

Capítulo 2

Gráficos

2.1. Curvas planas

Si se desea obtener la gráfica de la función $y = y(x)$ en el intervalo $[a, b]$, debemos tener presente que Matlab dibuja las curvas punto a punto; es decir, calcula los puntos $(x, y(x))$, para los valores de x que le indiquemos y representa dichos puntos unidos por un segmento. Por ello, se empieza estableciendo la matriz fila x cuyos elementos son los valores de x para los que se computará el valor correspondiente de $y(x)$. Lo usual será tomar puntos igualmente espaciados en el intervalo $[a, b]$, incluyendo los extremos. Tomando la distancia entre dos valores consecutivos de x convenientemente pequeña, el aspecto final será el de una verdadera curva en lugar de una poligonal.

Ejemplo . *Dibujar la curva de ecuación $y = x \sin x$ en el intervalo $[-2\pi, 2\pi]$.*

```
>> x=linspace(-2*pi,2*pi,60);      % Tomamos 60 valores de x igual  
                                     % mente espaciados en  $[-2\pi, 2\pi]$ 
```

```
y=x.^ 2.*sin(x); % matriz fila con los valores de  $y(x)$ 
```

```
plot(x,y)
```

Pulsando enter, se abre una ventana gráfica con la curva. Se pueden dibujar varias curvas en la misma ventana gráfica. Si el intervalo de variación de x es el mismo, se puede proceder como se muestra en el ejemplo siguiente.

Ejemplo. *Representar gráficamente las curvas $y_1 = x^2$ e $y_2 = xe^x$ en el*

intervalo $[-3,3]$.

```
>> x=linspace(-3,3,90);  
y1=x.^ 2;y2=x.*exp(x);  
plot(x,y1,x,y2)
```

De esta forma se consigue que se abra una ventana gráfica con las dos curvas. **Otra forma de conseguir el mismo resultado consiste en usar la orden hold on. Si ya tenemos una ventana gráfica con una curva y queremos dibujar una segunda curva en la misma ventana, ponemos hold on y a continuación las órdenes necesarias para dibujar la segunda curva.**

Si el intervalo donde se quiera dibujar cada curva no es el mismo, se puede conseguir el mismo resultado de la forma siguiente. Se dibuja primero una de las curvas, se teclea hold on y acto seguido se dibuja la otra

Ejemplo. Dibujar $y = x$ en $[-1,1]$ e $y = xe^x$ en $[0,2]$.

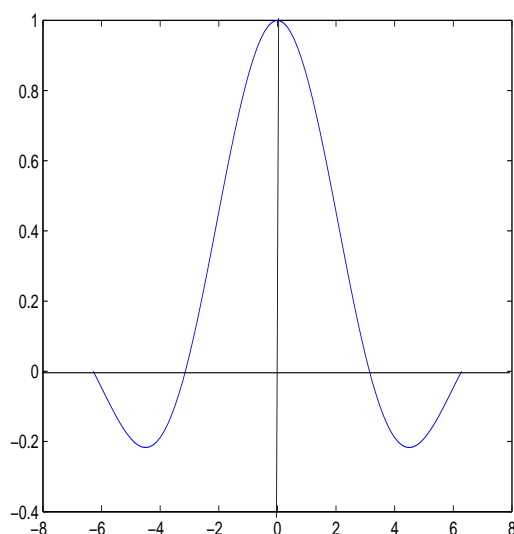
```
>> x1=-1:1:1;  
y1=x1;  
plot(x1,y1)  
hold on  
x2=0:1:2; y2=x2.*exp(x2);  
plot(x2,y2)
```

Por el contrario, cuando ya se tiene una ventana gráfica abierta y se quiere dibujar una nueva curva en otra ventana gráfica, pero sin perder la primera ventana, tecleamos figure y se abre una ventana gráfica nueva donde podremos hacer la nueva representación.

EJEMPLO. Supongamos que se desea obtener la gráfica de $f(x) = \sin(x)/x$, para $x \in [-2\pi, 2\pi]$. Se presenta el problema de que para $x = 0$ no está definida la función, aunque $\lim_{x \rightarrow 0} f(x)$ existe y vale 1. Manejando adecuadamente la variable eps, podemos evitar este problema.

```
>>x=linspace(-2*pi,2*pi,100);  
t=x+eps;  
y=sin(t)./t;
```

`plot(t,y)`
y obtenemos



Otras opciones de la función `plot`

1. Etiquetas sobre los ejes: el comando `xlabel('texto')` se usa para que en el eje OX aparezca el *texto* que se desea. Análogamente, `ylabel('texto')` para el eje OY. Para colocar cualquier cadena de texto en el punto que queramos de la ventana gráfica, se usa el comando `text(x,y,'texto')`, donde (x,y) son las coordenadas del punto donde queremos situar el centro izquierda del texto. Si usamos el comando `gtext`, entonces podemos colocar el texto donde queramos con el propio ratón.

2. Color de la curva y estilo de línea: Se puede dibujar la curva del color y con el estilo que se desee. Por ejemplo, si se escribe

`>>plot(x,y,'.',x,z,'- -')` De esta forma se consigue que la curva $y = y(x)$ aparezca con trazo punteado y la $z = z(x)$ como una línea de trazo discontinuo.

Si se quiere escoger el color y estilo:

`>> plot(x,y,'r - -')`

la curva aparece en rojo y con línea de trazo discontinuo. La siguiente tabla muestra la sintaxis de los diferentes colores y tipo de línea:

Símbolo	Color	Símbolo	Estilo de línea
y	amarillo	.	línea de puntos
m	magenta	o	círculo
r	rojo	+	más
g	verde	*	estrella
b	azul	--	trazo discontinuo
k	negro	-	línea sólida

Ejes a medida

Para fijar los valores máximo y mínimo de los ejes:

```
>>axis([xmin xmax ymin ymax])
```

Para que la escala sea la misma en ambos ejes:

```
>>axis equal o axis('equal')
```

Para que la gráfica sea un cuadrado:

```
>>axis square
```

Si se quiere que aparezca una rejilla: `grid on`.

Dibujo de poligonales

Supongamos que se desea dibujar la poligonal de vértices (x_i, y_i) con $i = 1, \dots, n$. Definiríamos las matrices fila x e y que contienen las coordenadas correspondientes y el comando `plot(x,y)` dibuja la poligonal (recordar cómo dibuja las curvas Matlab). Si la poligonal es cerrada, el último vértice ha de ser (x_1, y_1) .

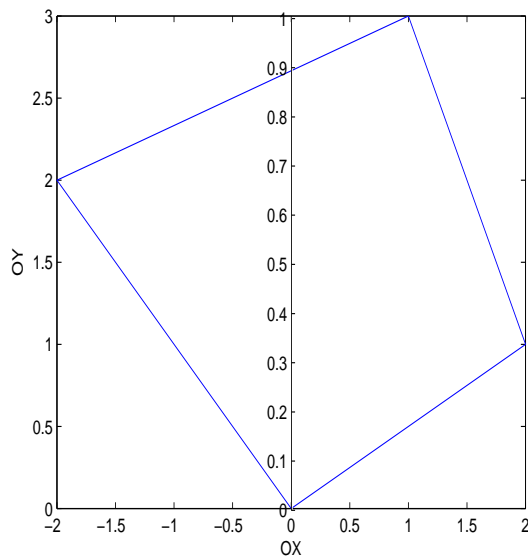
Ejemplo. *Dibujar la poligonal cerrada de vértices $(0, 0)$, $(2, 1)$, $(1, 3)$ y $(-2, 2)$.*

```
>>x=[0 2 1 -2 0];
```

```
y=[0 1 3 2 0];
```

```
plot(x,y)
```

y el resultado sería



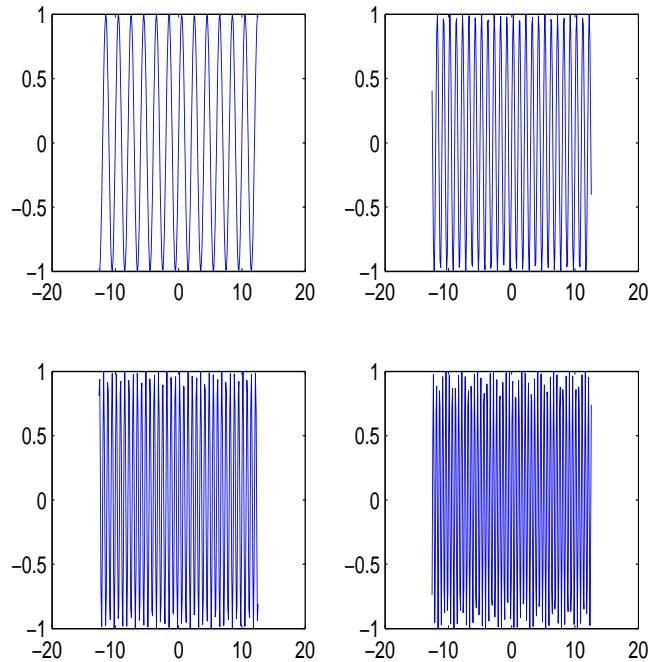
2.2. El comando subplot

A veces nos interesará disponer en una misma ventana gráfica de varias subventanas para dibujar en cada una de ellas una curva distinta, con el objetivo de poder compararlas más cómodamente. Veamos un ejemplo.

Ejemplo. Queremos dibujar en una misma ventana gráfica las curvas $y = \sin(k\pi x)$, para $k = 1, \dots, 4$ y $x \in [-4\pi, 4\pi]$, cada una en una subventana diferente.

```
>>x=linspace(-4*pi,4*pi,240);
subplot(2,2,1)
plot(x,sin(pi*x))
subplot(2,2,2)
plot(x,sin(2*pi*x))
subplot(2,2,3)
plot(x,sin(3*pi*x))
subplot(2,2,4)
plot(x,sin(4*pi*x))
```

Y se obtiene



En general, si se necesitan $m \times n$ subventanas, se tendrá en cuenta que se numeran de izquierda a derecha y de arriba hacia abajo. Cuando tecleamos `subplot(m,n,k)`, estamos indicando que vamos a dibujar en la subventana que ocupa el lugar k . En el ejemplo anterior, todas las subventanas responden a la forma `subplot(2,2,k)`, porque teníamos 4 curvas y parece lo más adecuado disponerlas en dos filas y dos columnas.

2.3. Coloreado de polígonos

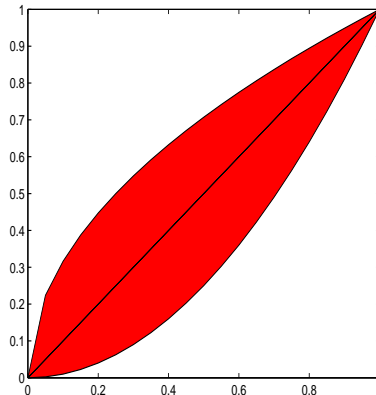
En la sección anterior hemos visto cómo se dibuja una poligonal con Matlab. Ahora vamos a ver cómo se colorea la región interior del color que queramos.

Ejemplo. Se desea dibujar las curvas $y = x^2$ y $x = y^2$ en el primer cuadrante. La región que encierran debe aparecer en color rojo.

```
>> x=0:0.1:1;  
y1=x.^2;y2=sqrt(x);  
plot(x,y1,x,y2)
```

De esta forma se han dibujado las curvas en cuestión. Para colorear de rojo la región encerrada, Matlab dispone de la función `fill`. Su sintaxis es la siguiente: `fill(x,y,'r')` (x e y son matrices fila que contienen las coordenadas x e y de los vértices de la poligonal. El programa anterior se continuaría de la forma siguiente:

```
X=[x x];  
y=[y1 y2];  
fill(X,y,'r')
```



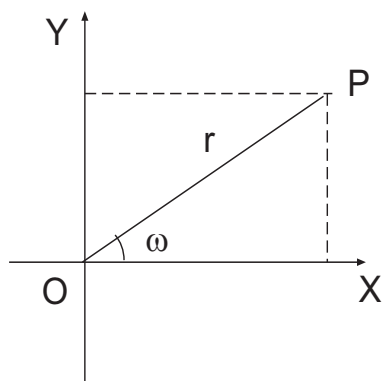
Si se desea escoger un color indicando las coordenadas (en el sistema RGB) se escribe

```
>>fill(x,y,[r g b])
```

Los números r , g y b pertenecen a $(0,1)$ y representan las proporciones en que se deben tomar los colores principales (rojo, verde y azul) para crear el color en cuestión.

2.4. Curvas en polares

Empezamos recordando cómo se relacionan las coordenadas polares con las cartesianas.



Vemos en la figura que ω es el ángulo que forman el vector de posición del punto P con la dirección positiva del eje OX y r es el módulo de dicho vector. Por un lado, el Teorema de Pitágoras nos dice que $r^2 = x^2 + y^2$ y, por otro, usando las definiciones de $\sen \omega$ y $\cos \omega$, obtenemos $x = r \cos \omega$ e $y = r \sen \omega$. Si de una curva plana sabemos que las coordenadas polares de sus puntos, (r, ω) , verifican la igualdad

$r = r(\omega)$, para $\omega \in [\omega_1, \omega_2]$, diremos que $r = r(\omega)$ es la ecuación de la curva en coordenadas polares. La ecuación de la circunferencia unidad en cartesianas es $x^2 + y^2 = 1$ y en coordenadas polares $r = 1$. En general, para obtener la ecuación en polares, conocida la ecuación de una curva en cartesianas, basta sustituir en esta última ecuación x e y por $r \cos \omega$ y $r \sen \omega$, respectivamente.

Ejemplo. *Dibujar la curva de ecuación $r = 1 + \cos \omega$, para $0 \leq \omega \leq 2\pi$.*

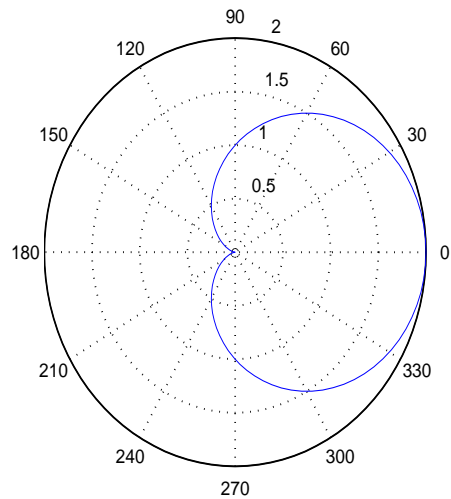
Por comodidad, vamos a usar w en lugar de ω .

```
>> w=linspace(0,2*pi,60);
```

```
r=1+cos(w);
```

```
polar(w,r)
```

pulsando enter se abre una ventana gráfica que muestra la curva siguiente (denominada cardioide).



2.5. Curvas en el espacio

Supongamos que se quiere dibujar la curva de ecuaciones paramétricas $x = \cos t$, $y = \sin t$, $z = t$, para $t \in [0, 6\pi]$. Podemos usar el comando `plot3` o el comando `ezplot3`.

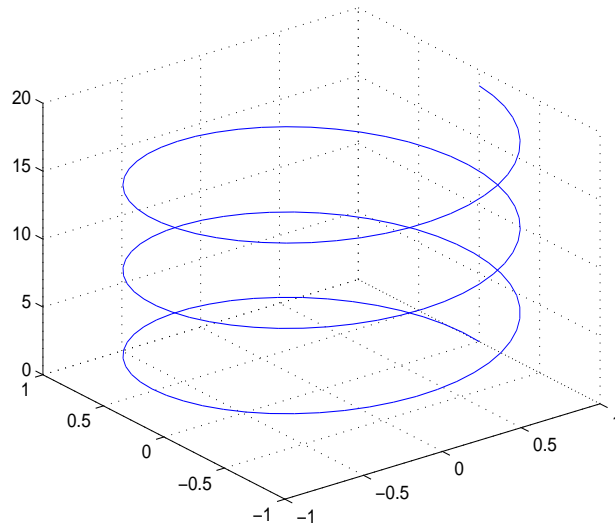
a) Con `plot3`:

```
>>t=linspace(0,6*pi,150);
```

```
plot3(cos(t),sin(t),t)
```

```
grid on
```

y el resultado es



b) Con `ezplot3`:

```
ezplot3('cos(t)','sin(t)','t', [0,6*pi])
```

Terminamos esta sección indicando cómo puede conseguirse que aparezcan los vectores tangentes al dibujar curvas en paramétricas .

Ejemplo. Representar la curva de ecuaciones paramétricas

$$x = \cos(t), y = \sin(t), t \in [0, \pi].$$

```
>>t=linspace(0,pi,30);
```

```
plot(cos(t),sin(t))
```

Si se quiere que aparezcan los vectores tangente, se usa la función **quiver**.

```
>>t=linspace(0,pi,30);
```

```
plot(cos(t),sin(t))
```

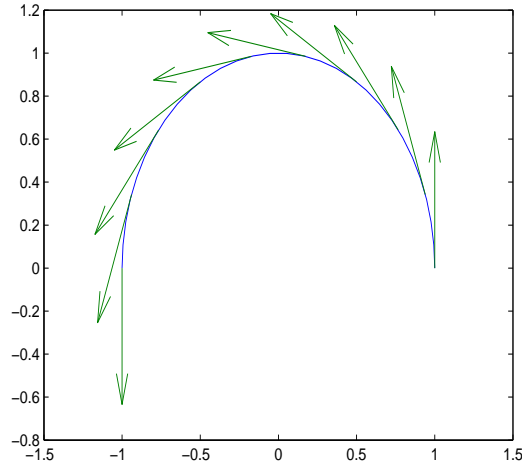
```
hold on
```

```
>> t=linspace(0,pi,10); %Dibujamos el vector tangente en sólo 10 puntos
```

```
% intermedios de la curva
```

```
quiver(cos(t),sin(t),-sin(t),cos(t))
```

y se obtiene



2.6. Superficies

Para dibujar una superficie de ecuación $z = z(x, y)$, se comienza por establecer los intervalos de variación de x e y . Con la orden `[x,y]=meshgrid(-2:.1:2;-1:.1:1)` Matlab crea una matriz x con todas sus filas iguales a `-2:.1:2` y una matriz y con todas sus columnas iguales a `-1:.1:1`. De este modo resultan dos matrices con la misma dimensión. Veamos un ejemplo simple. Consideremos la función $z = x^2 + y^2$ y escribimos la orden

```
>> [x,y]=meshgrid(-1:.5:1,0:.5:1);
```

que produce las matrices

$$x = \begin{pmatrix} -1 & -.5 & 0 & .5 & 1 \\ -1 & -.5 & 0 & .5 & 1 \\ -1 & -.5 & 0 & .5 & 1 \end{pmatrix}, \quad y = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ .5 & .5 & .5 & .5 & .5 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

De esta forma, cuando escribamos `z=x.^2+y.^2` en la ventana de comandos, z será la matriz 3×5 que contiene los valores de z en todos y cada uno de los puntos (x, y) con x igual a uno de los valores `-1,-.5,0,.5,1` e y igual a uno de los valores `0,.5,1`.

Ejemplo. Dibujar la superficie $z = \sqrt{x^2 + y^2}$ en el dominio $[-3, 3] \times [-3, 3]$

```
>> [x,y]=meshgrid(-3:.1:3,-3:.1:3);
z=sqrt(x.^2+y.^2);
surf(x,y,z)
```

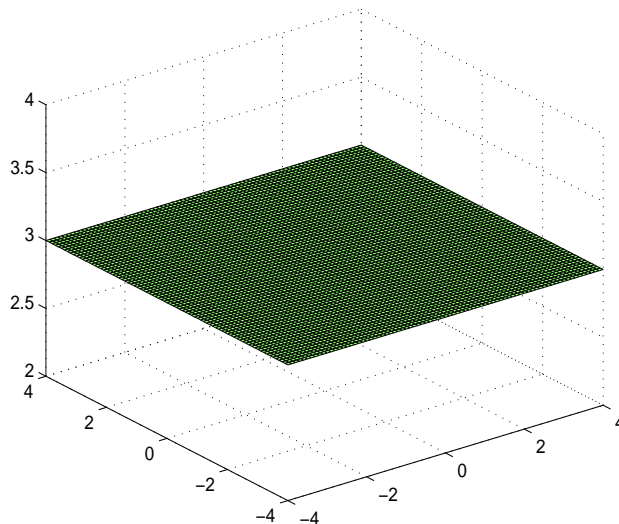
Además de la orden `surf(x,y,z)`, para dibujar una superficie, podemos emplear `plot3(x,y,z)` o `mesh(x,y,z)` (la diferencia con `surf` es que ésta rellena los espacios entre líneas).

La orden `rotate3d` permite girar la superficie con el ratón. La orden `colormap` permite cambiar el color de la superficie. Hay diversas opciones: `colormap(pink)`, `colormap(summer)`, `colormap(winter)`, etc.

En el ejemplo siguiente mostramos cómo se dibuja una superficie plana paralela a $z = 0$.

Ejemplo. Dibujar la superficie $z = 3$ en el primer octante.

```
>> [x,y]=meshgrid(-4:.1:4,-4:.1:4);
[m,n]=size(x); % Recordar lo que hemos dicho sobre el comando meshgrid,
% usamos size(x) para determinar las dimensiones de x de una manera se-
gura.
z=3*ones(m,n);
surf(x,y,z)
```



2.7. Curvas de nivel

Supongamos que hemos dibujado una superficie, $z = f(x, y)$, y queremos que en el plano $z = 0$ aparezcan dibujadas las curvas de nivel. Podemos proceder como sigue

```
>>hold on
```

```
contour(x,y,z,[c1,c2,c3,...,cn])
```

y aparecen representadas las curvas de nivel $f(x, y) = ci$, para $i = 1, \dots, n$.

Ejemplo. Representar la superficie de ecuación $z = x^2 + y^2$, para $(x, y) \in [-2, 2] \times [-2, 2]$. Además, deseamos que aparezcan representadas las curvas de nivel $f(x, y) = c$, para $c = 1, 2, 3$.

```
>>[x,y]=meshgrid(-2:.1:2,-2:.1:2);
```

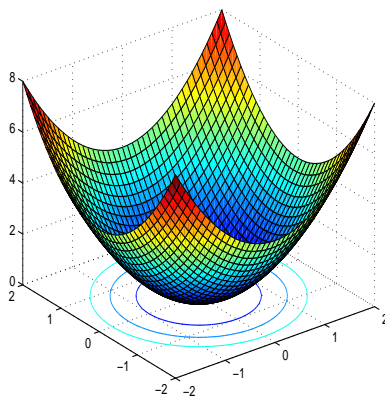
```
z=x.^2+y.^2;
```

```
surf(x,y,z)
```

```
hold on
```

```
contour(x,y,z,[1,2,3])
```

y se obtiene

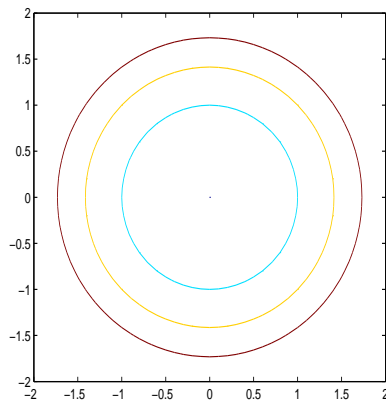


Si sólo se buscan las curvas de nivel (no se necesita dibujar la superficie), basta escribir

```
>>[x,y]=meshgrid(-2:.1:2,-2:.1:2);
```

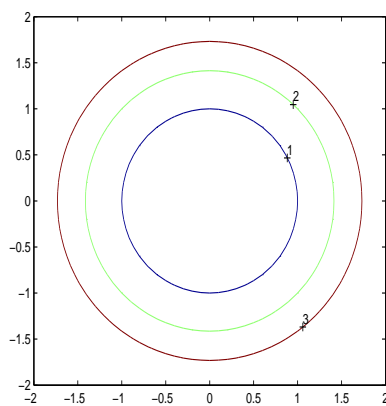
```
z=x.^2+y.^2;
```

`contour(x,y,z,[1,2,3])`
y se obtiene



Si se quiere que aparezcan etiquetas sobre cada curva de nivel que reflejen el valor de la constante c , se usa la función **clabel**

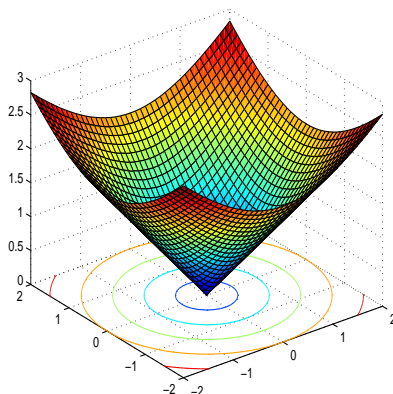
```
>>[x,y]=meshgrid(-2:.1:2,-2:.1:2);
z=x.^2+y.^2;
clabel( contour(x,y,z,[1,2,3]) )
resultando
```



Cuando necesitamos representar una superficie y deseamos que aparezcan las curvas de nivel, hay una forma muy cómoda que consiste en emplear la función **surf(x,y,z)**.

Ejemplo. Gráfica de la superficie $z = \sqrt{x^2 + y^2}$ con las correspondientes curvas de nivel.

```
>>[x,y]=meshgrid(-2:.1:2,-2:.1:2);  
z=sqrt(x.^2+y.^2);  
surf(x,y,z)  
y resulta
```



2.8. ¿Cómo exportar gráficos?

a) Word. Una vez que tenemos el gráfico en la ventana gráfica, vamos a Edit (en la ventana gráfica) y hacemos clic en Copy figure. A continuación abrimos el documento word y pegamos.

b) Latex. Primero debemos salvar el gráfico de forma adecuada. Vamos a File (en la ventana gráfica) y hacemos clic en Save as. Se abre una ventana (Save as) en la que debemos dar el nombre con que vamos a designar el archivo que contendrá el gráfico. Automáticamente, Matlab le añade la extensión que corresponda. Normalmente, elegiremos el tipo EPS file, en cuyo caso la extensión que Matlab añade es eps. Ahora es el momento de guardar el

archivo creado, nombre.eps, en la carpeta que contiene el archivo Latex donde queremos incluir el gráfico. Ahora debemos preparar el archivo Latex, lo que exige realizar los siguientes dos pasos

1) En el archivo principal y antes de la instrucción `\begin{document}`, debemos incluir

```
\usepackage{graphicx}
```

2) En el lugar donde vamos a incluir el gráfico:

```
\begin{center}
```

```
\includegraphics[width=5cm,height=6cm]{nombre.eps}
```

```
\end{center}
```