



Imprimir

**Note:** You can choose to save the module's content as a PDF document, instead of printing. To do so, click on the 'Print' link above. In the window that appears, select the PDF printer as the printer of choice, then click 'Print' and enter any additional information needed.

## Visualización avanzada de datos uni-, bi- y tridimensionales

En esta Unidad analizaremos y practicaremos con algunas de las muchas posibilidades que ofrece MATLAB para la representación de datos, estos datos pueden ser los datos contenidos en un vector (datos 1-D), los datos contenidos en una matriz (datos 2-D) o en una matriz tridimensional (datos 3-D).

### Introducción

---

*Instructions:* Seguir los enlaces

## Introducción

MATLAB ofrece un gran número de posibilidades para la representación de datos, estos datos pueden ser los datos contenidos en un vector (datos 1-D), los datos contenidos en una matriz (datos 2-D) o en una matriz tridimensional (datos 3-D). La representación de los datos unidimensionales produce un gráfico en 2-D, mientras que la representación de datos bidimensionales son gráficos 3-D.

En este [cuadro](#) se muestran los tipos básicos de representaciones básicas en 2-D y 3-D:

MATLAB dibuja curvas planas y superficies. Permite agrupar y superponer representaciones, con diferentes estilos y posibilidades para los ejes de coordenadas. Además, permite a su vez realizar gráficos de tipo estadístico: de barra, histogramas, etc.

Como el entorno MATLAB está especialmente pensado para trabajar con vectores y matrices, los gráficos están orientados a la representación gráfica de vectores y matrices.

Las gráficas se muestran en una ventana aparte que llamaremos “ventana gráfica”. En dicha ventana podremos representar una o más graficas y a cada gráfica se le puede poner un nombre, se pueden nombrar los ejes de coordenadas y añadir comentarios.

Dado el gran número de funciones para representación gráfica junto con las diferentes opciones para cada una de ellas, no vamos a hacer una exposición exhaustiva de todos los comandos y su uso, sino más bien una muestra de las diferentes posibilidades. Para información sobre cada una de las funciones, nos remitimos al Centro de Documentacion de The MathWorks, en el apartado correspondiente a [gráficos](#).

### Representación gráfica de datos 1-D

---

*Instructions:* Seguir los enlaces

## Funciones elementales para la representación de datos 1-D

Fuente: Centro de documentacion de The MathWorks, "[2-D and 3-D Plots](#)", empezaremos por el tipo de representación más básico: "[Create 2D Graphs and Customize lines](#)".

El comando básico para la representación de los datos incluidos en un vector, que además pueden representar la función de una variable  $y(x)$ , es el comando **plot()**, que crea un gráfico a partir de vectores con escalas lineales sobre ambos ejes,

La function **plot** tiene diferentes formas según los argumentos de entrada. Por ejemplo, si  $Y$  es un vector,  $\text{plot}(Y)$  produce un gráfico de los valores de  $Y$  frente al índice de los elementos de  $Y$ . Si especificamos dos vectores  $X$  e  $Y$  de la misma longitud  $\text{plot}(X,Y)$  representa gráficamente  $Y$  frente a  $X$ .

Para representar la gráfica de una función  $y=f(x)$  hay que generar una tabla de valores con dos vectores de la misma longitud o número de elementos, el  $X$  que se representa en el eje de abscisas y el  $Y$  que se presenta en el de ordenadas. MATLAB representa gráficamente los puntos  $(x_1,y_1)$ ,  $(x_2,y_2)$ .... y los une según diferentes estilos de línea a indicar en las opciones del comando **plot**.

Para generar el vector  $X$  de abscisas, MATLAB dispone de la función **linspace** que genera vectores de puntos espaciados de forma lineal:

```
>> X = linspace(a,b,n);
```

(la función **logspace** genera un vector de puntos espaciados de forma logarítmica)

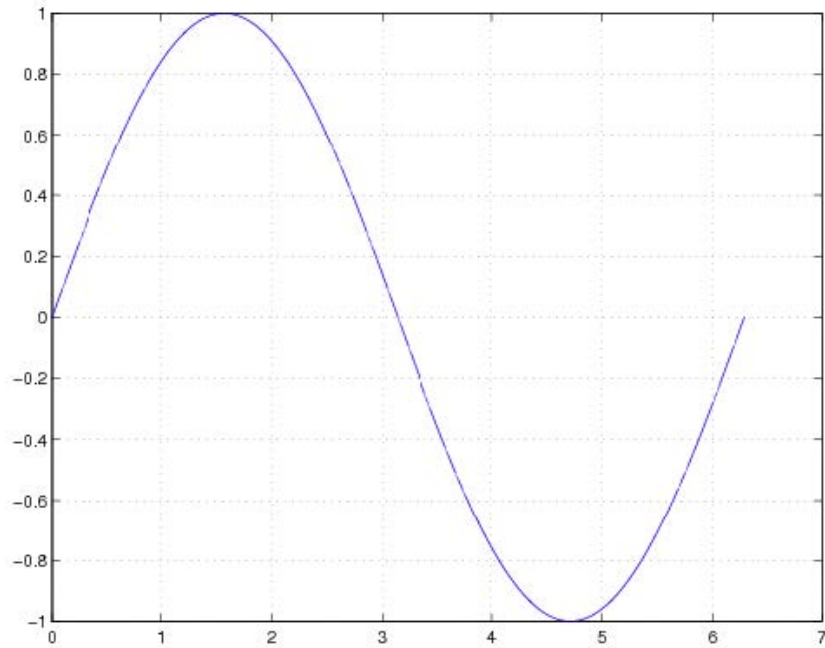
En la línea de comandos escribiremos:

```
>> plot(X,Y, 'opción'); 'opción': permite elegir color y trazo de la curva.
```

Como ejemplo, las siguientes sentencias en primer lugar se genera un vector con 100 valores equiespaciados en el intervalo  $[0, 2\pi]$ , en segundo lugar se evalúa en cada valor de dicho vector la función  $f(x)=\sin(x)$  y, finalmente, MATLAB representa el vector en el eje  $x$  y los valores de la función seno en el eje  $y$ :

```
x = linspace(0, 2*pi,100);  
y = sin(x);  
plot(x,y)  
grid on % Comando opcional, en el gráfico se representar una cuadrícula.
```

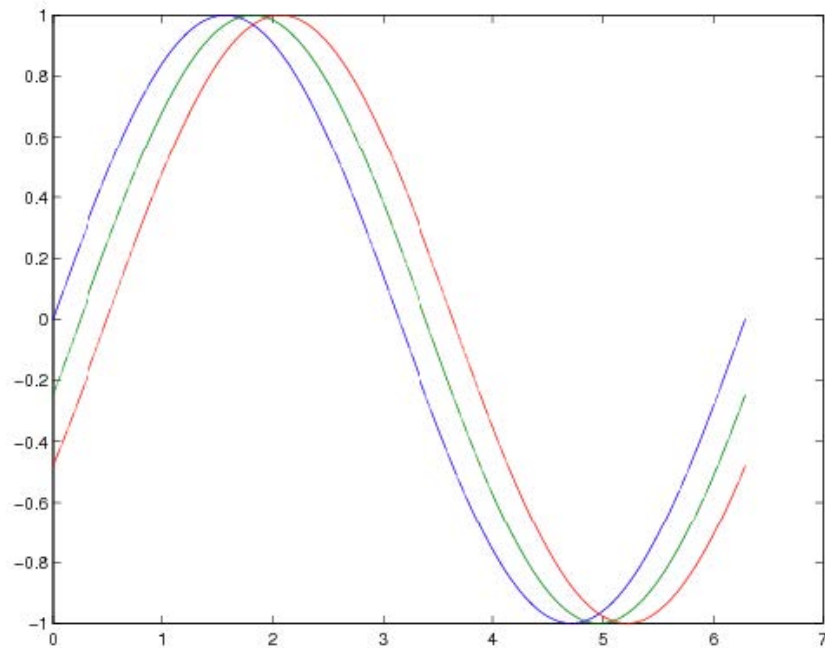
Si no se añade nada más, los ejes de coordenadas se seleccionan de manera automática.



Existen gran número de posibilidades, se pueden representar múltiples gráficas en unos mismos ejes, MATLAB las colorea automáticamente en colores diferentes para distinguir los valores de las diferentes listas de datos, aunque si se quiere se puede seleccionar el color, el tipo de trazo...

Por ejemplo, representamos dos curvas como función de  $t$  en diferentes colores

```
y = sin(t);  
y2 = sin(t-pi/4);  
plot(t,y,t,y2)
```



Nótese que la sintaxis para representar, múltiples curvas

```
>> plot(x1,y1,x2,y2,...,'opciones');
```

En general, salvo que utilizemos el comando tal y como acabamos de ver para representaciones múltiples, cada vez que mandamos representar una gráfica, se abre una ventana o “figura” nueva. Se puede utilizar el comando

```
>> hold on / hold off
```

Para que todos los gráficos que se ordene dibujar entre los comandos hold on y hold off se representan en la misma figura. Si hay una figura abierta se dibujan en ésta.

También es posible dibujar una función con el comando **fplot** cuya sintaxis es la siguiente:

```
>> fplot('f(x)',[xmin,xmax]);
```

Así, este comando admite como argumento un nombre de función o de un fichero .m en el que está definida la función a representar. Por ejemplo:

```
>> fplot('sin(x)',[0,2*pi,-1,1])
```

También es habitual querer visualizar varias gráficas diferentes en una misma figura pero sin sobre ponerlas en los mismos ejes. Para ello es útil el comando **subplot**, con el cual una ventana gráfica se puede dividir en m particiones horizontales y n verticales para representar mxn figuras. Cada una de las particiones tendrá sus ejes aunque las propiedades serán comunes a todas ellas. Ejemplo:

```
x=0:0.1:2*pi;
y1=2*sin(x);
y2=2*cos(x);
y3=exp(1-x.^2);
y4=x.^2;
subplot(2,2,1), plot(x,y1)
subplot(2,2,2), plot(x,y2)
subplot(2,2,3), plot(x,y3)
subplot(2,2,4), plot(x,y4)
```

Comprobar vosotros mismos el resultado de las sentencias anteriores.

El número de posibilidades para representar los datos 1-D con MATLAB son muy grandes, acabamos de ver una pequeña muestra a la que se pueden añadir como

- Especificar el estilo de la línea que una los puntos.
- Seleccionar los ejes y el estilo de los ejes.
- Especificando el color y el grosor de la curva
- Añadir gráficas a una existente.
- Representar sólo los puntos con datos, estilos.
- Representar líneas y marcadores si la salida es en blanco y negro.
- Definir el estilo de las líneas por defecto.

La información correspondiente la tenéis el siguiente documento en la ayuda de MATLAB al que se puede acceder a través del botón de ayuda en la pantalla principal del programa. Como hemos dicho al principio de esta sección, la documentación referente a las funciones básicas para representación en 2-D de datos unidimensionales o gráficas de funciones junto con varios ejemplos de uso, podéis encontrarlas en el apartado "[Create 2-D Graphs and Customize Lines](#)".

También podéis ver los siguientes vídeos (en la misma página también podéis encontrar otros vídeos sobre representación de datos que pueden resultaros de interés):

- [Using plotting basic functions](#)
- [Creating a basic plot interactively](#)

## Tipos de gráficos 2-D y 3-D

*Instructions:* Seguir los enlaces

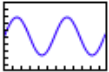
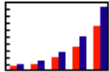
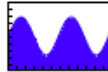
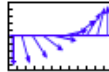

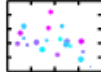
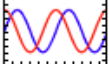


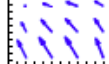


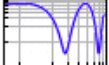




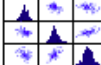
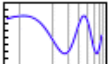

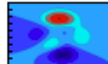

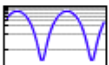
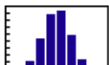
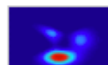
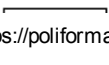
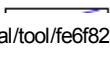
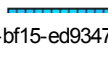
## Tipos de gráficos 2-D y 3-D

Aparte del commando básico plot, MATLAB ofrece una gran variedad de funciones para representar datos junto con una amplia gama de opciones para modificar y personalizar los gráficos. En el apartado "[2D and 3D Plots](#)" se analizan con detalle y se dan múltiples ejemplos de cómo generar este tipo de gráficos.

Dado el gran número de tipos de gráficos tanto en 2-D como 3-D, no haremos una exposición exhaustiva sino que mostraremos las diferentes posibilidades que hay. En las siguientes tablas se muestran los tipos de gráficos que se pueden crear con ayuda de MATLAB tanto 2-D como 3-D, si seguís en enlace y luego pincháis sobre cualquiera de las órdenes, os mostrará la sintaxis y un ejemplo de uso. En el caso de gráficos en 2-D incluye diagramas de barras, histogramas, curvas en coordenadas polares, líneas de nivel, representación de campos vectoriales.... En el caso de gráficos 3-D, se dibujan curvas paramétricas en 3-D, superficies, volúmenes.

### Tipos de gráficos en 2-D

La tabla se encuentra en el siguiente enlace ([Common Graphics Functions in MATLAB](#)) en el cual pinchando en cada una de las funciones nos lleva a la documentación de dicha función.

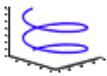
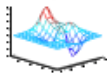

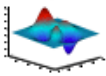
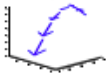
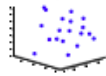

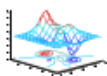

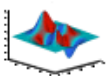

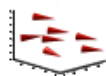
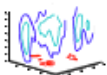
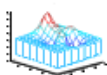

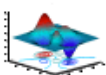
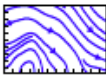
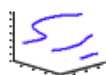

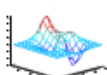

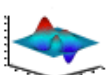
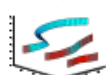





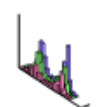


Line Graphs	Bar Graphs	Area Graphs	Direction Graphs	Radial Graphs	Scatter Graphs
<a href="#">plot</a> 	<a href="#">bar (grouped)</a> 	<a href="#">area</a> 	<a href="#">feather</a> 	<a href="#">polar</a> 	<a href="#">scatter</a> 
<a href="#">plotyy</a> 	<a href="#">barh (grouped)</a> 	<a href="#">pie</a> 	<a href="#">quiver</a> 	<a href="#">rose</a> 	<a href="#">spy</a> 
<a href="#">loglog</a> 	<a href="#">bar (stacked)</a> 	<a href="#">fill</a> 	<a href="#">comet</a> 	<a href="#">compass</a> 	<a href="#">plotmatrix</a> 
<a href="#">semilogx</a> 	<a href="#">barh (stacked)</a> 	<a href="#">contourf</a> 		<a href="#">ezpolar</a> 	
<a href="#">semilogy</a> 	<a href="#">hist</a> 	<a href="#">image</a> 			
<a href="#">stairs</a> 	<a href="#">pareto</a> 	<a href="#">pcolor</a> 			

					
contour 	errorbar 	ezcontourf 			
ezplot 	stem 				
ezcontour 					

### Tipos de gráficos en 3-D

En esta tabla se muestran los tipos de gráficos 3-D y de funciones para representación de volúmenes. Algunas funciones generan figuras geométricas (esferas, cilindros, elipsoides) que pueden utilizarse para formar otras figuras geométricas mas complejas y sobre las que se pueden añadir datos propios.

La tabla se encuentra en el siguiente [enlace](#), en el cual pinchando en cada una de las funciones nos lleva a la documentación de dicha función.

Line Graphs	Mesh Graphs and Bar Graphs	Area Graphs and Constructive Objects	Surface Graphs	Direction Graphs	Volumetric Graphs
<p>plot3</p> 	<p>mesh</p> 	<p>pie3</p> 	<p>surf</p> 	<p>quiver3</p> 	<p>scatter3</p> 
<p>contour3</p> 	<p>meshc</p> 	<p>fill3</p> 	<p>surfl</p> 	<p>comet3</p> 	<p>coneplot</p> 
<p>contourslice</p> 	<p>meshz</p> 	<p>patch</p> 	<p>surfc</p> 	<p>streamslice</p> 	<p>streamline</p> 
<p>ezplot3</p> 	<p>ezmesh</p> 	<p>cylinder</p> 	<p>ezsurf</p> 		<p>streamribbon</p> 
<p>waterfall</p> 	<p>stem3</p> 	<p>ellipsoid</p> 	<p>ezsurfc</p> 		<p>streamtube</p> 
	<p>bar3</p> 	<p>sphere</p> 			
	<p>bar3h</p> 				



## Representación de datos 2-D: superficies, volúmenes

*Instructions:* Seguir los enlaces

### Representación gráfica de datos 2-D

En este apartado veremos cómo representar datos bidimensionales, donde a cada par de valores de  $x - y$  le asociamos un valor que se representará en el eje  $z$ . Dichos datos vendrán dados como los elementos de una matriz bidimensional. También valdrá para representar gráficamente la gráfica de una función de dos variables  $f(x, y)$ . MATLAB define una superficie como la gráfica de una función de dos variables, a partir de la coordenada  $z$  definida sobre un malla en el plano  $x-y$ . La gráfica se construye uniendo los puntos adyacentes con líneas rectas. Las representaciones gráficas con superficies son útiles para visualizar matrices demasiado grandes para mostrar de forma numérica y para visualizar gráficas de funciones de dos variables.

La documentación exhaustiva de todas las funciones, junto con ejemplos de cómo usarse está en el enlace: "[Representing Data as a Surface](#)" y "[Surface and Mesh Plots](#)"

MATLAB puede crear diferentes tipos de superficies; dispone de diferentes comandos para representar solo las líneas o el enmallado sobre la superficie (gráficas con la función **mesh**) o colorear toda la facetas de la superficie (función **surf**).

Los comandos **mesh** y **surf** crean superficies 3-D a partir de los datos de una matriz. Si  $Z(i,j)$  es una matriz cuyos valores representan las alturas de una superficie sobre una malla  $(i,j)$ , entonces la sentencia

```
>> mesh(Z)
```

genera una superficie coloreada uniendo los puntos adyacentes por rectas y la muestra sobre un sistema de tres ejes de referencia.

De forma análoga, la sentencia

```
>> surf(Z)
```

genera una superficie coloreada, aunque a diferencia del caso anterior, rellena las facetas de la superficie con color. Normalmente, estas facetas son cuadriláteros, cada uno de ellos de un único color. Existen comandos (**shading**) para eliminar las líneas o aumentarlas.

Se pueden controlar la apariencia visual de las superficies con las propiedades de las superficies, pudiendo especificar el estilo de las líneas, los marcadores de los vértices, la forma de colorear las facetas, la iluminación, etc.

Para visualizar una función de dos variables  $z=f(x,y)$ , primero hay que generar la malla donde evaluar la función, eso puede hacerse con la función **meshgrid**.

### Ejemplos

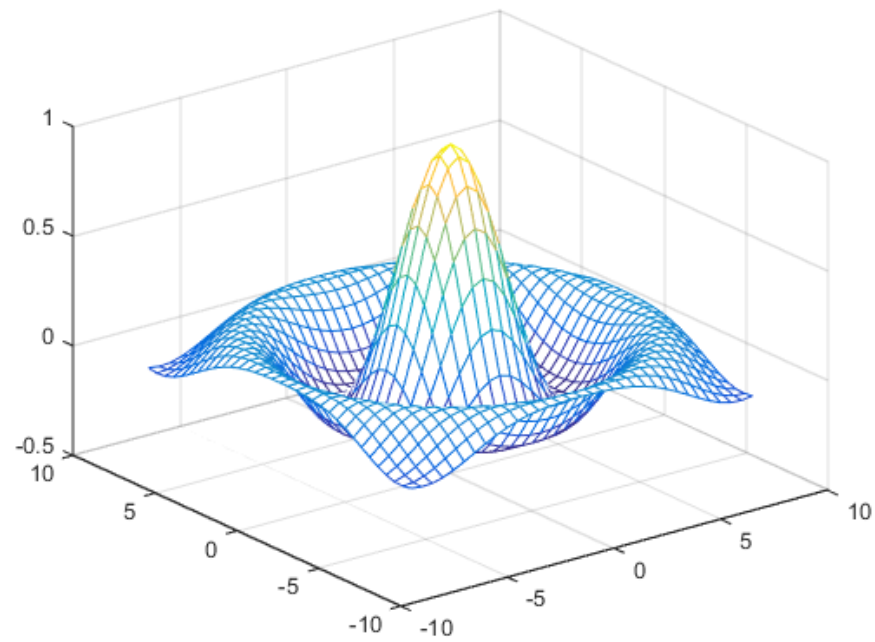
Para ilustrar el uso de los comandos anteriores, vamos a considerar la función  $\sin(r)/r$  y evaluarla en el conjunto con  $x$  e  $y$  entre  $-8$  y  $8$ .

```
[X,Y] = meshgrid(-8:5:8);
R = sqrt(X.^2 + Y.^2) + eps;
```

La matriz **R** contiene las distancias desde el centro de la matriz que es el origen de coordenadas. Añadiendo **eps** se evita dividir por cero al calcular la función. Con los órdenes generamos la función y la representamos en 3-D:

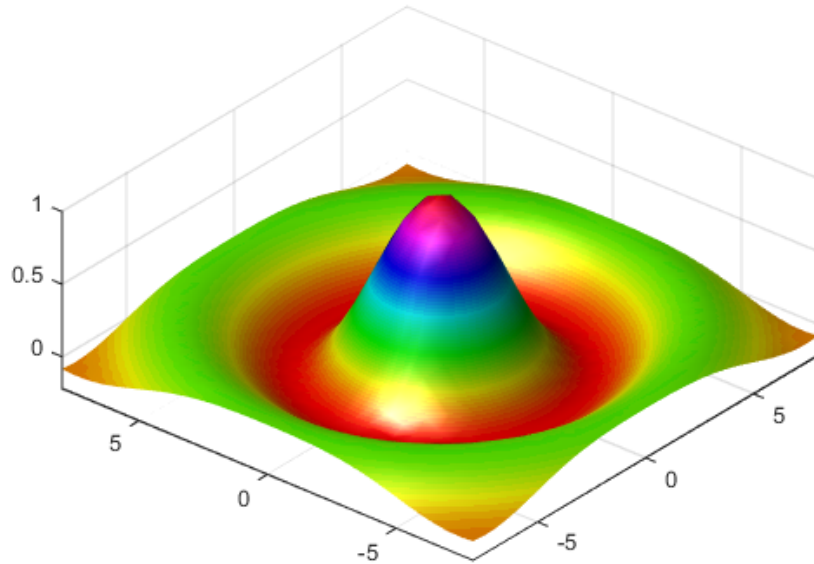
```
Z = sin(R)./R;
figure
```

```
mesh(X,Y,Z)
```



Podemos controlar diferentes opciones:

```
figure
surf(X,Y,Z,'FaceColor','interp',...
      'EdgeColor','none',...
      'FaceLighting','phong')
daspect([5 5 1])
axis tight
view(-50,30)
camlight left
```



Podéis ver el ejemplo entero en el siguiente enlace "[Representing Data as a Surface](#)" y "[Coloring Mesh and Surface Plots](#)".

## Visualización de datos 3-D

---

*Instructions:* Seguir los enlaces

## Visualización de datos 3-D

Visualizar datos 3-D consiste en la creación de algún tipo de representación gráfica de los datos definidos sobre una matriz o malla tridimensional. Dichos datos pueden ser escalares o vectoriales. Los escalares consisten en un dato único en cada punto de la malla, mientras que en el caso vectorial en cada punto hay asociado un vector, generalmente, con tres componentes.

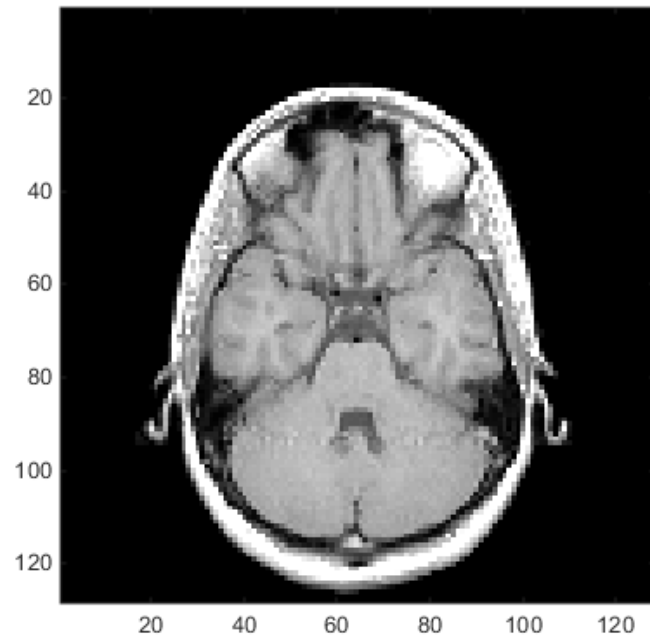
La documentación exhaustiva de todas las funciones, junto con ejemplos de cómo usarse está en el "[Volume Visualization](#)", "[Techniques for Visualizing scalar Volume Data](#)" y "[Overview of Volume Visualization](#)".

## Volumen de datos escalares:

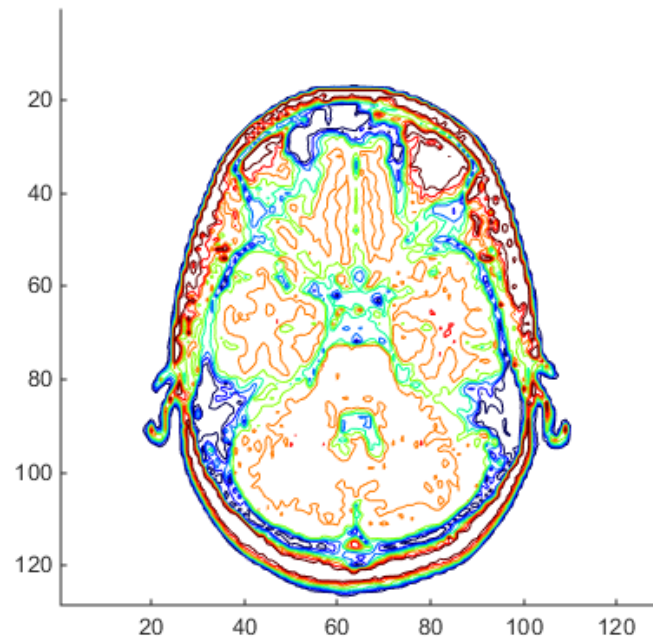
Un volumen típico de datos está formado por una matriz 3-D de datos (que representarían cierta función de tres variables) junto con tres matrices tridimensionales de las mismas dimensiones que contienen, respectivamente, los valores de las coordenadas x, y, z de los puntos a los que corresponden los datos contenidos en la matriz de datos.

MATLAB posee varias funciones para visualizar este tipo de datos: slice, contourslice...

Un ejemplo de este tipo de datos es el de una MRI (imagen de resonancia magnética nuclear), los datos son tridimensionales aunque luego se pueden mostrar imágenes 2-D de diferentes secciones. Ejemplo:



Utilizando el comando contourslice:



Véase el ejemplo completo en el [enlace](#).

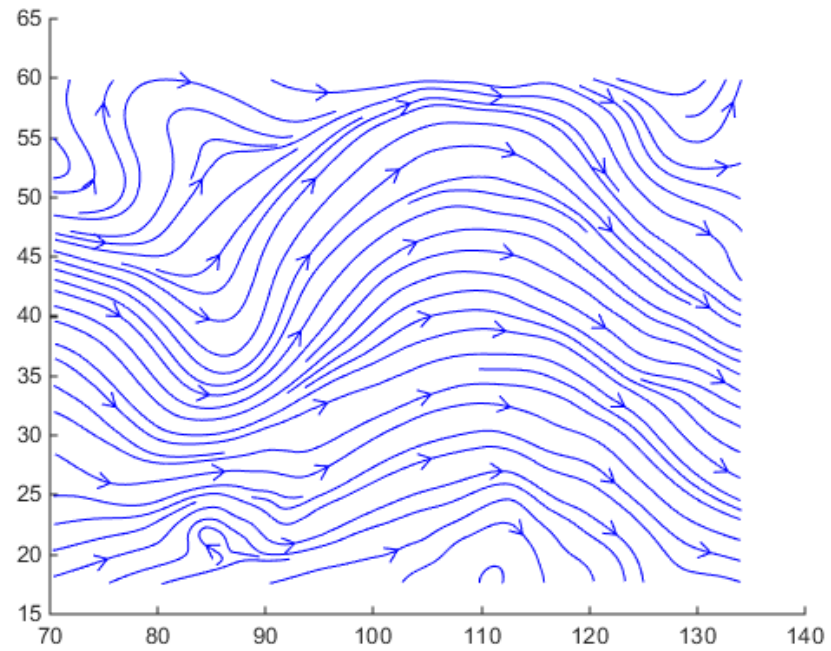
### Volumen de datos vectoriales:

Los datos vectoriales 3-D contienen más información que los escalares ya que en cada punto de coordenadas de la mala tridimensional hay asociado un vector con tres componentes. La velocidad de la corriente de un fluido sería un ejemplo de datos vectoriales.

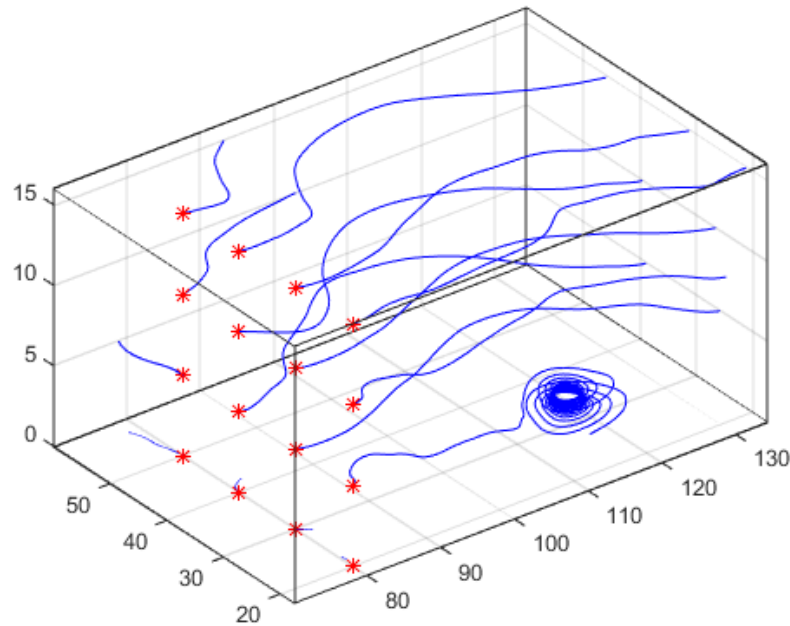
La documentación exhaustiva de todas las funciones, junto con ejemplos de cómo usarse está en el enlace "[Visualizing Vector Volume Data](#)".

En la documentación podemos ver MATLAB posee diferentes técnicas para visualizar este tipo de datos. Remitimos allí para su análisis completo.

Veamos un ejemplo: a partir de los datos tridimensionales que dan la velocidad del viento, se pueden obtener gráficos de tipo:



Se pueden obtener trayectorias a partir de unos puntos iniciales:



El ejemplo completo lo podéis encontrar en el enlace anterior "[Visualizing Vector Volume Data](#)".

También es muy interesante el análisis de los datos contenidos en un volumen analizando cortes por distintos planos, eso lo podéis encontrar en el enlace "[Exploring Volumes with Slice Planes](#)".

También son muy interesantes las posibilidades de MATLAB para representación de campos vectoriales. Los alumnos interesados podéis visitar el siguiente enlace "[Vector Fields](#)" para consultar las funciones básicas y ver algún ejemplo.

### Herramienta gráfica interactiva

*Instructions:* Seguir los enlaces

### Herramienta gráfica interactiva

En versiones modernas de MATLAB existe una herramienta gráfica que permite la construcción de gráficos de forma interactiva. Al teclear en la línea de comandos del entorno MATLAB la orden:

```
>> plottools
```

se abre una ventana de figura con la que podemos construir gráficos 2-D y 3-D de forma interactiva.

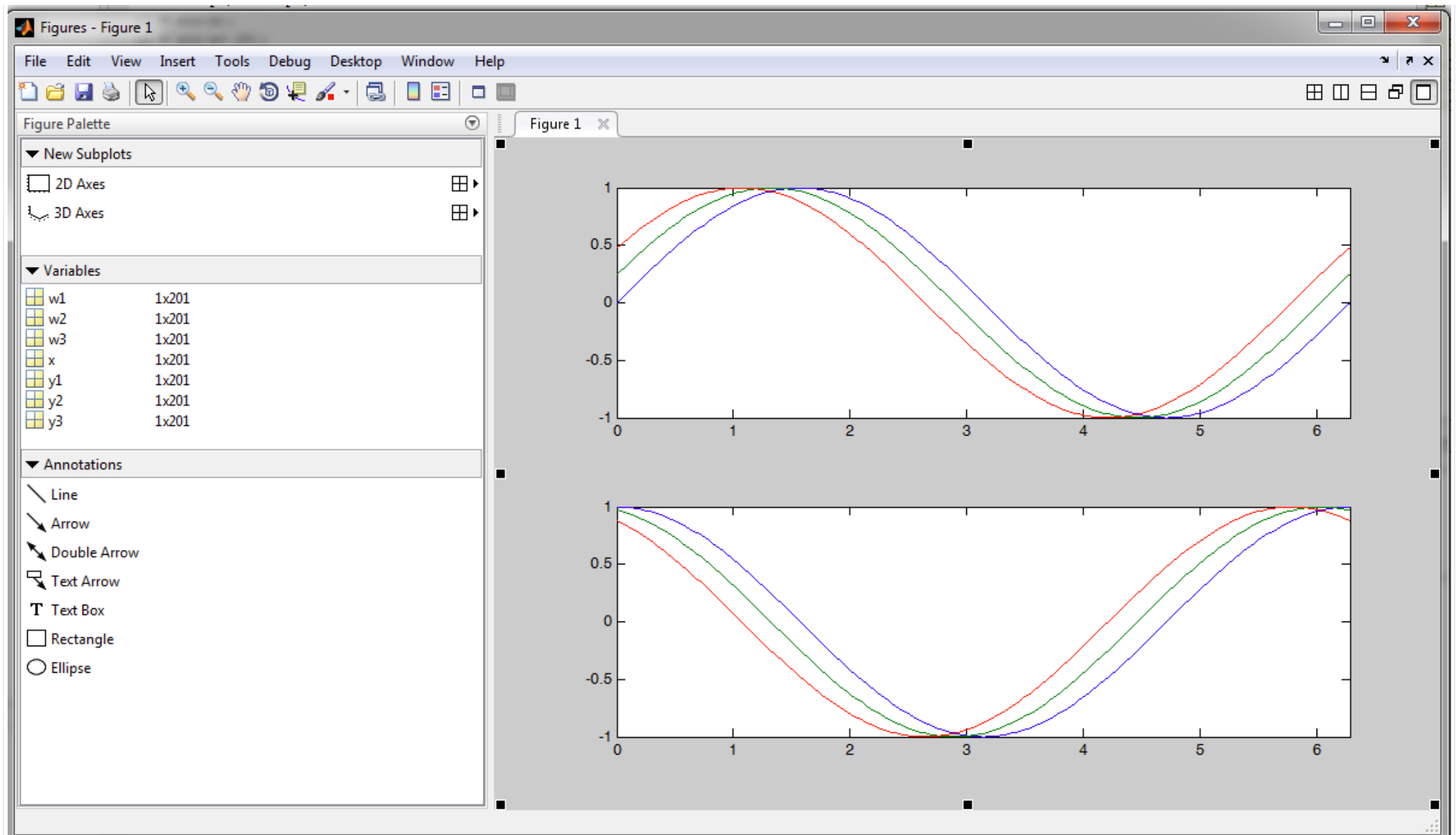
Véase la documentación para el uso de esta función en el enlace a la orden "[plottools](#)", y al enlace "[Customize Graph Using Plot tools](#)".

Por ejemplo, las siguientes sentencias generan las gráficas de seis funciones para representar en dos subventanas (subplot) diferentes:

```
% Primer subplot
x = 0:pi/100:2*pi;
y1 = sin(x);
y2 = sin(x+.25);
y3 = sin(x+.5);
subplot(2,1,1);
plot(x,y1,x,y2,x,y3);
axis tight;
% Segundo subplot
w1 = cos(x);
w2 = cos(x+.25);
w3 = cos(x+.5);
subplot(2,1,2);
plot(x,w1,x,w2,x,w3);
axis tight;
```

Se cargan en la ventana de "figura", escribiendo plottools, y se abre la siguiente ventana:





En la figura aparecen las dos subventanas con las gráficas superimpuestas, y a un lado aparece un cuadro (Figure Palette) con herramientas para la manipulación de la gráfica.

Esta herramienta es especialmente útil para la exploración visual de los datos de forma interactiva. Véase el apartado "[Data Exploration](#)" en el Centro de Documentación de The Mathworks.

## Actividades propuestas

---

*Instructions:* Realizar las actividades y no pasar a las siguientes unidades hasta no haber sido capaz de completarlas con éxito.

## Actividades

### ¿Qué tengo que hacer?

Un segundo paso para el seguimiento adecuado del curso es la representación gráfica de los datos. Por tanto es necesario alcanzar cierta destreza en la creación de gráficos 2-D y 3-D. Se recomienda que intentéis por vuestra cuenta repetir con MATLAB algunos de los ejemplos que habéis encontrado en los apartados anteriores.

Para comprobar que habéis alcanzado un nivel adecuado en este aspecto, vamos a proponer algunas actividades para este tema. Van a consistir en generar datos 1-D y 2-D y representarlos utilizando diferentes opciones.

IMPORTANTE: No pases a las siguientes unidades hasta no haber sido capaz de completar con éxito estas actividades. Las actividades de este Tema NO hay que subirlas a ningún sitio ni entregarlas. Las dudas consultarlas en el chat/foro o con el profesor.

### Actividad 1

Considerar las siguientes funciones de una variable con  $x$  variando entre 0 y 5:

- $f(x)=e^{-x}*\cos(4*\pi*x)$
- $f(x)=e^{-x}*\sin(4*\pi*x)$

Realizar los siguientes ejercicios:

1. Generar dos vectores de tamaño 100 conteniendo cada una de las funciones.
2. Representar las dos funciones en la misma figura.
3. Representar las gráficas utilizando diferentes estilos para las líneas, los marcadores...
4. Añadir nombre a las gráficas y a los ejes. Personalizar las gráficas utilizando la función **plottools**.

Reproducir para estas funciones los diferentes modos de representación gráfica de las líneas y colores de los ejemplos que se encuentran en el [enlace](#).

### Actividad 2

Considerar la función de dos variables con  $x$  e  $y$  entre  $-\pi$  y  $\pi$ :

- $f(x,y)=\sin(x-y)*\cos(x+y)$

Realizar los siguientes ejercicios:

1. Generar una malla para los valores de  $x$  e  $y$  (entre  $-\pi$  y  $\pi$ ) con 50 puntos en cada dirección (por tanto, 50x50 en total), usando **meshgrid**.
2. Generar una matriz que contenga los valores de la función  $f(x,y)$ .
3. Representar la superficie usando **mesh** y **surf**.
4. Hacer pruebas con distintas opciones para el color, ejes....

Reproducir para esta función los diferentes modos de representación gráfica y coloreado de los ejemplos que se encuentran en el [enlace](#).