

4.2 Graficación 3-D en Matlab

Esta clase de graficación se refiere al hecho de poder representar en un entorno gráfico a las funciones matemáticas que están definidas en dos variables, por ejemplo, $z = f(x, y)$.

Se cuenta con dos modalidades de gráficas 3-D: en primer lugar, la graficación tridimensional de una curva ó gráfica en x , y y z ; y en segundo lugar, la graficación tridimensional de superficies y contornos bidimensionales.

Matlab provee una serie de varias funciones para la evaluación y el graficado de funciones de dos variables.

4.2.1 Graficación 3-D de una curva en x , y y z

Para la obtención de este tipo de gráficas, se cuenta con la versión tridimensional del comando *plot*:

```
[93] >>plot3 ( x , y , z );
ó
>>plot3 ( x , y , z , 'características' );
```

En donde, si x , y y z son tres vectores de igual longitud, el comando *plot3* graficará una curva en el espacio tridimensional usando como coordenadas a los valores de los elementos de x , y y z .

Si x , y y z son matrices del mismo tamaño, se graficarán varias curvas, **obtenidas de los valores de las columnas** de x , y y z .

Para el comando *plot3* se cuenta con todas las opciones disponibles para las 'características'; tanto como para el tipo de línea, así como para la selección de color que ya se utilizaron anteriormente en el comando *plot*, además de poder usar todas las demás opciones de título, leyendas, texto y retícula disponibles en las gráficas XY.

También, es posible colocar un rótulo para el nuevo *eje z* de la gráfica tridimensional con:

```
[94] >>zlabel ( 'texto' );
ó
>>zlabel ( [ 'texto' , 'prop1' , valor1 , ... ] );
```

El rótulo del eje z se especifica en '*texto*'. La(s) pareja(s) de '*prop_n*' y '*valor_n*' especifican los **atributos** que tendrá el rótulo y que funcionan igual que en el comando *title*, estudiado previamente.

El comando *axis* también se aplica a la nueva versión de *plot3*, de la forma:

```
axis ( [ x_min , x_max , y_min , y_max , z_min , z_max ] )
```

Y podemos agregar otras opciones de *axis*:

```
>>axis ( 'auto' );
```

Esta opción de *axis* sirve para volver a colocar a la escala de ejes en *automático*, es decir, para que se **autoajusten** a los valores mínimos y máximos de x , y y z .

```
>>axis ('ij');
```

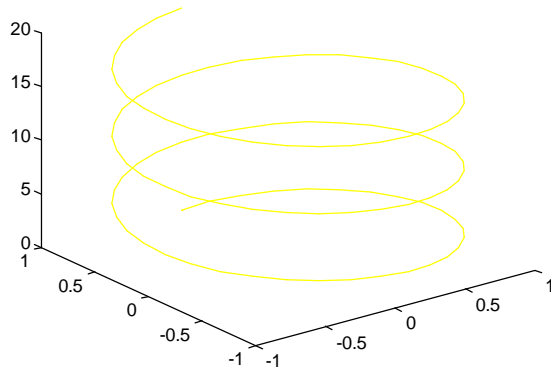
Esta opción de *axis* sirve para **invertir** la dirección de las ordenadas en el *eje y*. A esta forma de establecer los ejes se le llama “modo matricial”.

```
>>axis ('xy');
```

Esta forma de *axis* coloca a los ejes en su “modo cartesiano”; es decir, con la dirección habitual del *eje y*.

Veamos el siguiente ejemplo de *plot3*:

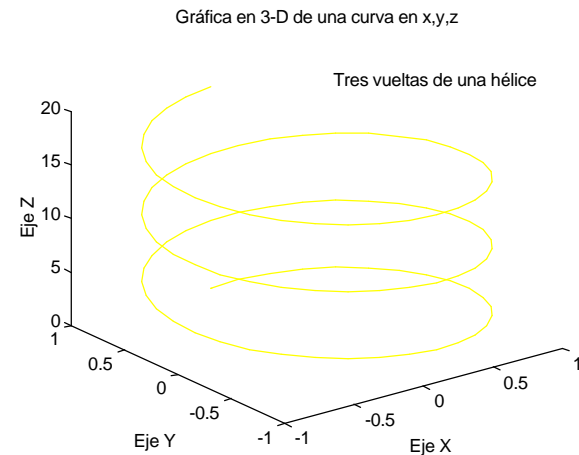
```
>>% preparamos los vectores
>>% de igual tamaño
>>z = linspace ( 0, 6*pi );
>>x = sin ( z );
>>y = cos ( z );
>>%graficamos
>>plot3 ( x, y, z );
```



Podemos agregar rótulos a los ejes, un título y texto a la gráfica mediante:

```
>>xlabel ('Eje X' );
>>ylabel ('Eje Y' );
>>zlabel ('Eje Z' );
>>title ('Gráfica en 3-D de una curva en x,y,z' );
>>text ( .5, .5, 20, 'Tres vueltas de una hélice' );
```

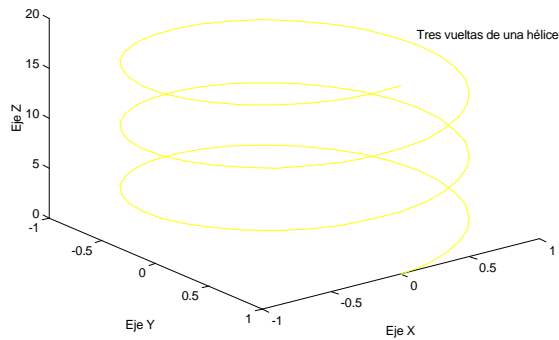
Para obtener:



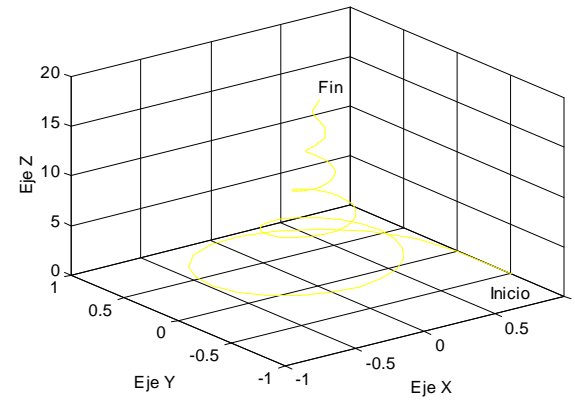
Observe que en la instrucción *text* anterior, se agregó la coordenada para el *eje z* (valor 20) que no se había usado previamente.

Si ahora aplicamos la opción de *axis ij* para invertir el sentido del *eje y* y se observa el siguiente efecto sobre la gráfica:

Gráfica en 3-D de una curva en x,y,z



Gráfica de una Curva en 3-D

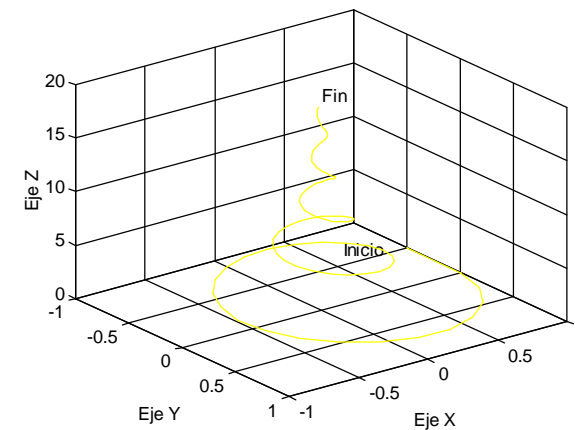


Veamos otro ejemplo:

```
>>clear
>>z = linspace ( 0 , 6*pi ) ;
>>x = cos ( z * pi/2 ) .* exp ( -0.2 * z ) ;
>>y = sin ( z * pi/2 ) * exp ( -0.2 * z ) ;
>>plot3 ( x, y, z ) ;
>>plot3 ( [1,1] , [-0.5,0] , [0,0] ) ;
>>title ( 'Gráfica de una Curva en 3-D' ) ;
>> text ( .7 , -.7 , 0 , 'Inicio' ) ;
>> text ( 0 , 0 , 20 , 'Fin' ) ;
>>xlabel ( 'Eje X' ) ;
>>ylabel ( 'Eje Y' ) ;
>>zlabel ( 'Eje Z' ) ;
>>grid on
```

Y con la opción de axis('ij') :

Gráfica de una Curva en 3-D



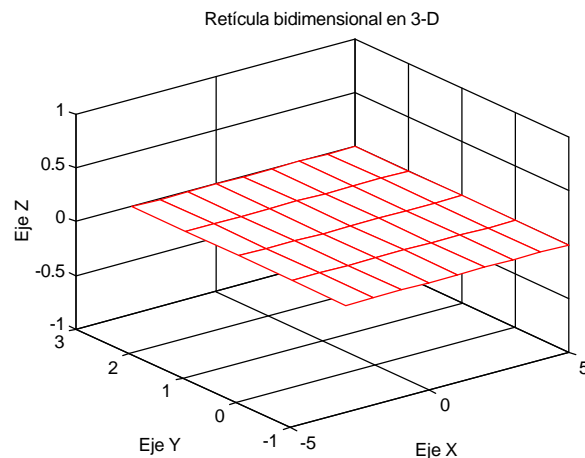
4.2.2 Gráficas 3-D de malla y de superficie

Para poder graficar una función en dos variables del tipo de $z = f(x, y)$, primero es necesario crear una especie de malla ó **retícula bidimensional** en el plano XY. En segundo lugar, evaluamos la función z en los puntos de dicha retícula para determinar puntos en la **superficie tridimensional** que se define.

En Matlab, una retícula bidimensional en el plano XY queda definida cuando se utilizan dos matrices. Una matriz contendrá todas las coordenadas x de todos los puntos de la retícula y de igual forma, la otra contendrá las coordenadas y de todos los puntos de dicha malla.

Por ejemplo, supongamos que se tienen dos vectores x y y , cuyos valores son los siguientes: $x = -3 : 5$ y $y = -1 : 3$.

La retícula bidimensional que podríamos construir con dichos vectores es:



Los valores de las matrices, digamos, X y Y necesarias para crear dicha retícula tendrían los valores:

$$X = \begin{pmatrix} -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 & 5 \\ -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 & 5 \\ -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 & 5 \\ -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 & 5 \\ -3 & -2 & -1 & 0 & 1 & 2 & 3 & 4 & 5 \end{pmatrix}$$

$$Y = \begin{pmatrix} -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 & 2 \\ 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 & 3 \end{pmatrix}$$

Recuerde que Matlab cuando grafica matrices de igual tamaño, lo hace dibujando líneas que se obtienen de los valores de sus columnas. Entonces, las matrices anteriores, tomadas columna a columna, representan las coordenadas de todos los puntos en las intersecciones de las líneas de la retícula bidimensional mostrada.

Así pues, el punto de la esquina inferior de la retícula tiene coordenadas $(-3, -1)$ y el punto de la esquina superior tiene las coordenadas $(5, 3)$.

También se puede observar que toda la retícula está colocada de forma horizontal en un espacio tridimensional; es decir, todas las coordenadas de sus puntos tienen un valor de 0 para el eje z .

Por lo que, las coordenadas correctas de los puntos mencionados serían: $(-3, -1, 0)$ y $(5, 3, 0)$.

La generación de las matrices que constituyen una retícula bidimensional puede ser hecha en forma automática con el comando:

```
[95] [ X Y ] = meshgrid ( x, y );
```

En donde x y y son dos vectores de tamaño m y n , respectivamente. Y el comando proporciona las matrices X y Y donde ambas tendrán una dimensión de $n \times m$.

La matriz X contiene los valores de x repetidos en cada fila, y la matriz Y contiene los valores de y repetidos en cada columna.

Una vez que se tienen definidas las matrices de la retícula bidimensional, será posible evaluar cualquier función matemática en dos variables para $z = f(x, y)$.

Supongamos que se desea evaluar la función:

$$z = x * e^{-x^2 - y^2}$$

La instrucción correspondiente se escribe:

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

Es sumamente importante notar que la obtención de Z se hace utilizando las matrices X y Y; de la retícula bidimensional, no los vectores x y y originales.

Esto es un error que se comete frecuentemente al estarse familiarizando con la forma de generar gráficas tridimensionales.

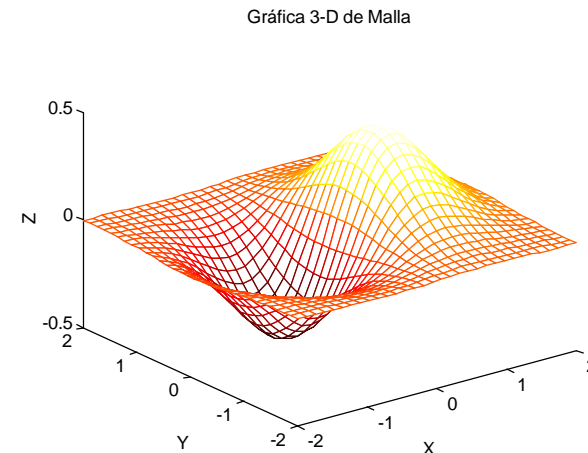
Una vez que ya se tiene evaluada la función en Z , podemos obtener una **Gráfica de Malla** con el comando:

```
[96] >>mesh ( X, Y, Z );
```

Por ejemplo, el listado,

```
>>clear
>>x = linspace ( -2 , 2 , 32 );
>>y = linspace ( -2 , 2 , 32 );
>>[ X Y ] = meshgrid ( x , y );
>>Z = X .* exp ( - X .^ 2 - Y .^ 2 );
>>mesh ( X , Y , Z );
>>title ( 'Gráfica 3-D de Malla' );
>>xlabel ( 'X' ); ylabel ( 'Y' ); zlabel ( 'Z' );
```

Despliega la gráfica:



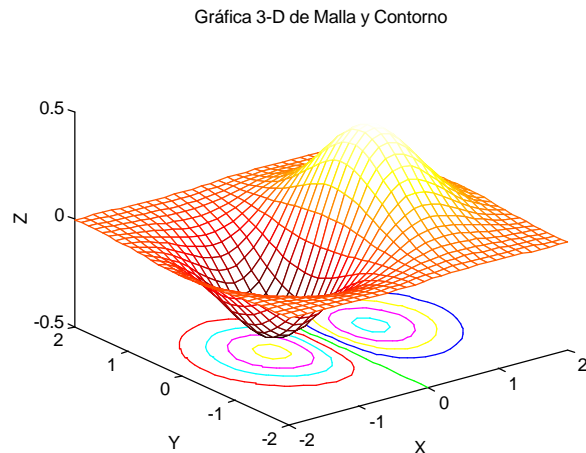
Una variante de la gráfica de malla es la llamada **Gráfica de Malla y Contorno** con el comando:

```
[97] >>meshc ( X, Y, Z );
```

El listado,

```
>>clear
>>x = linspace ( -2 , 2 , 32 );
>>y = linspace ( -2 , 2 , 32 );
>>[ X Y ] = meshgrid ( x , y );
>>Z = X .* exp ( - X .^ 2 - Y .^ 2 );
>>meshc ( X , Y , Z );
>>title ( 'Gráfica 3-D de Malla' );
>>xlabel ( 'X' ); ylabel ( 'Y' ); zlabel ( 'Z' );
```

Proporciona:



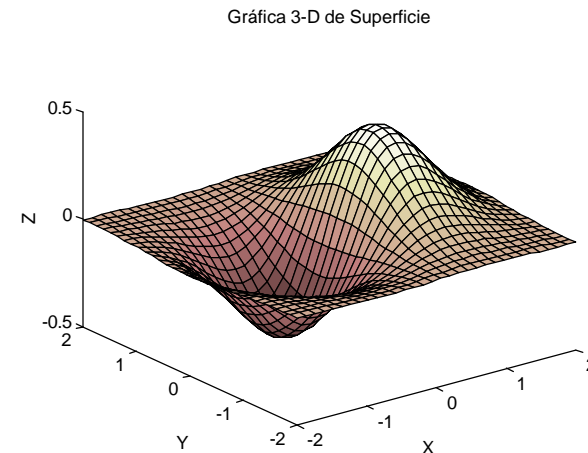
Otra de las opciones de graficación en 3-D con que se cuenta es la **Gráfica de Superficie** usando:

```
[98] >>surf ( X, Y, Z );
```

Con las instrucciones,

```
>>clear
>>x = linspace ( -2 , 2 , 32 );
>>y = linspace ( -2 , 2 , 32 );
>>[ X Y ] = meshgrid ( x , y );
>>Z = X .* exp ( - X .^ 2 - Y .^ 2 );
>>surf ( X , Y , Z );
>>title ( 'Gráfica 3-D de Superficie' );
>>xlabel ( 'X' ); ylabel ( 'Y' ); zlabel ( 'Z' );
```

Se despliega:



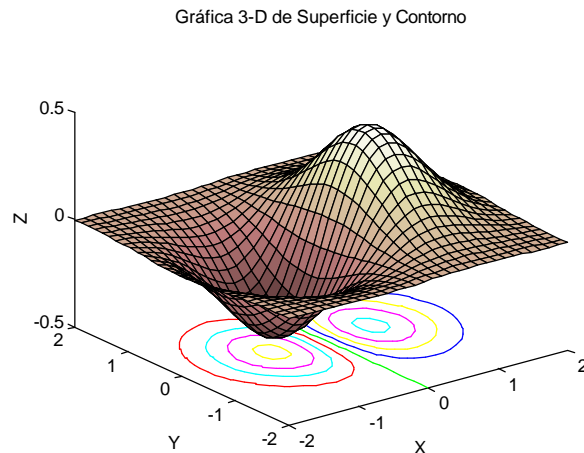
De una forma muy similar podemos obtener una **Gráfica de Superficie y Contorno** con la instrucción:

```
[99] >>surf( X, Y, Z );
```

Con los comandos,

```
>>clear
>>x = linspace ( -2 , 2 , 32 );
>>y = linspace ( -2 , 2 , 32 );
>>[ X Y ] = meshgrid ( x , y );
>>Z = X .* exp ( - X .^ 2 - Y .^ 2 );
>>surf ( X , Y , Z );
>>title ( 'Gráfica 3-D de Superficie y Contorno' );
>>xlabel ( 'X' ); ylabel ( 'Y' ); zlabel ( 'Z' );
```

Obtenemos:



Todas las gráficas construidas con las instrucciones: mesh, meshc, surf y surfc tienen la característica de que son presentadas en color de acuerdo a lo que se llama un **Mapa de Color**.

Dicho mapa de color hace que a medida que aumenten (o disminuyan) los valores en el eje z, el modo en que se despliega la gráfica vaya cambiando de acuerdo con un cierto gradiente de color relacionado con la altura alcanzada.

El usuario tiene la oportunidad de crear sus propios mapas de color. Para construirlos, habrá que especificar una matriz que puede tener un número indefinido de renglones pero necesariamente tres columnas. Los renglones de la matriz de un mapa de color contendrán los valores correspondientes a los tres colores primarios: **r** – red (rojo), **g** – green (verde) y **b** – blue (azul) de la forma:

$$\begin{bmatrix} \mathbf{r} & \mathbf{g} & \mathbf{b} \\ \cdot & \cdot & \cdot \\ \cdot & \text{etc.} & \cdot \end{bmatrix}$$

Cada renglón representa un color que estará definido por los valores de la cantidad de rojo, verde y azul que se especifiquen en cada columna en el rango desde 0.0 hasta 1.0. Por ejemplo:

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ .5 & .5 & .5 \\ .49 & 1 & .83 \end{bmatrix} \begin{array}{l} \text{(negro)} \\ \text{(blanco)} \\ \text{(gris)} \\ \text{(turquesa)} \end{array}$$

La construcción de mapas de color por parte del usuario queda fuera del alcance de este curso y sólo se menciona como información adicional. Sin embargo, para simplificar un poco las cosas, Matlab ya le ofrece al usuario algunos mapas de color predefinidos como son:

bone	hot
cool	hsv
copper	jet
flag	pink
gray	prism

Estos mapas de color son activados utilizando el comando:

```
[100] >>colormap ( 'mapa_de_color' );
```

En donde *mapa_de_color* representa el nombre del mapa de color seleccionado para ser activado, ya sea un nombre del usuario o de la lista anterior.

Con la variante,

```
>>colormap ( 'default' );
```

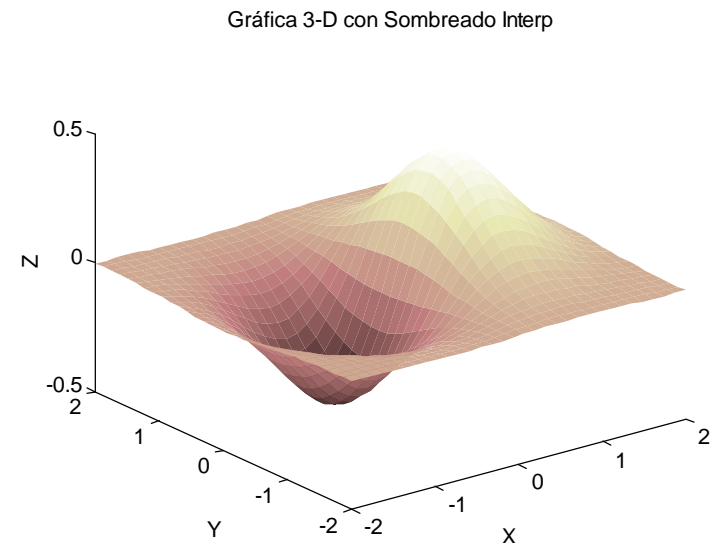
Regresamos al mapa de color predefinido en Matlab que es el *hsv*.

Hasta este momento las gráficas en 3-D se han obtenido como una malla o una superficie en un cierto mapa de color pero su apariencia no es todavía como si fuera un objeto real. Esto se logra con las opciones para establecer *sombreado*.

Podemos dar una apariencia muy real y convincente a una gráfica 3-D si se aplica alguna de las opciones para **sombreado** del comando:

```
[101] >>shading faceted ; (el de default)
ó
>>shading flat ;
ó
>>shading interp;
```

Por ejemplo, la misma gráfica que antes pero usando el sombreado *flat* e *interp*:



4.2.3 Gráficas 3-D de contorno

Un **Mapa de Contorno** se puede interpretar como un mapa que refleja diferentes elevaciones y que contiene un grupo de líneas que conectan a puntos que presentan una elevación igual con respecto a un nivel de referencia.

Un corte transversal de terreno a una elevación dada define a una línea de contorno. Si se tiene un mapa con muchas líneas que muestren diferentes elevaciones se podrán conocer las crestas y los valles de la gráfica de contorno.

Un mapa de contorno se genera a partir de los mismos datos de las gráficas tridimensionales de malla o de superficie; es decir, se requiere de una retícula bidimensional y de una matriz **Z** que refleje los valores de las elevaciones; exactamente igual que como se hizo anteriormente.

El comando que se usa para las gráficas 3-D de contorno es:

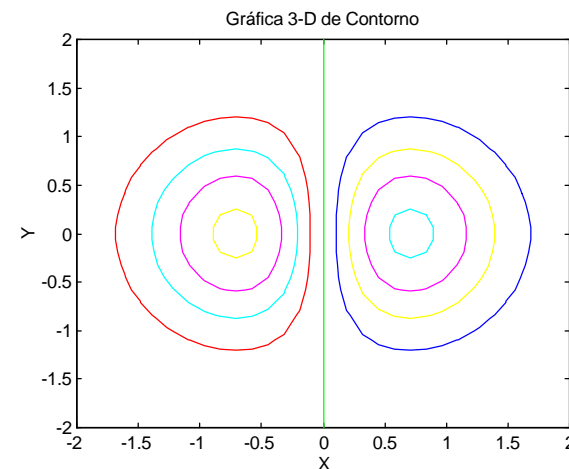
```
[102] >>contour ( X , Y , Z ) ;
ó
      >>contour ( X , Y , Z , n ) ;
ó
      >>contour ( X , Y , Z , n , 'características' ) ;
```

En la primera versión de este comando, tanto el número de líneas de contorno para la gráfica, como sus valores se escogen en forma automática.

Veamos un ejemplo del comando *contour*. Si graficamos el mismo ejemplo de función para las gráficas de malla o superficie:

```
>>clear
>>x = linspace ( -2 , 2 , 32 ) ;
>>y = linspace ( -2 , 2 , 32 ) ;
>>[ X Y ] = meshgrid ( x , y ) ;
>>Z = X .* exp ( - X .^ 2 - Y .^ 2 ) ;
>>contour ( X , Y , Z ) ;
>>title ( 'Gráfica 3-D de Contorno' ) ;
>>xlabel ( 'X' ) ; ylabel ( 'Y' ) ;
```

Produce:

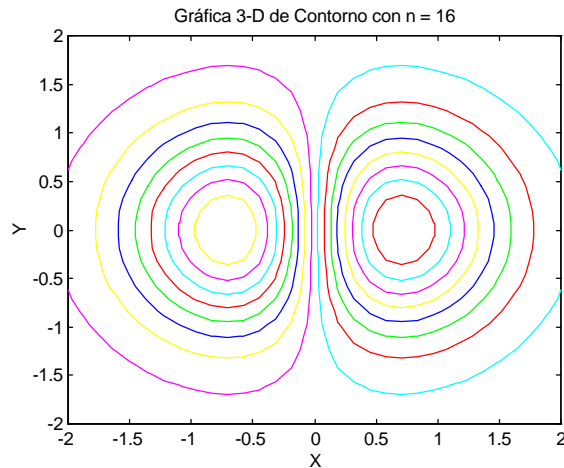


Si el número de curvas de nivel que se generaron automáticamente no fuese suficiente, podemos utilizar la segunda versión de este comando; incluyendo un número entero *n* que representa cuántas líneas de contorno se desean en total para valles y crestas.

El mismo ejemplo pero usando $n=16$ en la instrucción:

```
>>contour ( X , Y , Z , n ) ;
```

Genera:



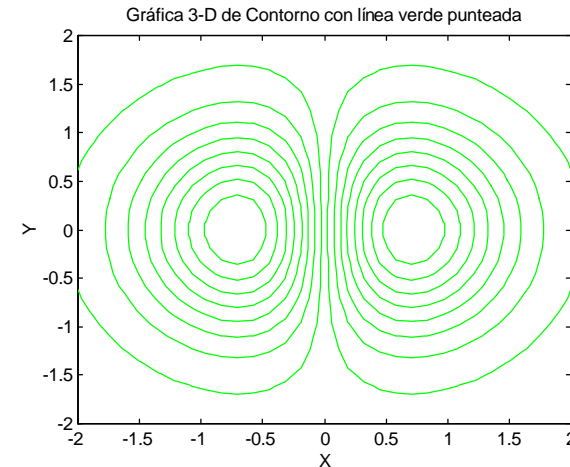
En este documento no es posible apreciar que las gráficas de contorno se están desplegando con un color distinto para cada línea indicando su elevación con respecto a un nivel de referencia.

Podemos ocupar las *'características'* a las que se refiere la última versión de este comando y que son las mismas para el tipo y color de línea que se estudiaron en el comando *plot* para generar gráficas de contorno con un tipo especial de línea ó de marca y que sea de un solo color.

Para generar el mismo contorno pero ahora en color verde y usando línea punteada:

```
>>contour ( X , Y , Z , n , 'g-.' ) ;
```

Que da por resultado:



Otra instrucción sumamente útil cuando se generan gráficas de contorno es:

```
[103] >>clabel ( CS ) ;
```

ó

```
>>clabel ( CS , 'manual' ) ;
```

Que permite colocar rótulos con el valor de las elevaciones en las líneas de contorno ó curvas de nivel que se hayan generado.

En ambos casos, **CS** se refiere a una **Contour Structure** (estructura contour) que se puede generar al ejecutar el comando *contour* con la variante:

```
>>CS = contour ( X , Y , Z , n , 'características' ) ;
```

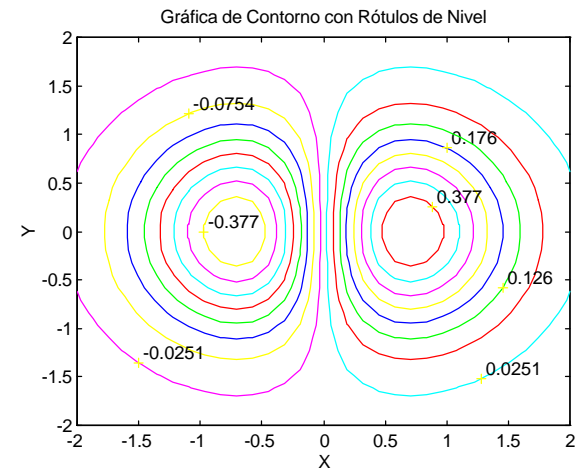
En la primera versión del comando, la colocación de los rótulos la hace Matlab de forma aleatoria en la posición que considera más adecuada.

La segunda versión de *clabel* permite que el usuario coloque los rótulos a las curvas de nivel en forma interactiva; es decir, aparece un mensaje indicando que se active al entorno gráfico y en él, se mostrará un cursor de ratón en forma de una cruz. Con dicha cruz se marcan las líneas de contorno para las que se quiera colocar su rótulo y en el lugar en donde se haga click, aparecerá el número que indica su altura.

Para terminar de colocar rótulos de alturas de nivel en las líneas de contorno se oprime la tecla [enter] y desaparece el cursor de cruz, indicando que ya se terminó el proceso.

Si después de haber suspendido la colocación de rótulos de altura se desea volver al entorno gráfico y seguir marcando algunas otras líneas, bastará con que se ejecute la segunda versión del comando nuevamente y podremos continuar. Las marcas colocadas anteriormente no se pierden.

Una gráfica de contorno con rótulos de nivel puestos por el usuario podría verse como:



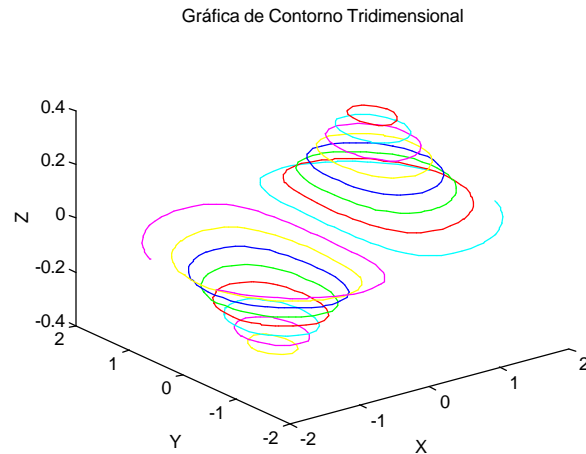
Otra instrucción relacionada con las gráficas de contorno es:

```
[104] >>contour3 ( X , Y , Z ) ;  
ó  
      >>contour3 ( X , Y , Z , n ) ;  
ó  
      >>contour3 ( X , Y , Z , n , 'características' ) ;
```

Este comando permite visualizar las gráficas de contorno pero en forma tridimensional.

Su operación es exactamente igual que el comando *contour* estudiado anteriormente. Salvo que *clabel* no funciona con este comando.

La siguiente es la gráfica de contorno tridimensional del ejemplo anterior:



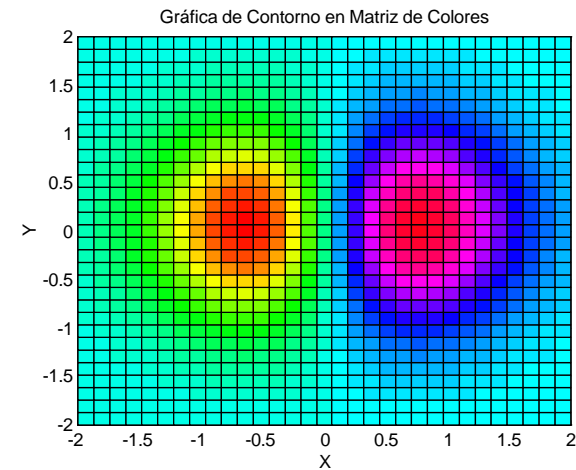
Otra forma de visualizar las gráficas de contorno es utilizando el comando:

```
[105] >>pcolor ( X , Y , Z ) ;
```

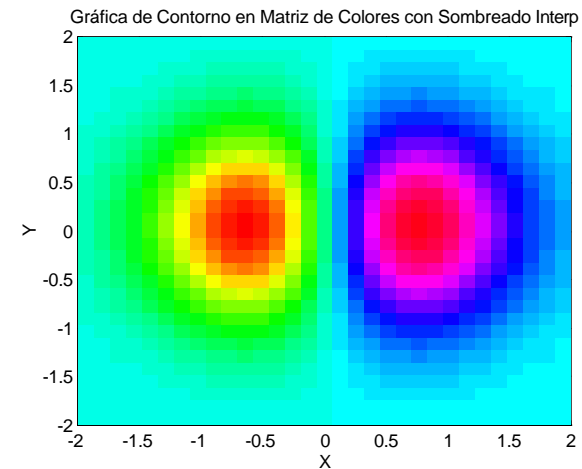
Este comando interpreta cada color de acuerdo a la altura de nivel que tengan los datos de la gráfica y la presenta en una matriz de colores.

Es el equivalente de un comando *surf* pero con un punto de vista de planta (o visto desde arriba). También se pueden aplicar las opciones de *shading* para mejorar sustancialmente su aspecto.

Veamos un ejemplo de gráfica de contorno normal, elaborada en matriz de colores:



Y ahora veamos el efecto de incluir el sombreado de tipo *interp*:



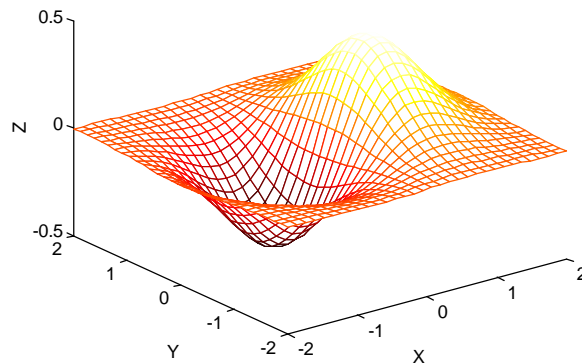
4.2.4 Cambiando el *punto de vista* (*Viewpoint*)

Una vez que se ha obtenido una gráfica de malla o de superficie, notaremos que Matlab siempre utiliza el mismo ***punto de vista* ó *viewpoint***, es decir, la posición en que se observa la gráfica siempre será la misma.

Este punto de vista tiene dos parámetros que son: el ***azimuth*** y la ***elevación***. Matlab usa por default los valores siguientes para las gráficas 3-D:

Azimuth = - 37.5 °
Elevación = 30 °

Gráfica 3-D de Malla con Punto de Vista: Az=-37.5 , El=30



El ***azimuth*** lo podemos considerar como la cantidad de giro alrededor del *eje z*, en donde los valores positivos indicarán un giro en el sentido contrario a las manecillas del reloj. El valor de azimuth 0 está colocado al frente del valor 0 del eje x.

La ***elevación*** la podemos considerar como la cantidad de altura con respecto al plano XY de la gráfica; en donde los valores positivos indicarán que estamos subiendo el punto de vista. Valores negativos indican que estamos por debajo de la gráfica.

Es muy importante percatarse que los valores tanto del ***azimuth*** como de la ***elevación*** están **definidos en grados** y no en radianes como sucede en otros comandos.

Matlab provee una instrucción con la cuál podemos cambiar el ***viewpoint*** (punto de vista) de una gráfica con el fin de poder analizarla más detalladamente. Dicha instrucción es:

```
[106] >>view ( Az , El ) ;  
ó  
>>[ Az , El ] = view ;  
ó  
>>view ( 2 ) ;  
ó  
>>view ( 3 ) ;
```

La primera forma de la instrucción nos permite cambiar el ***viewpoint*** a cualquier punto que se desee únicamente proporcionando los valores de azimuth y elevación.

La segunda opción nos permite obtener los valores del ***azimuth*** y ***elevación*** para el punto de vista de la gráfica tridimensional que se esté desplegando en el entorno gráfico actual.

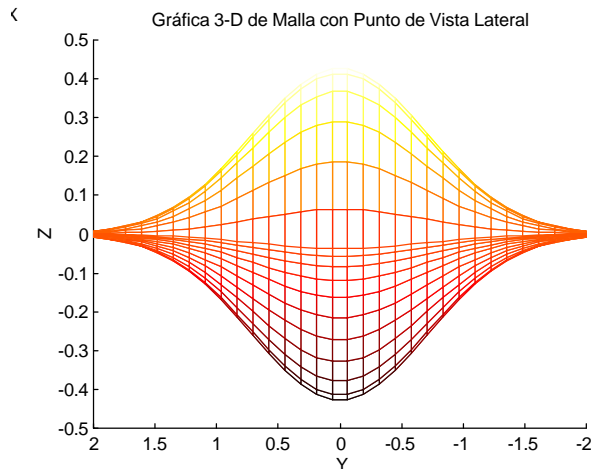
La tercera opción permite que se haga una *vista de planta* de la gráfica; es decir, desde arriba. Esto equivale a una gráfica 2-D del plano XY visto desde arriba (como si fuera un *viewpoint* con $Az = 0$, $EI = 90$).

La última opción restaura la vista tridimensional original de la gráfica; es decir, con los valores por default para *azimuth* y *elevación*. Sirve básicamente para regresar al punto de partida después de haber analizado una gráfica desde varios puntos de vista.

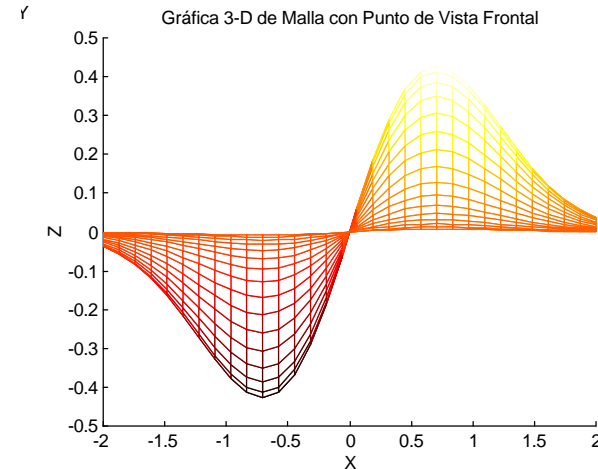
Algunos valores de *azimuth* y *elevación* para puntos comunes preferidos son:

Az=-90	EI=0	Vista lateral
Az=180	EI=0	Vista posterior
Az=0	EI=0	Vista frontal
Az=-15	EI=60	La vista del <i>ingeniero</i>

Ejemplo de vista lateral:



El punto de vista frontal es:



4.3 Exportando las gráficas

Una gráfica puede ser fácilmente exportada desde Matlab hasta otro paquete; digamos por ejemplo, Microsoft word, utilizando la opción de copy del menu edit en el entorno gráfico.

La opción más práctica cuando se va a exportar la gráfica a otro paquete es copiarla como Windows Metafile (WMF) ya que brinda algunas opciones adicionales. Si se va a llevar la gráfica a un paquete para editarla, como Paint ó Corel, es probable que convenga entonces copiarla como Windows bitmap (BMP).

En cualquier caso, al copiar una gráfica se almacenará en el *portapapeles*; el cuál podemos *pegar* en los paquetes con la secuencia de hacer *paste*: **ctrl-v**.