**Facilidades gráficas de Matlab**

Matlab incluye facilidades gráficas que permiten representar los datos y las operaciones de una manera cómoda para su interpretación. En este apartado veremos algunas de tales facilidades.

En primer lugar creemos una sinusoide y mostremos dicha sinusoide por pantalla. Para ello comenzamos creando un vector de abscisas, concretamente, un vector de muestras temporales separadas diez milisegundos:

*>> X=0:(1/100):1;*

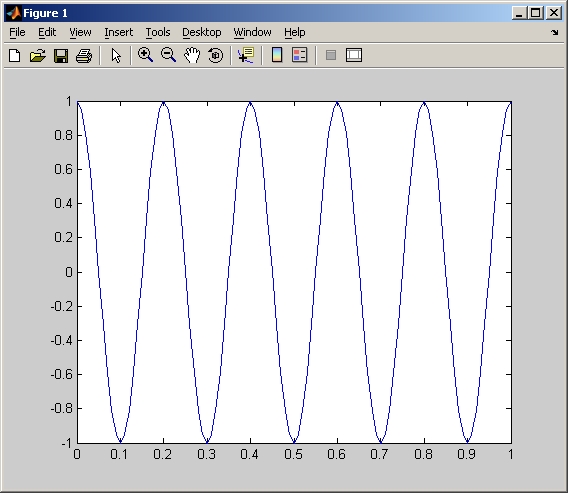
A continuación, sobre dichas muestras temporales, definamos los puntos de la sinusoide:

*>>Y=cos(2\*pi\*5\*X);*

Con ello habremos creado muestras de una sinusoide de 5 Hz. Para comprobar los resultados podemos ejecutar

*>> plot(X,Y)*

y el resultado debería ser el que se indica a continuación



La función *plot*dibuja una nueva figura en la ventana activa (en todo momento Matlab tiene una ventana activa de entre todos las ventanas gráficas abiertas), o abre una nueva figura si no hay ninguna abierta, sustituyendo cualquier cosa que hubiera dibujada anteriormente en esa ventana. A continuación, puede probarse lo siguiente (se comienza cerrando la ventana activa, para que al crear la nueva ventana aparezca en primer plano):

*>>% el carácter % se utiliza para añadir código que no se ejecuta (comentarios)*

*>> close % se cierra la ventana gráfica activa anterior*

*>> grid % se crea una ventana con una cuadrícula*

*>> plot(X,Y) % se dibuja la función coseno borrando la cuadrícula*

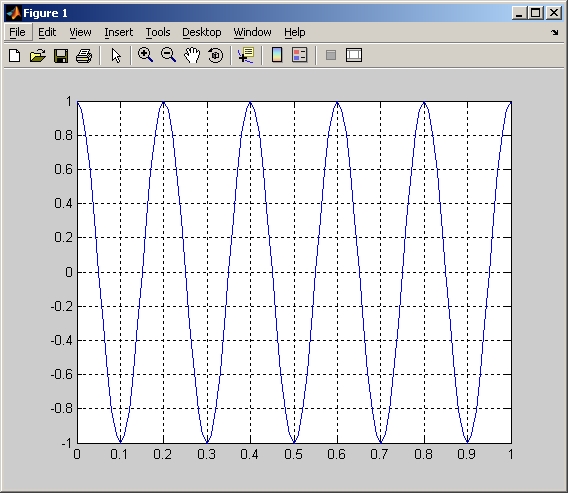
El resultado en este caso es el mismo que el anterior. Se puede observar la diferencia con la secuencia que sigue:

*>> close*

*>> plot(X,Y) % se crea una ventana y se dibuja la función coseno*

*>> grid % se añade la cuadrícula sin borrar la función coseno*

Ahora debería obtener la siguiente figura:



En el primer caso MATLAB ha creado la cuadrícula en una ventana nueva y luego la ha borrado al ejecutar la función *plot*. En el segundo caso, primero ha dibujado la función y luego ha añadido la cuadrícula. Esto es así porque hay funciones como *plot*que por defecto crean una nueva figura, y otras funciones como *grid*que se aplican a la ventana activa modificándola, y sólo crean una ventana nueva cuando no existe ninguna ya creada. A continuación veremos cómo (mediante la función *hold*)pueden añadirse gráficos a una figura ya existente respetando su contenido.

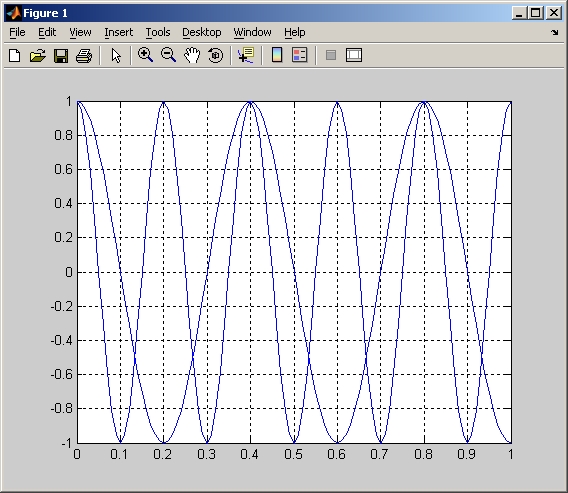
Pruebe ahora lo siguiente:

*>>Z=cos(2\*pi\*2.5\*X);*

*>>hold on*

*>>plot(X,Z)*

Debería obtener la siguiente figura:



Con esta secuencia de comandos hemos dibujado en la misma gráfica otra sinusoide de frecuencia 2.5 Hz. En este momento resulta interesante explorar cómo se pueden dibujar las dos sinusoides mediante diferentes colores sin más que añadir un tercer argumento a la función *plot*. Ejecute *help plot* y podrá ver las distintas opciones que se le presentan.

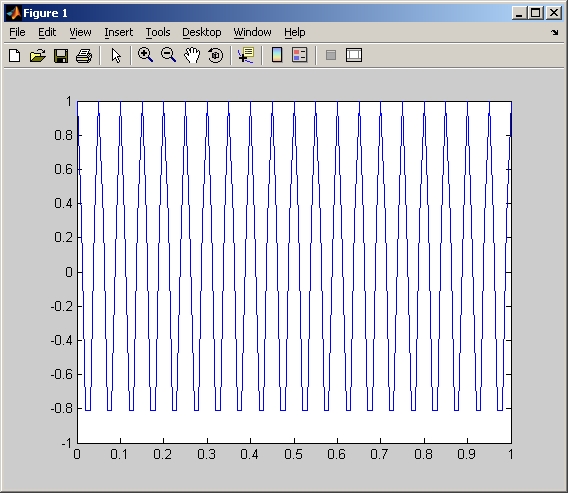
Observemos ahora lo que ocurre si intentamos representar una sinusoide de mayor frecuencia. Ejecute lo siguiente:

*>>close all; % cerramos todas las ventanas gráficas*

*>>W=cos(2\*pi\*20\*X);*

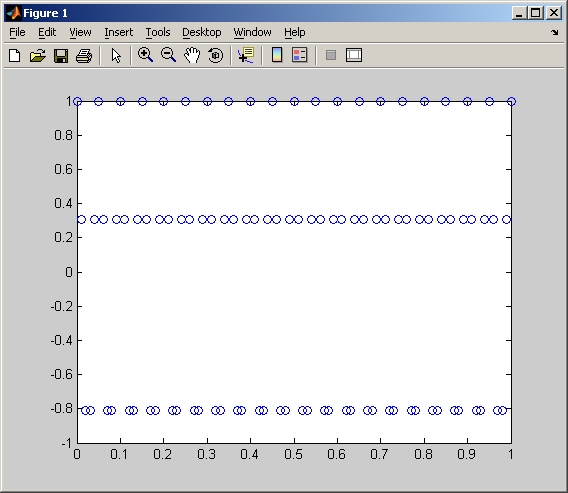
*>>plot(X,W)*

Obtendrá la siguiente figura:



El gráfico obtenido en este caso no se corresponde con la sinusoide esperada. Lo que ocurre es que Matlab representa los puntos que contiene el vector W frente a los del vector X y utiliza líneas por defecto para unirlos. Cuando el número de puntos es suficientemente grande, no apreciamos distorsión en la figura. Sin embargo, cuando no contamos con muestras suficientes, la representación obtenida no coincide con lo esperado. Para observar las muestras que realmente pinta Matlab (sin las líneas que las unen), basta con teclear lo siguiente:

*>>plot(X,W,'o')*



En este caso hemos utilizado el símbolo "o" para representar la función, evitando las líneas que automáticamente incluye Matlab. De nuevo consultando la ayuda en línea se puede investigar sobre ésta y otras posibilidades de la función *plot*. En concreto, resulta interesante ver qué ocurre cuando la función sólo recibe un vector como argumento, cuando alguno de los argumentos es una matriz, y otras posibilidades de representación conjunta sin hacer uso de la orden *hold*.

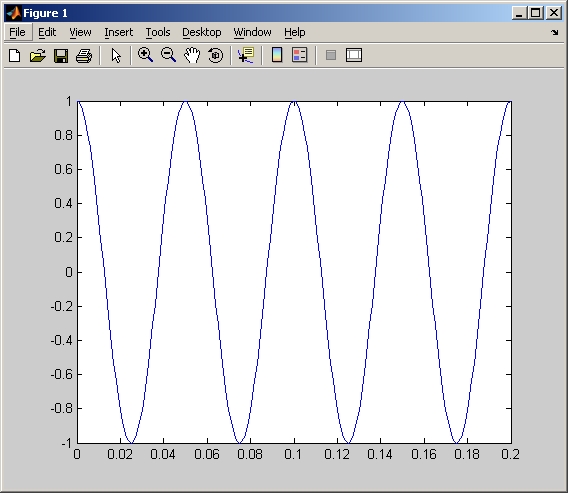
Para ver correctamente la sinusoide de 20 Hz, bastaría con utilizar un eje de abscisas más "granulado":

*>>X2=0:(1/1000):0.2; %abscisas con menor separación (nótese que representamos sólo hasta 0.2)*

*>>W2=cos(2\*pi\*20\*X2);*

*>>plot(X2,W2)*

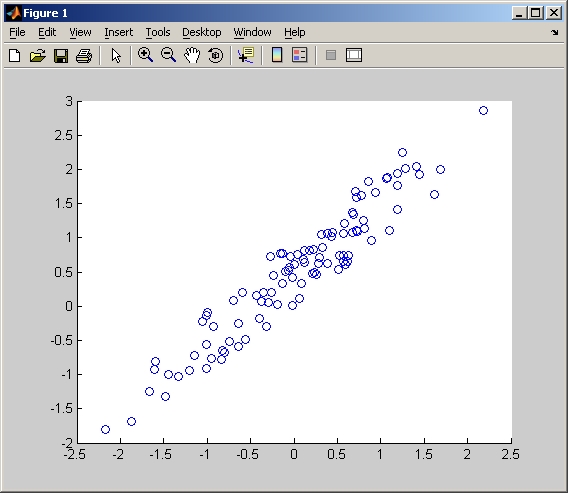
La figura obtenida en este caso es la que se muestra a continuación. Nótese que el número de períodos representados ha disminuido debido a que sólo se toman puntos hasta 0.2 en el eje de abscisas.



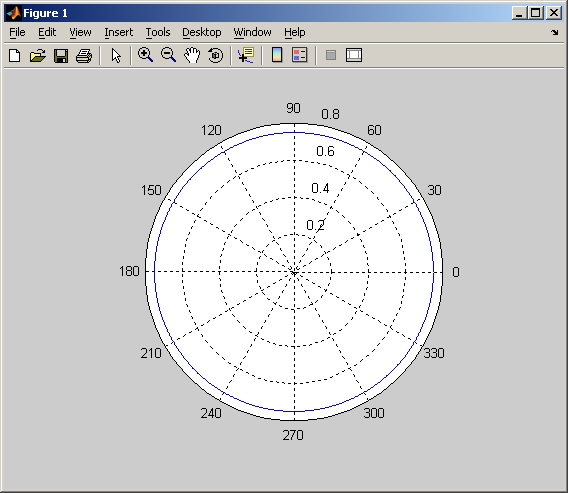
Otro tipo de representación es el conocido como **diagrama de dispersión.** En este caso se pretende mostrar cómo se distribuyen en el plano un determinado conjunto de datos obtenido. Cada dato vendrá definido por su par de coordenadas (x,y). Concretamente:

>> abscisas=randn(100,1);   
>> ordenadas=abscisas+rand(100,1);   
>> scatter(abscisas,ordenadas)

En este ejemplo se ha creado el vector  **abscisas**, consistente en 100 números aleatorios. A continuación se ha creado el vector ordenadas sumando a abscisas otro vector de números aleatorios independiente del anterior. Finalmente, se muestra en pantalla el diagrama de  dispersión correspondiente a estos datos así creados.  El resultado (en cada ejecución será distinto) tendrá un aspecto como el que mostramos a continuación



**Ejercicio:** haciendo uso de la orden **polar,** dibuje en pantalla una circunferencia de radio a escoger. Debería obtener algo parecido a la siguiente figura:

  
    
    
    
**LENGUAJE Y SISTEMAS DE COMPUTACIÓN.**

**1º Ingeniería Química. Curso 2001/02**

MATLAB: Guión de Prácticas (ii)

1. ***Representación simultánea de más de una gráfica***

La función **plot** admite como argumentos tantos pares de vectores como se deseen, lo que permite representar en una misma gráfica más de un conjunto de datos. Para ello, si los pares de datos a representar son los vectores x1, y1, x2, y2, …, xn, yn, la función se invocaría con la siguiente sintaxis:

**plot(x1, y1, x2, y2, …, xn, yn)**

* ***Ejercicio 2***

Representar en una única gráfica las cuatro representaciones del ejercicio anterior.

Con esta utilización de **plot** cíclicamente se le asignan distintos colores a cada conjunto de datos. Pueden modificarse las características particulares de la representación de cada conjunto, añadiendo la constante de caracteres correspondiente después de cada par.   
 

* ***Ejercicio 3***

Repetir la representación anterior realizando las gráficas con distintos tipos de línea, símbolos de representación y colores.

Para ello se debe seguir una sintaxis a la que utilizasteis con el plot simple del ejercicio 1, de la manera que muestra el siguiente ejemplo:

**plot(x1, y1, 'r\*--', x2, y2, 'cx-', …, xn, yn, 'ko:')**   
 

Al finalizar este ejercicio, puedes pasar al siguiente guión pulsando en la flecha:  [I:\ARCHIVOS MATLAB\Archivos HTML\Archivos HTML-2\Manual_Matlab\Graficos2 MATLAB_archivos\flecha.gif](http://www-gsi.dec.usc.es/~alberto/lsc/matlab/ezplot.html)

**LENGUAJE Y SISTEMAS DE COMPUTACIÓN.**

**1º Ingeniería Química. Curso 2001/02**

MATLAB: Guión de Prácticas (iii)

1. ***Características de las representaciones gráficas.***

En la ventana de gráficos es posible incluir los siguientes elementos: título de la gráfica, de los ejes, texto en cualquier punto de la misma, añadir una leyenda y también modificar características de la representación. Una forma sencilla consiste en activar la opción "**Enable Plot Editing**" en el menú "**Tools**", lo que va a permitir modificar características de forma interactiva.

Así, mediante un doble click sobre el trazo de la figura o sobre los puntos que la forman se despliega el menú de *edición de línea*, donde se pueden cambiar características como el símbolo para representar los puntos, tipo de trazado, ...

Con un doble click sobre cualquier otro punto se entra en el menú de *edición de ejes*, que permite modificar los títulos de la figura y los ejes, límites y tipo de éstos (lineal o logarítmico) o incluir una rejilla (*grid*) en la figura.

Desde el menú "Tools" es posible también añadir una leyenda a la figura, cuyo contenido de texto puede cambiarse con doble click, así como su posición en la figura (arrastrándola).

Finalmente, se disponde de los botones de Añadir texto, flechas o líneas, que permiten modificar la figura según las preferencias del usuario: para utilizar estas opciones hay que seleccionar el botón correspondiente y hacer un click sobre la figura en el punto donde se desee añadir alguno de estos gráficos. En caso de "Añadir texto" esta acción presenta un cursor que permite añadir el texto que se desee. En los otros casos, este punto marcará el punto origen de la flecha/línea, siendo necesario marcar el punto destino de la flecha con otro click en el lugar donde se desee. Las características del trazo pueden editarse posteriormente mediante un doble click sobre el trazo de la flecha/línea.

***Cómo copiar/guardar una representación gráfica.***

Al igual que cualquier otra ventana Windows, la figura puede "copiarse  y pegarse" a otra aplicación de Windows, como por ejemplo Word, Paint, ... Para ello basta seleccionar la opción "Edit"+"Copy Figure" en el menú. De esta manera se realiza una copia de la figura en un buffer intermedio, y desde ahí puede llevarse a cualquier otra ventana de cualquier otra aplicación Windows, sin más que seleccionar desde ésta la opción "Pegar".

Las opciones de Guardar ("Save" y "Save as...") de la ventana de una figura permiten guardar la figura en dos formatos propios de MATLAB: el formato FIG y el formato M. Cuando seleccionamos la opción de guardar, se puede escoger uno u otro formato en el menú desplegable que aparece en la parte inferior de la ventana.

* Si guardamos una figura en el formato "fig" únicamente podremos recuperarla posteriormente desde una ventana de figura de Matlab. Para hacerlo, debemos seleccionar la opción "File+New+Figure" en el menú de la ventana de comandos. Desde ahí basta abrir la figura a través del menú "File+Open". También se puede hacer de forma más directa pulsando el botón de ABRIR (el icono de la carpeta abierta) en la ventana de comandos de MATLAB.

* Por su parte, el formato "programa de Matlab" (M-file) guarda una secuencia de órdenes Matlab (un programa) que permite volver a generar la figura de una forma un poco más cómoda. De este modo, si guardamos una figura en este formato en un fichero de nombre "figura1", lo que se hace realmente es almacenar un fichero que contiene un programa Matlab de nombre "figura1.m". Como consecuencia, dispondremos de un "nuevo comando" Matlab de nombre "figura1". Si invocamos dicho comando desde la ventana de comandos Matlab:

>> figura1

se volverá a generar dicha figura, siempre y cuando hayamos guardado la figura en la carpeta actual. Si no fue así, tendremos que cambiar la carpeta actual para que pase a ser aquella donde hemos guardado la figura en formato M.

En realidad, cuando se guarda una figura en este formato, Matlab realiza dos tareas: por un lado, guardar el programa (fichero .m), y por otro lado, también guarda la figura en formato .fig. Podéis comprobarlo usando el explorador de Windows y viendo que realmente existen esos dos ficheros.

|  |
| --- |
| **¡¡OJO!!**: CUANDO SE GUARDAN FIGURAS EN FORMATO .M , HAY QUE TENER ESPECIAL CUIDADO EN QUE LA CARPETA DE DESTINO SEA LA CARPETA ACTUAL ***("CURRENT DIRECTORY")***, Y QUE ADEMÁS, ÉSTA SEA UNA DE LAS VUESTRAS. PARA CAMBIAR LA CARPETA ACTUAL PODÉIS ACCEDER A LA OPCIÓN **"FILE"+"SET PATH"** DEL MENÚ DE LA VENTANA DE COMANDOS DE MATLAB. |

  ***Ejercicio 4***   
Añadir título a la figura, a los ejes y modificar las características (color, línea de trazo) de la figura a) del ejercicio 1. Guardarla en formato "**fig"** en una de vuestras carpetas, con el nombre "figura1". Hacer lo mismo con la figura d) de dicho ejercicio, y guardarla en formato "fichero M", con el nombre "figura2", también en una de vuestras carpetas. Moved esos ficheros a un disquete, comprobando que los ficheros han desaparecido de vuestra carpeta (si no fue así, borradlos una vez copiados en el disquete), y cerrad todas las ventanas de figuras que tengáis abiertas. Ahora se trata de recuperarlas desde el disquete de la forma que se indicó anteriormente. Poned especial atención en el caso de "figura2", y en particular al tema de la carpeta actual.

1. ***Función ezplot***

Una manera simple de obtener la representación de una función *f*(*x*) es utilizando la función **ezplot**. Su sintaxis es:

**ezplot(f, xmin, xmax)**

donde **f** es una variable de tipo carácter que define de la función y

**xmin** y **xmax** son dos variables numéricas que indican el intervalo de representación.

Así, por ejemplo, la representación en el intervalo [0, 2· ] de la función cos(*x*) se puede realizar, usando **ezplot**, con las instrucciones siguientes:

1. Definimos la variable carácter, también llamada simbólica (fijaos en las comillas simples):

**f=’cos(t)’;**

NOTA IMPORTANTE: Aquí, la variable "t" en la definición de la función es simplemente un carácter (también llamada variable simbólica), que se puede sustituir por cualquier otro. No es, por tanto, un vector numérico que deba ser definido con anterioridad, como en el caso de utilizar **plot**. Esta es la diferencia más importante de esta función, que no requiere definir vectores numéricos X e Y.

2. Definimos los límites numéricos de representación:

**xmin=0;**

**xmax=2\*pi;**

3. Ejecutamos la función **ezplot**

**ezplot(f, xmin, xmax)**

Otra diferencia importante es que en las expresiones simbólicas, los operadores \*, /, ^ tienen significado convencional, y no matricial. Es decir, que cuando usemos ezplot no debemos utilizar el operador ".", ya que no estamos operando con vectores numéricos. Tenedlo en cuenta en el siguiente ejercicio:

* ***Ejercicio 5***

Utilizar ezplot para representar en el intervalo [0,4 ] las funciones siguientes:

tan(x)

sin2(x)-log(x)

e-3·x cos(6·x)

nombrando convenientemente los ejes y la figura. Recordad que la forma de definir las operaciones y de usar las funciones matemáticas es siempre la misma, tanto si se trata de expresiones simbólicas como estas como si son numéricas (que eran las que usamos con la función plot). Lo único que cambia ahora es que los operadores ya no tienen significado matricial, pero nada más.     
    
Al finalizar este ejercicio, puedes pasar al siguiente guión pulsando en la flecha:  [I:\ARCHIVOS MATLAB\Archivos HTML\Archivos HTML-2\Manual_Matlab\Graficos3  MATLAB_archivos\flecha.gif](http://www-gsi.dec.usc.es/~alberto/lsc/matlab/matrices.html)  

**LENGUAJE Y SISTEMAS DE COMPUTACIÓN**

**1º Ingeniería Química. Curso 2001/02**

**MATLAB: Guión de Prácticas (vi)**

**Representación gráfica en 3D**

La representación de funciones 3D ( es decir, funciones *f*: **R**2 **R**) se puede realizar en MATLAB de forma directa, utilizando alguna de las utilidades predefinidas en dicho lenguaje. Para ello, en cualquier caso, es necesario que, previamente, se definan todos los puntos (*x*, *y*) sobre los cuales va a tomar valores la función *f*. Esta tarea se realiza discretizando, en primer lugar, los ejes *x* e *y* que definen el plano y, a partir de esta discretización inicial, se calculan todos los pares de puntos (*x, y*) utilizando la función ***meshgrid***.

Esta función devuelve dos pares de matrices en las cuales se almacenan, respectivamente, las coordenadas *x* e *y* de los puntos de **R**2 en que se va a representar la función *f*.

***Ejemplo***

Si deseamos representar una función *f*:[0,2]x[0,2] **R** será necesario, en primer lugar, definir los ejes *x* e *y* correspondientes a dichos intervalos, con la discretización deseada:

**>>x=0:0.5:2;**

**>>y=0:0.5:2;**

de modo que cada eje se supone discretizado en 5 puntos. Seguidamente, se construyen todos los pares (*x*, *y*) directamente utilizando

**>>[X Y]=meshgrid(x, y);**

Las matrices *X* e *Y* así generadas almacenan las coordenadas de los 25 puntos en que se discretiza el plano [0,2]x[0,2], de modo que en *X* se guardan las coordenadas *x* de los puntos, y en *Y* las coordenadas *y*.

El siguiente paso consiste en calcular los valores de la función para cada uno de los puntos (*x*, *y*), lo cual se hace directamente manejando las matrices *X* e *Y* como si fuesen variables.

Así, si la función es escribiríamos

**>>Z=exp(X.^2+Y.^2);**

(ojo con los operadores ".", porque tanto *X* como *Y* son matrices numéricas)

de modo que, ahora, en las matrices *X*, *Y* y *Z* tenemos ya guardadas las coordenadas 3D de los puntos que queremos representar. Estas matrices son las que se utilizan con las funciones predefinidas para representación 3D en MATLAB. Por ejemplo, puede probarse la función

**>> mesh(X,Y,Z)**

En este ejemplo, la representación no es muy precisa, ya que la discretización escogida para los ejes (0.5) no es demasiado fina.

***Ejercicio 10***

1. Repetir la representación del ejemplo para una discretización de los ejes de 100 puntos. Editar la figura para nombrar los ejes con las etiquetas *x*, *y*, *f*(*x,y)* y titular la representación "*f(x,y)=e^(x^2+y^2)*"
2. Representar en el intervalo [-2, 2] la función utilizando la función ***mesh***. Repetir la representación con la función **surf(X,Y,Z)**. Utilizar la ayuda de MATLAB para analizar las diferencias entre ambas funciones.
3. Representar las funciones siguientes (ojo, como siempre, con el significado de los operadores, ya que estamos manipulando matrices numéricas):

en [- ,  ]

en [- /2,  /2]

***Otras funciones de representación***

En todos los casos de representación gráfica (y esto incluye a las representaciones 2D) es posible modificar el "punto de vista" del observador de la gráfica. En la versión 5.3 de MATLAB esto puede hacerse de forma interactiva utilizando el ratón. Para ello basta activar el botón 3D y establecer con el ratón el nuevo punto de vista de la gráfica. Los parámetros que definen el "punto de vista" son la **elevación** y el **azimut**, que indican, respectivamente, la altura sobre el eje z y el ángulo respecto del plano XY en que se sitúa el observador.

Además de las funciones ya descritas, es posible, en ambos casos, añadir a la representación gráfica las curvas de nivel correspondientes a la función que se representa. Para ello se utilizan las funciones **meshc** y **surfc** en lugar de **mesh** y **surf**. Estas curvas de nivel se proyectan sobre el plano XY automáticamente.

***Ejercicio 11***

Repetir las representaciones gráficas del ejercicio anterior utilizando **meshc** y **surfc.** Modificar el punto de vista de las gráficas hasta ver sus proyecciones sobre los planos XY, XZ e YZ.

Una vez finalizados los ejercicios de representación gráfica con MATLAB, puedes pasar al último de uso de MATLAB como herramienta de cálculo .