón

$J_1(x)$ es la función Bessel del primer tipo de orden 1



Ejemplo de gráfico en coordenadas polares

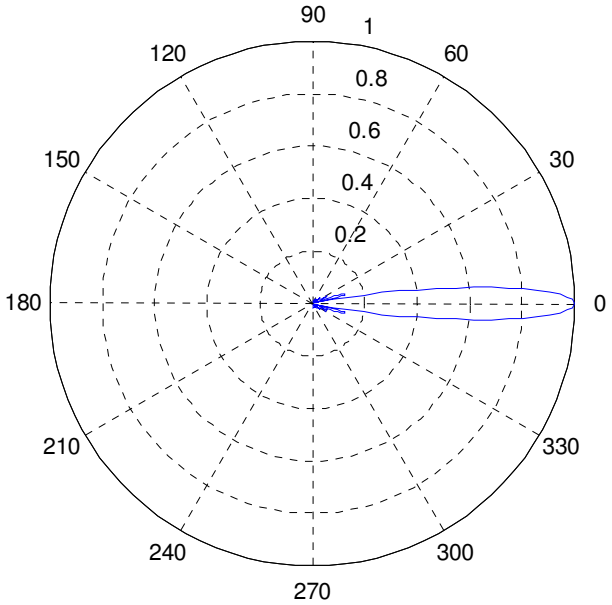
- El número de onda es el recíproco de la longitud de onda del sonido a la frecuencia *f*; por lo que *ka* es adimensional.
- El modelo es un aproximación a la dispersión angular del sonido desde un altavoz
- Se crea un gráfico polar del patrón de radiación normalizada para $ka = 6\pi$ cuando θ está en el rango $-\pi/2 < \theta < \pi/2$.
- Se selecciona esta solución para mostrar el uso de `polar`, que crea un gráfico en coordenadas polares

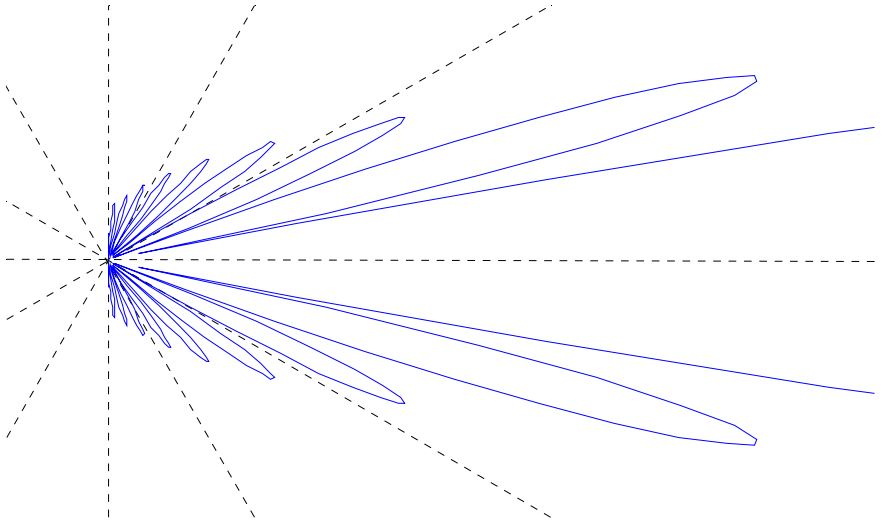


Ejemplo de gráfico en coordenadas polares

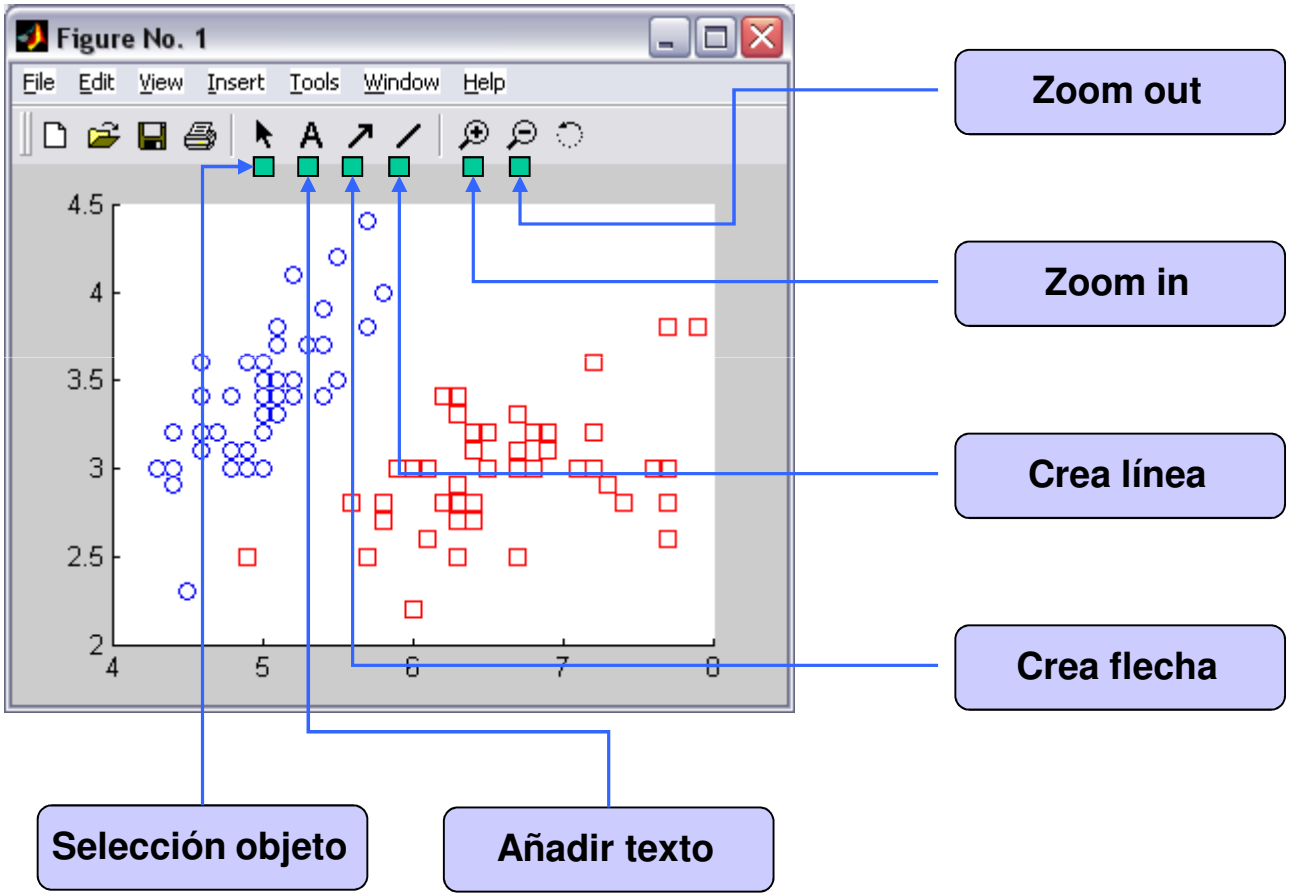
- El script es:

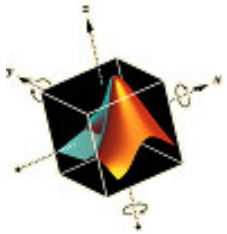
```
theta = linspace(–pi/2, pi/2, 300);
p = abs(besselj(1, 6*pi*theta)./(6*pi*theta));
polar(theta, p/max(p)) % figura de la izquierda
axis([-.02, 0.15, -0.05, 0.05]) % figura de la derecha
```



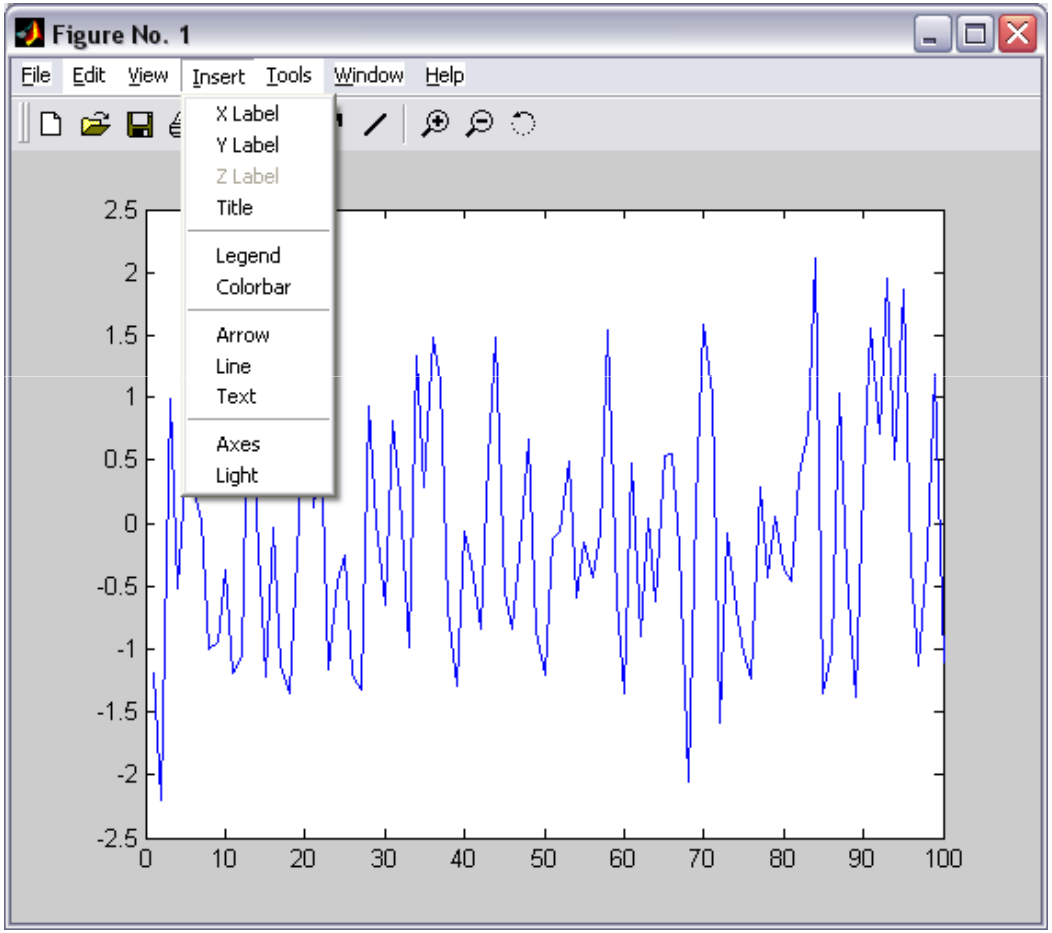


Modificación interactiva de gráficos



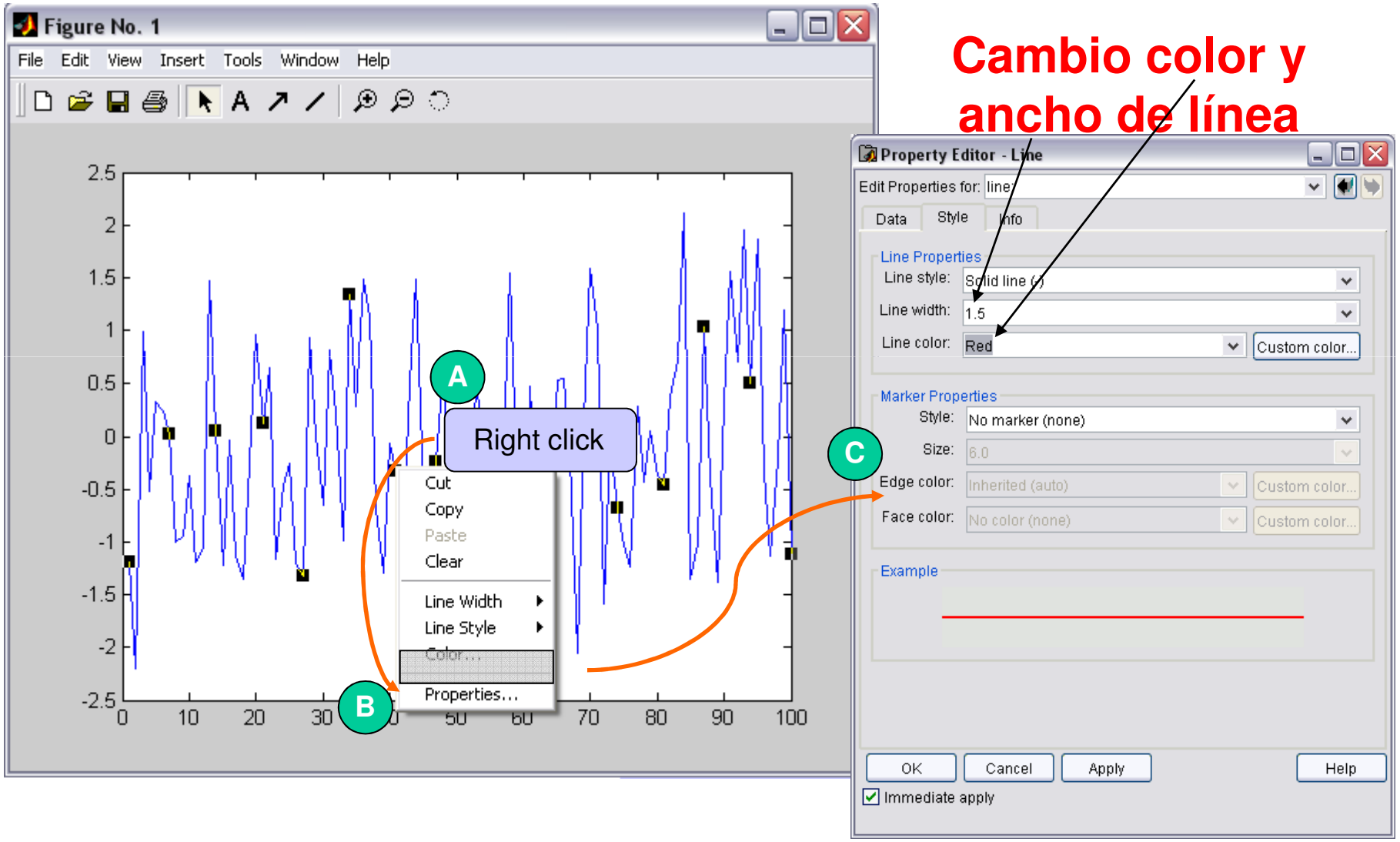


Modificación interactiva de gráficos

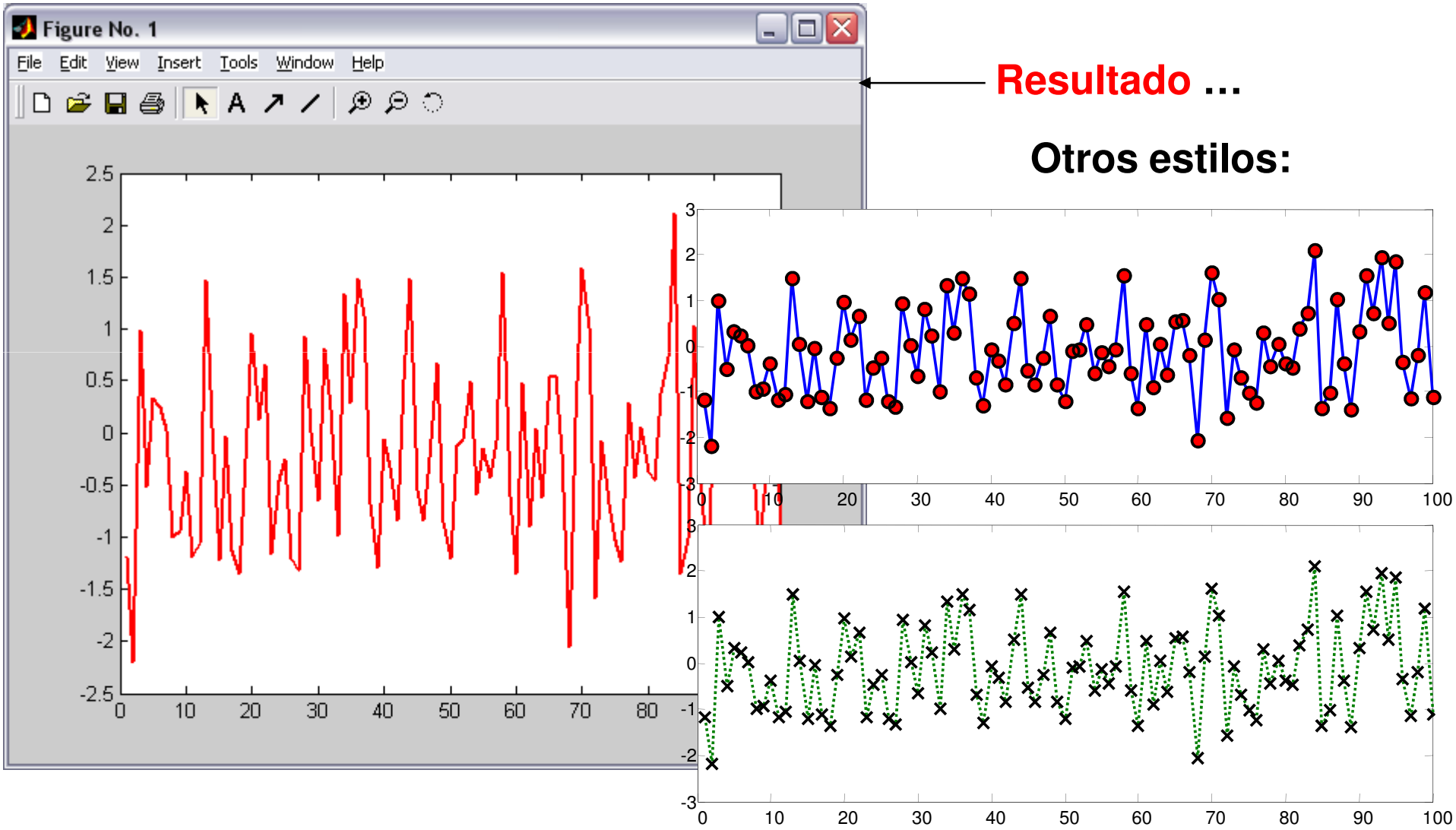


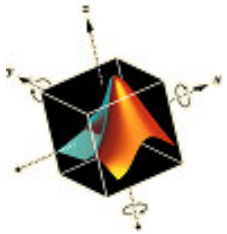
- *Añadir títulos*
- *Añadir etiquetas en ejes*
- *Cambiar las marcas*
- *Añadir grids a los ejes*
- *Cambiar color de líneas*
- *Cambiar espesor/estilo de líneas*
- *etc*

Modificación interactiva de gráficos



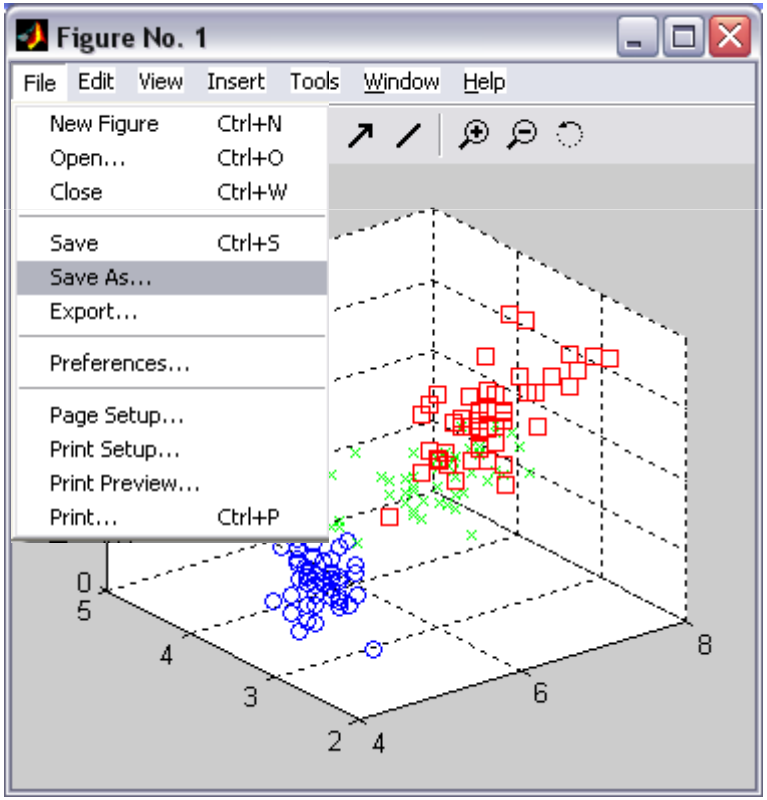
Modificación interactiva de gráficos

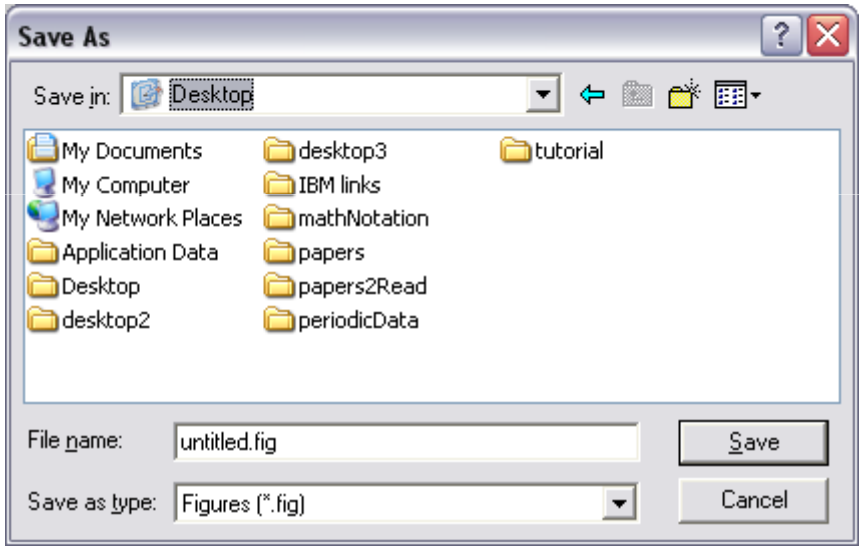




Guardando figuras

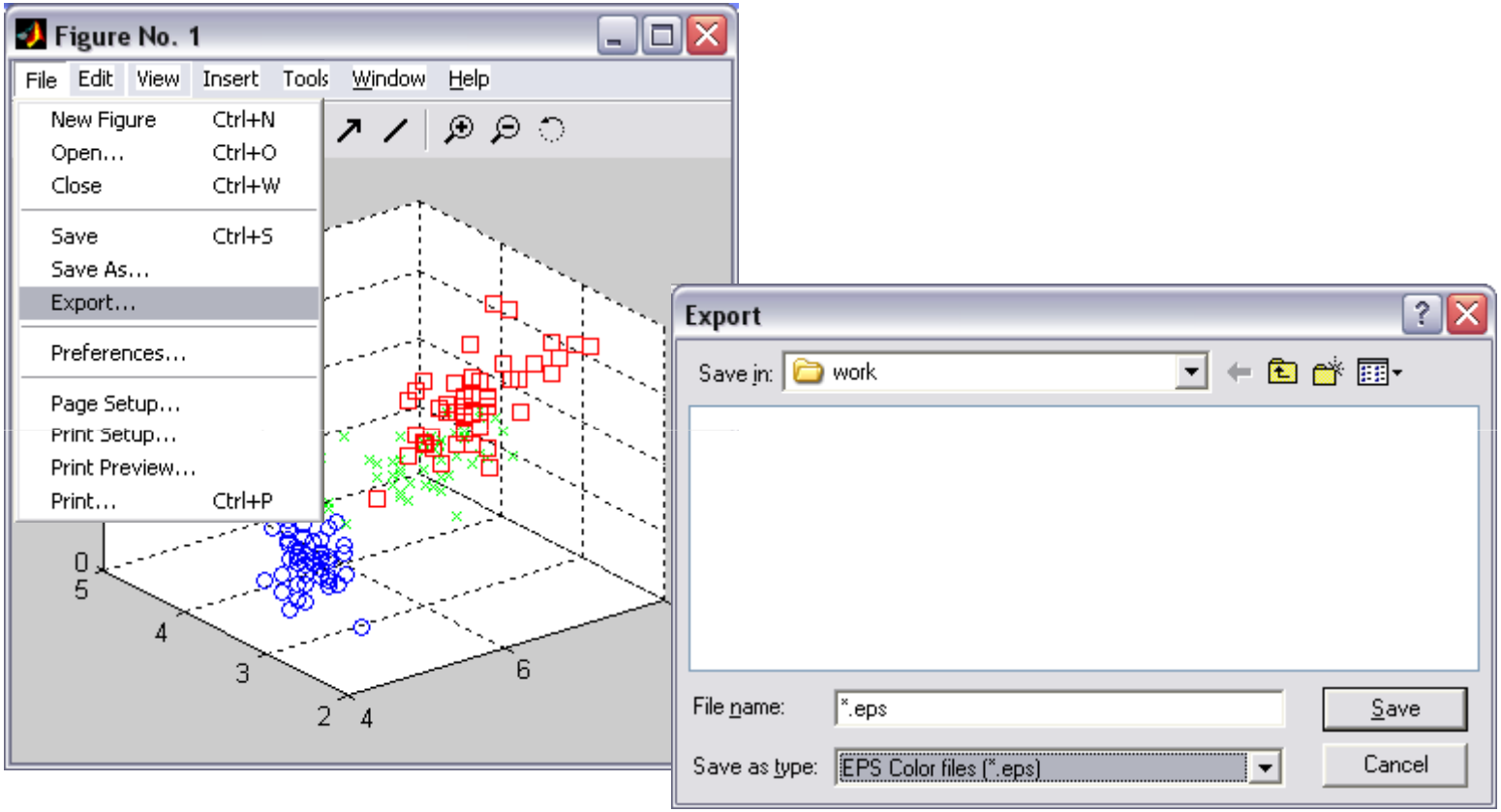
- Matlab permite guardar las figuras (.fig) para procesado posteriores





.fig se puede abrir después con Matlab

Exportando figuras



Exportación a: emf, eps, jpg, etc