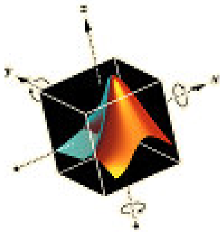
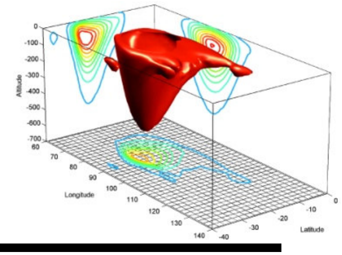




Gráficos 3D en MATLAB

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 gráficas en tres dimensiones
- Desarrollar la capacidad de generar gráficos interactivamente

Indice

- Líneas en 3D
- Superficies
- Creación de gráficos interactivamente

Líneas en 3D

- La versión 3D de `plot` es

  `plot3`(u1, v1, w1, c1, u2, v2, w2, c2,…)

  donde

  uj, *vj*, y *wj* son las coordenadas *x*, *y*, y *z*, respectivamente, de un punto

  Son escalares, vectores de la misma longitud, matrices del mismo orden, o expresiones que, cuando se evalúan, resultan en una de esas cantidades

  cj es una cadena de caracteres

  Un caracter especifica el color.

  Un caracter especifica las características del punto

  Uno o dos caracteres especifica el tipo de línea

Líneas en 3D

- Para dibujar un conjunto de *n* líneas sin conectar cuyos puntos finales son

  (x_{1j}, y_{1j}, z_{1j}) y (x_{2j}, y_{2j}, z_{2j}), $j = 1, 2, \ldots, n$

  se crean seis vectores:

$$\mathbf{x}_j = \left[\mathbf{x}_{j1} \; \mathbf{x}_{j2} \ldots \mathbf{x}_{jn} \right]$$

$$\mathbf{y}_j = \left[\mathbf{y}_{j1} \; \mathbf{y}_{j2} \ldots \mathbf{y}_{jn} \right] \qquad \mathbf{j = 1, 2}$$

$$\mathbf{z}_j = \left[\mathbf{z}_{j1} \; \mathbf{z}_{j2} \ldots \mathbf{z}_{jn} \right]$$

- Así, `plot3` es

  x1 = [...]; x2 = [...];
  y1 = [...]; y2 = [...];
  z1 = [...]; z2 = [...];
  `plot3`([x1; x2], [y1; y2], [z1; z2])

  donde [x1; x2], [y1; y2], y [z1; z2] son matrices de ($2 \times n$)

Líneas en 3D

- Todos los procedimientos de anotación descritas para los gráficos 2D son aplicables a las funciones de generación de curvas y superficies 3D, excepto que los argumentos de `text` se usa

`text`(x, y, z, s)

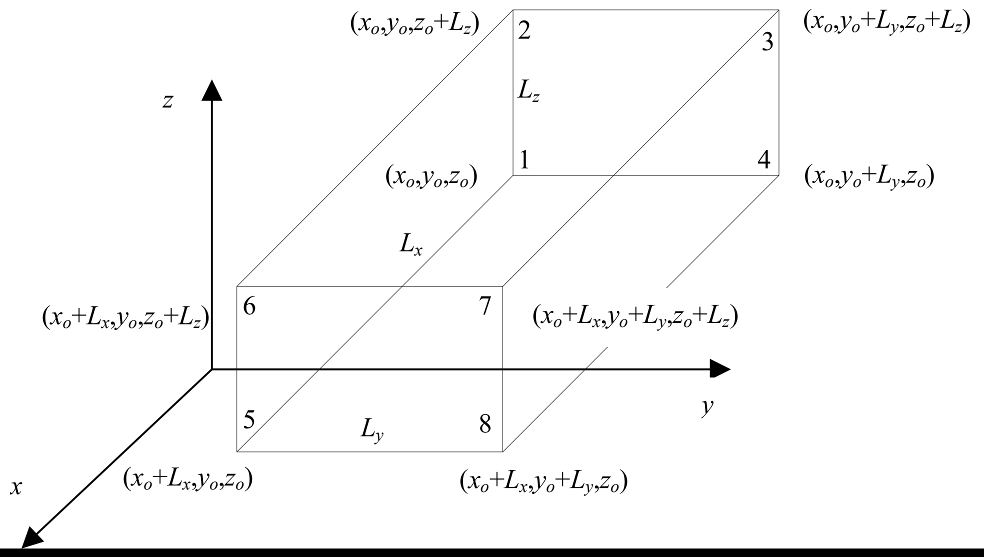
donde *s* es un string y

`zlabel`

se usa para etiquetar el eje *z*

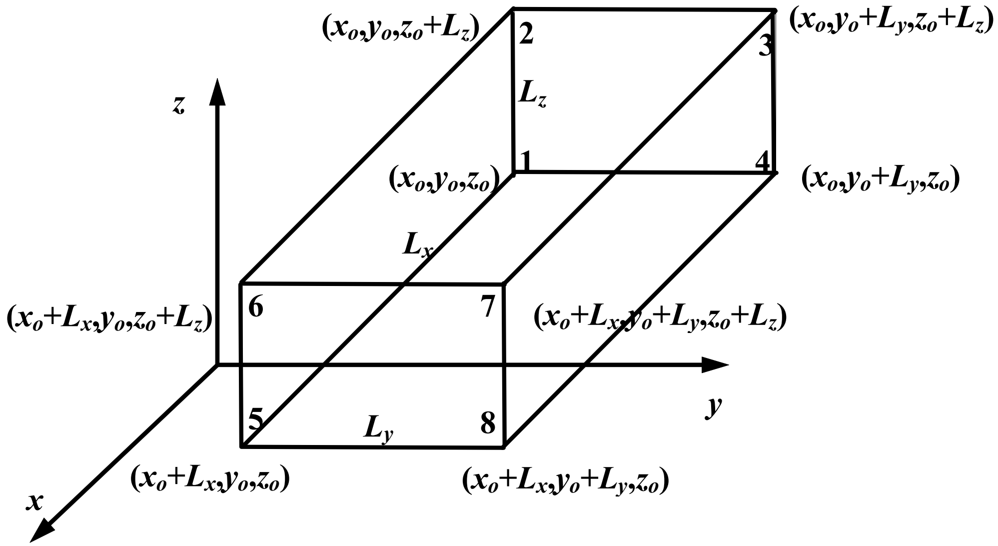
Ejemplo: Dibujo de cajas de alambres

- Se requiere una función *BoxPlot3* que dibuje las aristas (4) de cada una de las seis superficies de una caja. La ubicación y orientación de la caja está determinada por las coordenadas de la diagonal de caras opuestas $P(x_o,y_o,z_o)$ and $P(x_o+L_x, y_o+L_y, z_o+L_z)$



Ejemplo: Dibujo de cajas de alambres

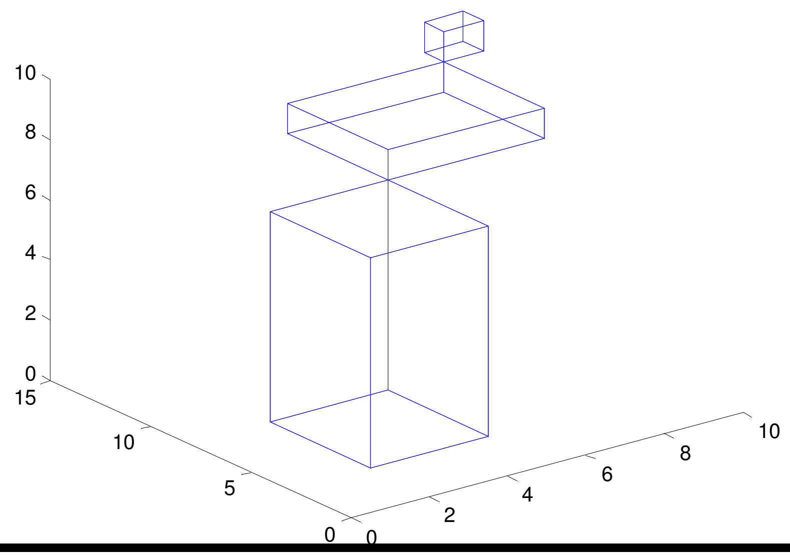
```
function BoxPlot3(x0, y0, z0, Lx, Ly, Lz)
x = [x0, x0, x0, x0, x0+Lx, x0+Lx, x0+Lx, x0+Lx];  %(1×8)
y = [y0, y0, y0+Ly, y0+Ly, y0, y0, y0+Ly, y0+Ly];   %(1×8)
z = [z0,  z0+Lz, z0+Lz, z0, z0, z0+Lz, z0+Lz, z0];    %(1×8)
index = zeros(6,5);
index(1,:) = [1 2 3 4 1];
index(2,:) = [5 6 7 8 5];
index(3,:) = [1 2 6 5 1];
index(4,:) = [4 3 7 8 4];
index(5,:) = [2 6 7 3 2];
index(6,:) = [1 5 8 4 1];
for k = 1:6
   plot3(x(index(k,:)), y(index(k,:)), z(index(k,:)))
   hold on
end
```



Ejemplo: Dibujo de cajas de alambres

- El script para generar tres cajas con las siguientes dimensiones y coordenadas (x_o, y_o, z_o)
  - Box #1
    Size: 3×5×7
    Location: (1, 1, 1)
  - Box #2
    Size: 4×5×1
    Location: (3, 4, 5)
  - Box #3
    Size: 1×1×1
    Location: (4.5, 5.5, 6)

```
BoxPlot3(1, 1, 1, 3, 5, 7)
BoxPlot3(4, 6, 8, 4, 5, 1)
BoxPlot3(8, 11, 9, 1, 1, 1)
```



Ejemplo: Onda senoidal sobre una superficie de un cilindro

- Las coordenadas de una onda senoidal sobre la superficie de un cilindro se obtiene con

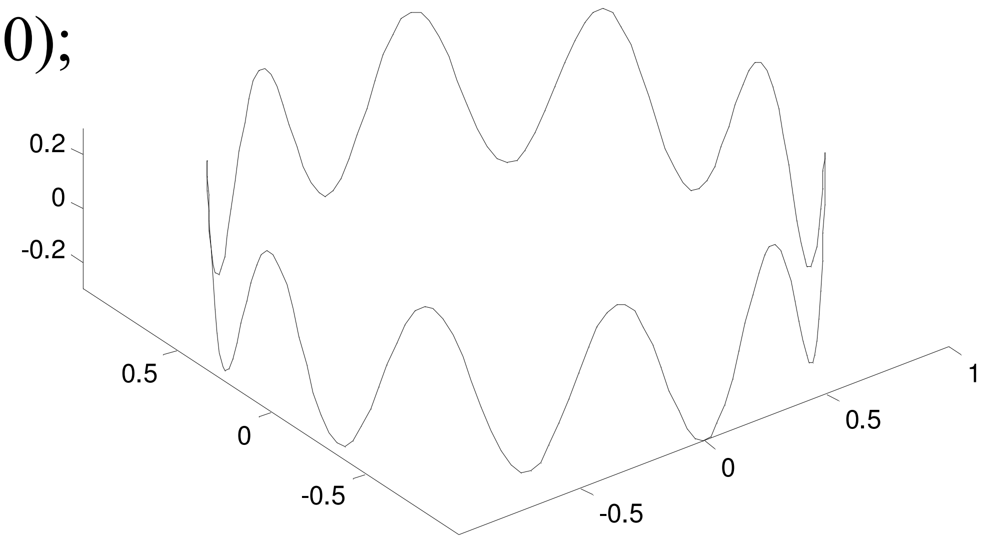
$$x = b\cos(t)$$

$$y = b\sin(t)$$

$$z = c\cos(at)$$

Si se asume que $a = 10.0$, $b = 1.0$, $c = 0.3$, y $0 \le t \le 2\pi$, el script es

```
t = linspace(0, 2*pi, 200);
a = 10;  b = 1.0;  c = 0.3;
x = b*cos(t);
y = b*sin(t);
z = c*cos(a*t);
plot3(x, y, z, 'k')
axis equal
```



Superficies

- Matlab contiene un conjunto de funciones gráficas 3D para crear superficies, contornos, y variaciones, así como especializaciones de esas formas básicas

- Una superficie se define por la expresión

$$z = f(x, y)$$

donde x e y son las coordenadas en el plano-*xy* y z es la altura resultante

Superficies

- Las funciones básicas de graficación de superficies son

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

  donde *x*, *y*, *z* son las coordenadas de los puntos en la superficie

  `surf` – dibuja una superficie compuesta de parches de colores que dependen de la magnitud *z*

  `mesh` – dibuja parches de superficies blancas que se definen por su contorno. Los colores de las líneas de los parches se determinan por la magnitud de *z*.

Ejemplo de superficie

- Se requiere dibujar una superficie definida por

$$z(x,y) = x^4 + 3x^2 + y^2 - 2x - 2y - 2x^2y + 6$$

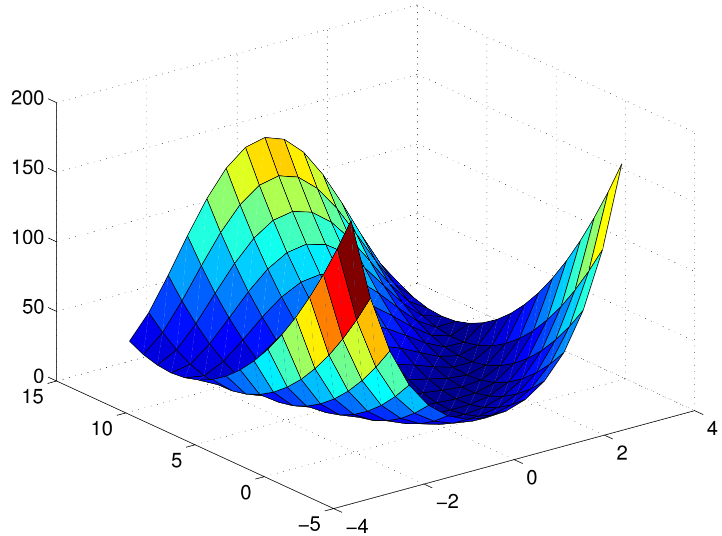
definida en el rango –3 < x < 3 y –3 < y < 13

Se genera la función *SurfExample* para calcular las coordenadas x, y ,z

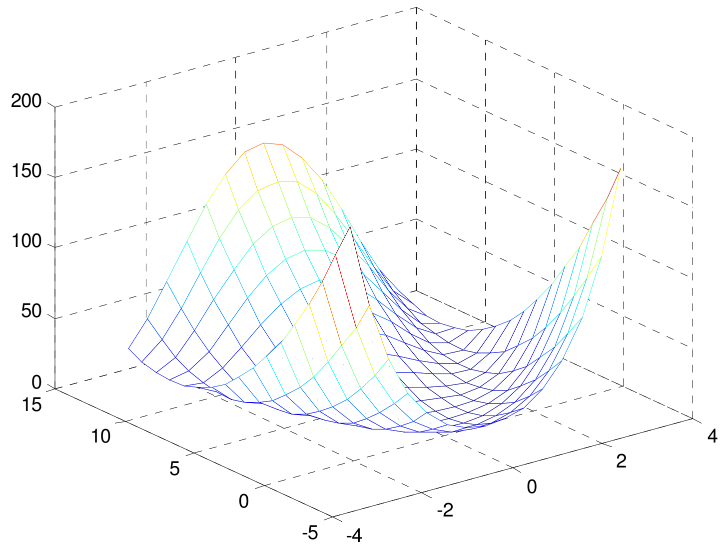
```
function [x, y, z] = SurfExample
x1 = linspace(-3, 3, 15);      % (1×15)
y1 = linspace(-3, 13, 17);     % (1×17)
[x, y] = meshgrid(x1, y1);     % (17×15)
z = x.^4+3*x.^2–2*x+6-2*y.*x.^2+y.^2-2*y;  % (17×15)
```

Ejemplos de superficies con surf y mesh

[x,y,z] = SurfExample;
`surf`(x, y, z)

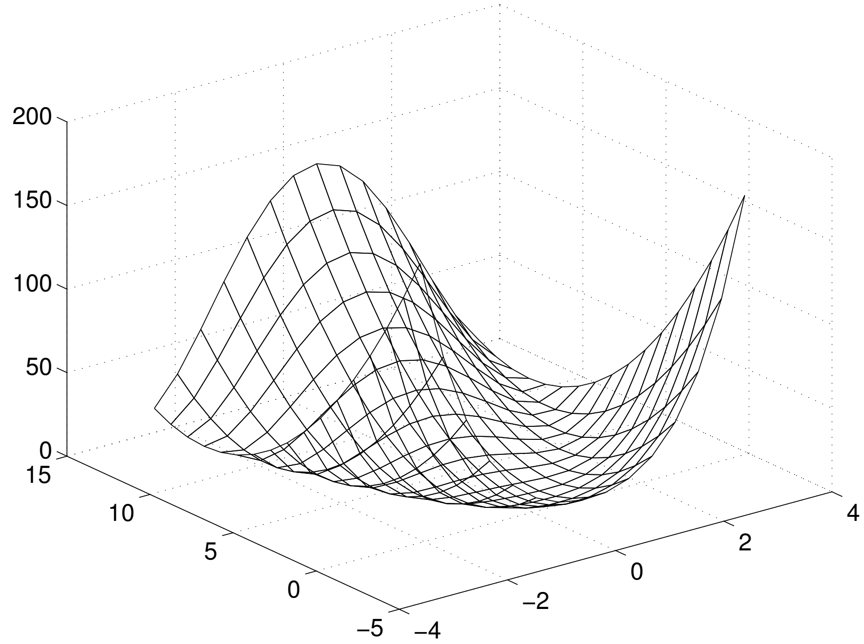


[x,y,z] = SurfExample;
`mesh`(x, y, z)



Ejemplos de superficies con surf y mesh

```
[x,y,z] = SurfExample;
mesh(x, y, z)
hidden off
```

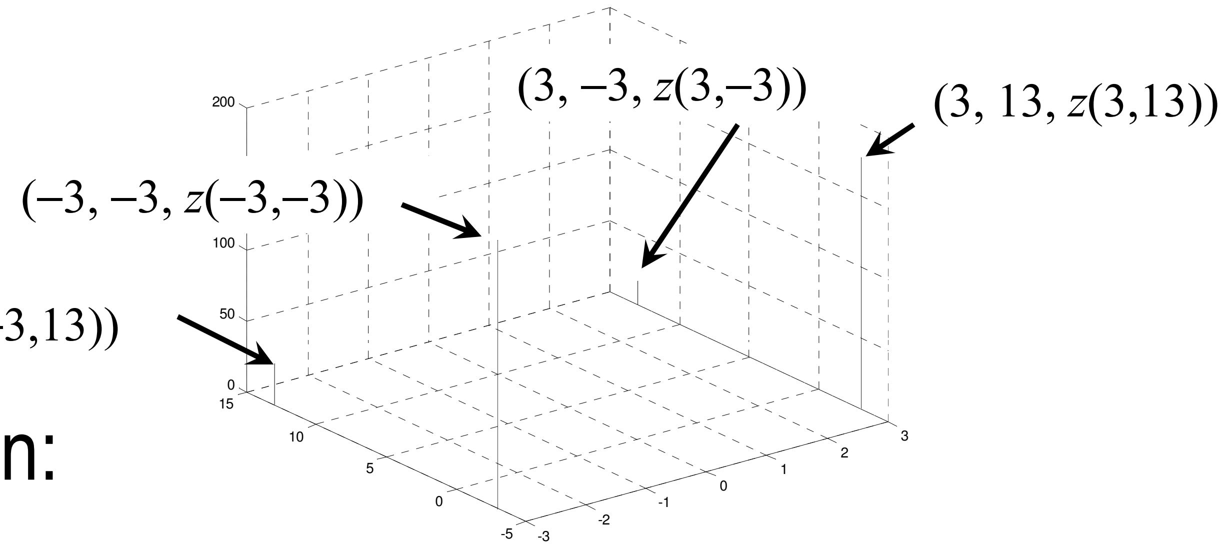


Combinando superficies y líneas

- Se puede combinar funciones de graficación 3D para dibujar múltiples líneas y superficies
- Como ejemplo se crean dos funciones
  Corners: que dibuja cuatro líneas conectando las esquinas de la superficie generada por *SurfExample* al plano *xy* que pasa por $z = 0$
  Disc: que crea un disco circular que interseca la superficie creada por *SurfExample* en $z_o = 80$,con radio de 10 unidades, y centro en (0,5)

Ejemplo: combinando superficies y líneas

- Las coordenadas de las esquinas son:

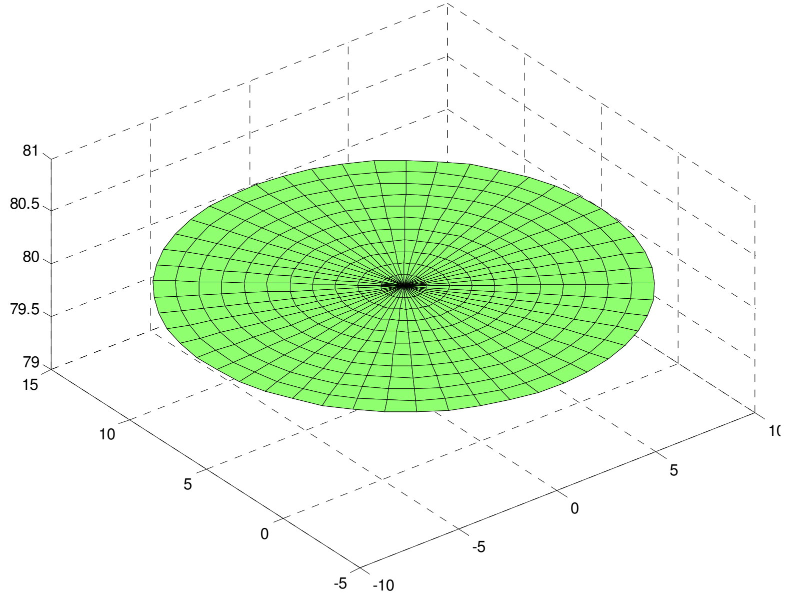


Las funciones son:

```
function Corners
xc = [-3, -3, 3, 3];
yc = [-3, 13, 13, -3];
zc = xc.^4+3*xc.^2–2*xc+6–2*yc.*xc.^2+yc.^2–2*yc;
hold on
plot3([xc; xc], [yc; yc], [zeros(1,4); zc], 'k')
```

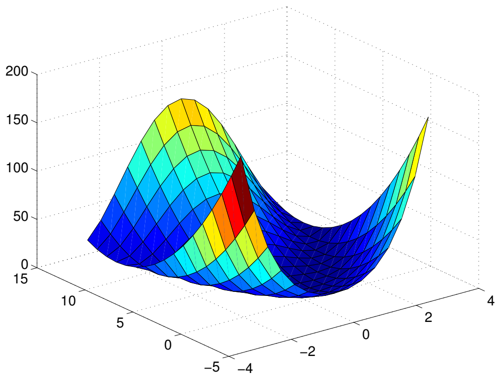
Ejemplo: combinando superficies y líneas

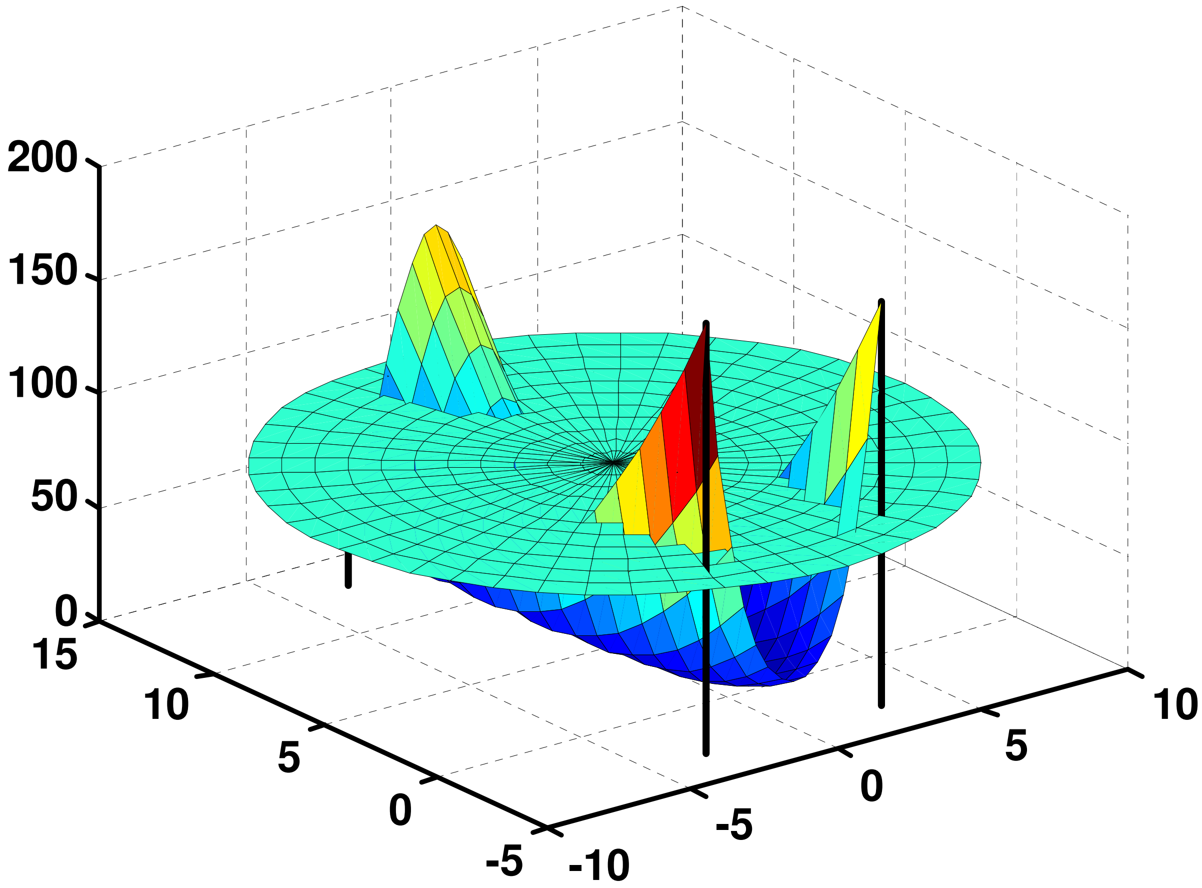
```
function Disc(R, zo)
r = linspace(0, R, 12);                  % (1×12)
theta =  linspace(0, 2*pi, 50);  % (1×50)
x = cos(theta')*r;                       % (50×12)
y = 5 + sin(theta')*r;                   % (50×12)
hold on
z = repmat(zo, size(x));  % (50×12)
surf(x, y, z)
```



Ejemplo: combinando superficies y líneas

[x, y, z] = SurfExample;
surf(x, y, z);
Disc(10, 80)
Corners





Modificación de la apariencia de gráficos

- Hay varias funciones que se pueden usar de forma combinada para modificar la apariencia de la superficie resultante

  `box on` o `box off`

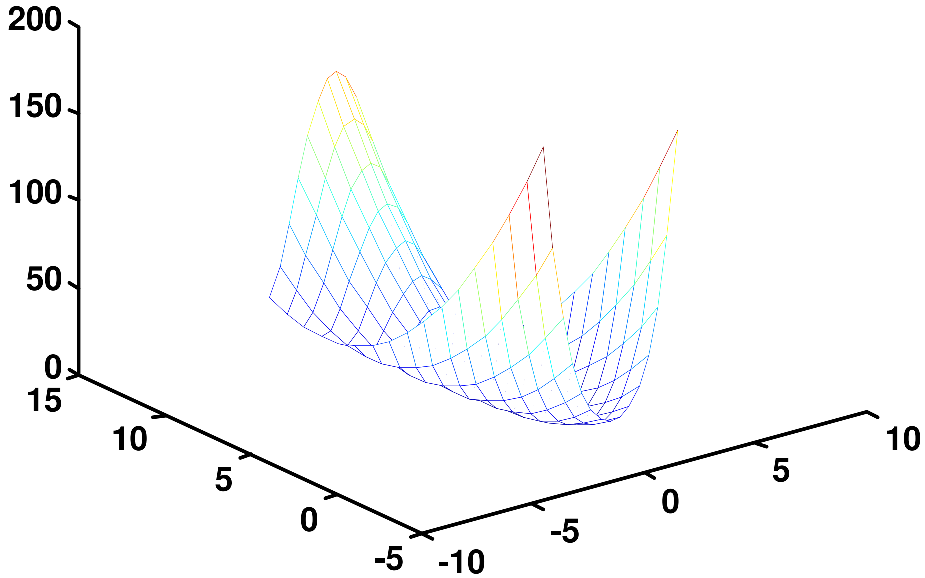
  `grid on` o `grid off`

  `axis on` o `axis off`

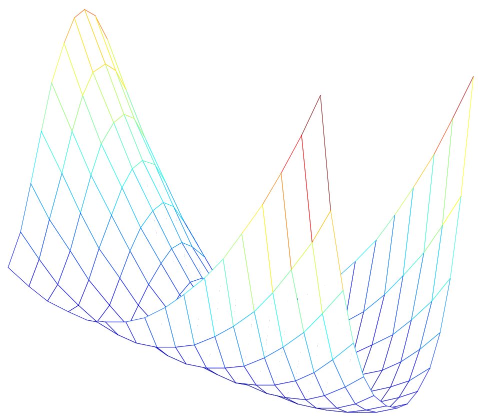
  La función `box on` sólo dibuja una caja si `axis on` ha sido seleccionada

Ejemplo: modificación de la apariencia de gráficos

```
[x,y,z] = SurfExample
mesh(x, y, z)
grid off
```

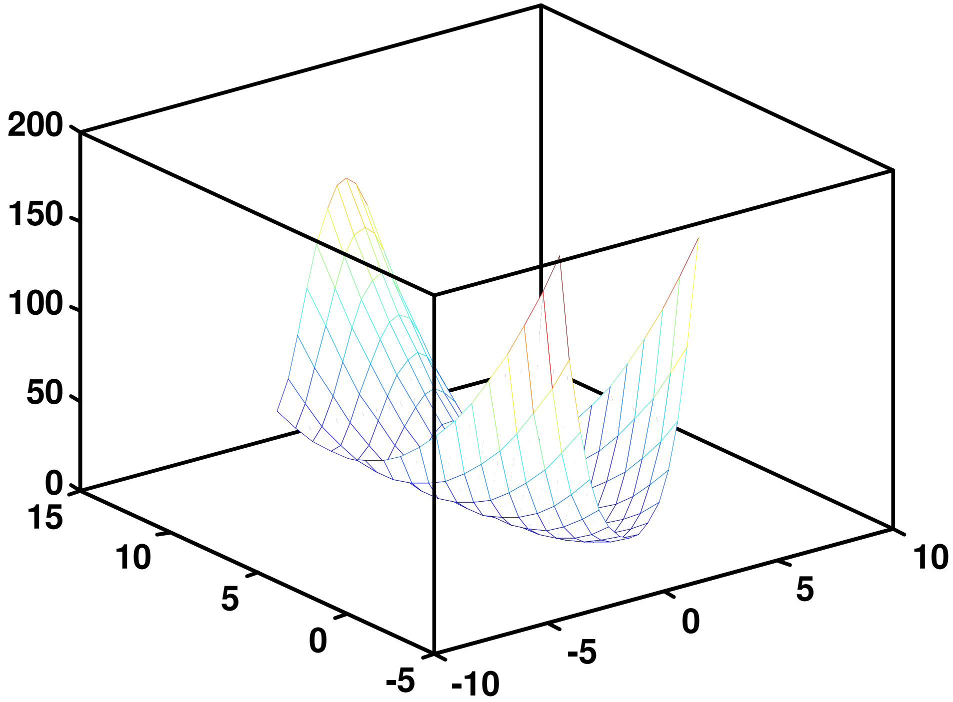



```
[x,y,z] = SurfExample
mesh(x, y, z)
axis off
grid off
```



Ejemplo: modificación de la apariencia de gráficos

```
[x,y,z] = SurfExample
mesh(x, y, z)
axis on
grid off
box on
```



Modificación de la apariencia de gráficos

- Los colores de los parches creados por surf o las líneas creadas por mesh se pueden cambiar a un color uniforme usando

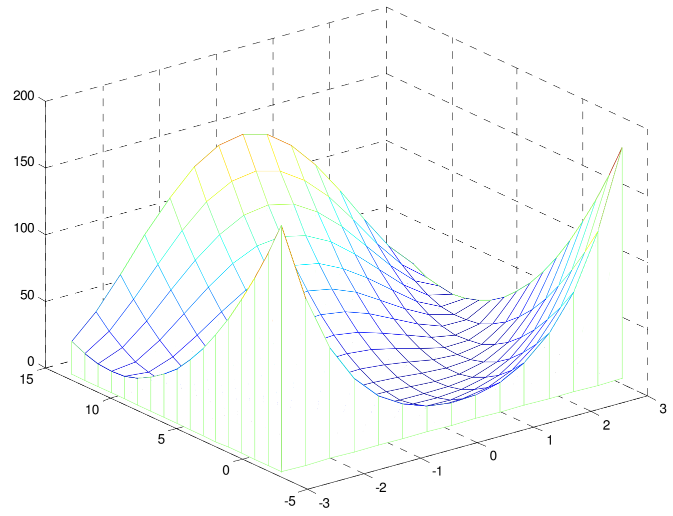
  `colormap`(c)

  donde *c* es un vector de tres elementos, cada uno de los cuales varía entre 0 y 1, correspondiendo a la intensidad del color rojo, verde y azul respectivamente (r, g, b). Ejm:

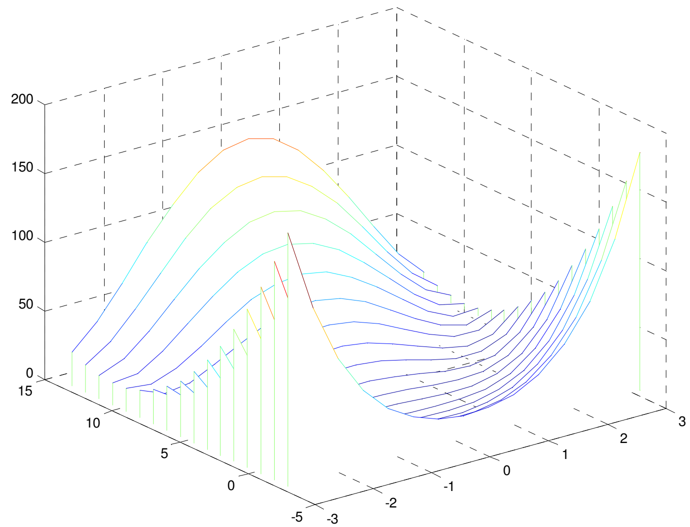
c	Color
[0 0 0]	black
[1 1 1]	white
[1 0 0]	red
[0 1 0]	green
[0 0 1]	blue
[1 1 0]	yellow
[1 0 1]	magenta
[0 1 1]	cyan
[0.5 0.5 0.5]	gray

Ejemplo: funciones adicionales para mejorar visualmente una superficie

```
[x,y,z] = SurfExample;
meshz(x, y, z)
```



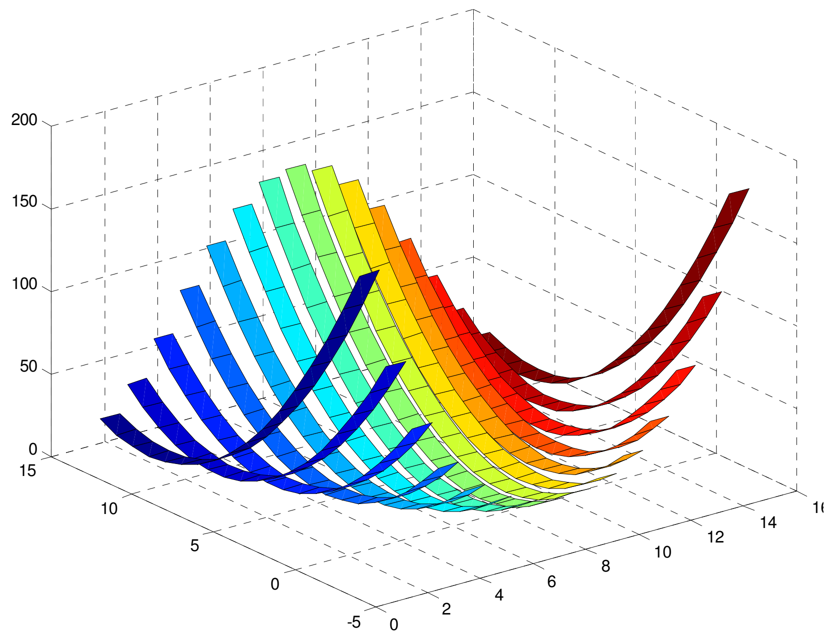

```
[x,y,z] = SurfExample;
waterfall(x, y, z)
```

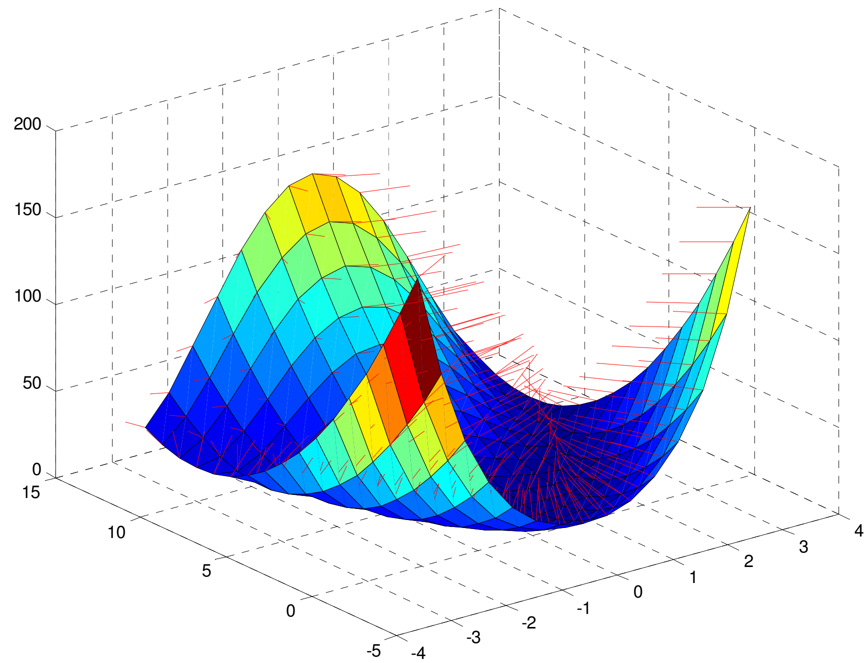


Ejemplo: funciones adicionales para mejorar visualmente una superficie

```
[x,y,z] = SurfExample;
ribbon(y, z)
```

```
[x,y,z] = SurfExample;
surfnorm(x, y, z)
```





Gráficos de contornos

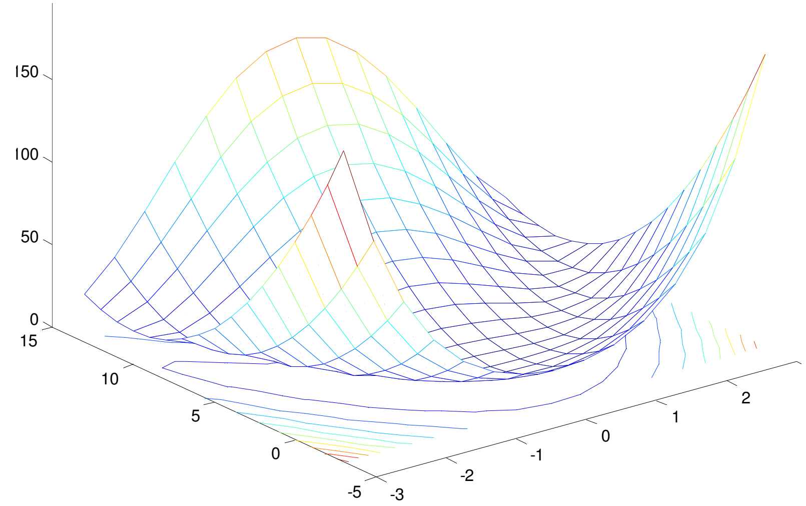
- Las superficies también se pueden transformar en gráficos de contornos, que son gráficos de curvas formadas por la intersección de la superficie y un plano paralelo al plano *xy* en valores específicos de z
- Las funciones

  `surfc`(x, y, z) y `meshc`(x, y, z)

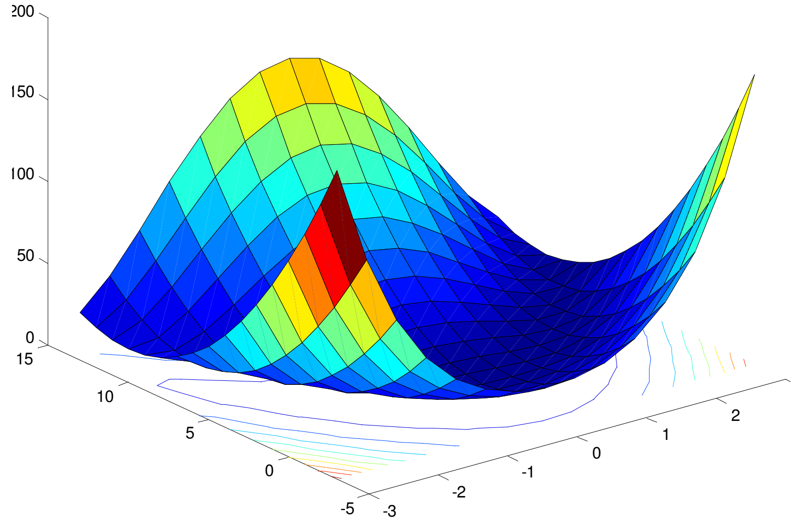
  crean superficies con contornos proyectados debajo de la superficie. *x*, *y*, *z* son los valores de las coordenadas de puntos que definen la superficie

Ejemplo de gráficos de contornos

```
[x,y,z] = SurfExample;
meshc(x, y, z)
grid off
```




```
[x,y,z] = SurfExample;
surfc(x, y, z)
grid off
```



Gráficos de contornos

- Se pueden crear contornos sin visualizar la superficie, con etiquetas o sin etiquetas
- La función

  `contour`(x, y, z, v)

  crea un gráfico de contorno donde

  x, *y*, *z* son las coordenadas de los puntos que definen la superficie

  v, si es un escalar, es el número de niveles de contornos a visualizar y, si es un vector de valores, los contornos de la superficie en los valores de *z*. El uso de *v* es opcional

Gráficos de contornos

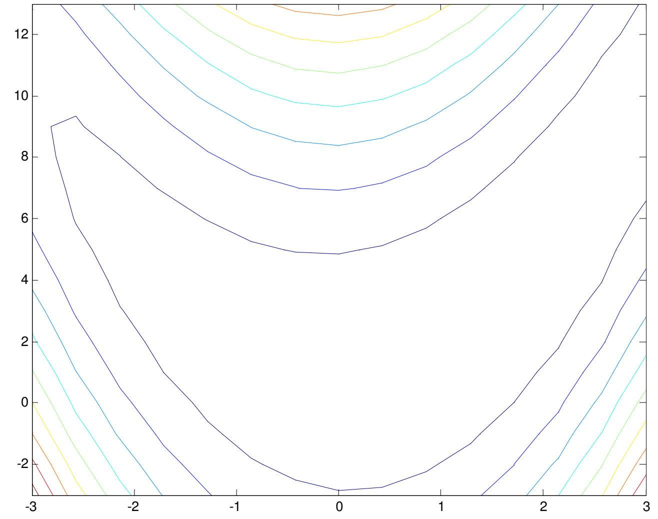
- Si se quiere etiquetar el contorno se usan las funciones

  [C, h] = `contour`(x, y, z, v)

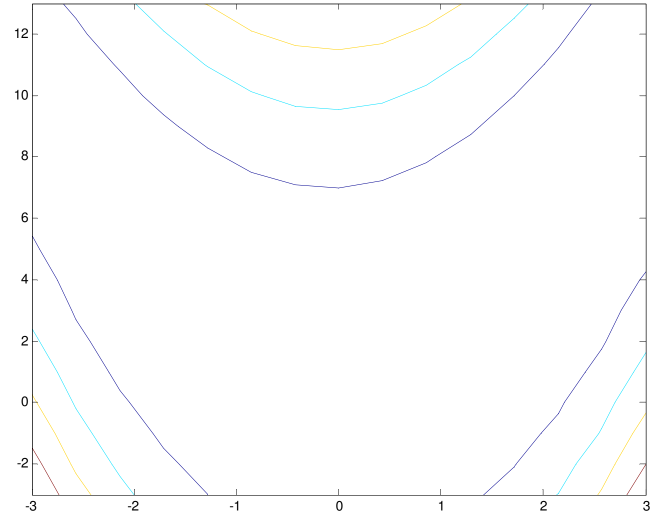
  `clabel`(C, h, v)

Ejemplos de contour

```
[x,y,z] = SurfExample;
contour(x, y, z)
```

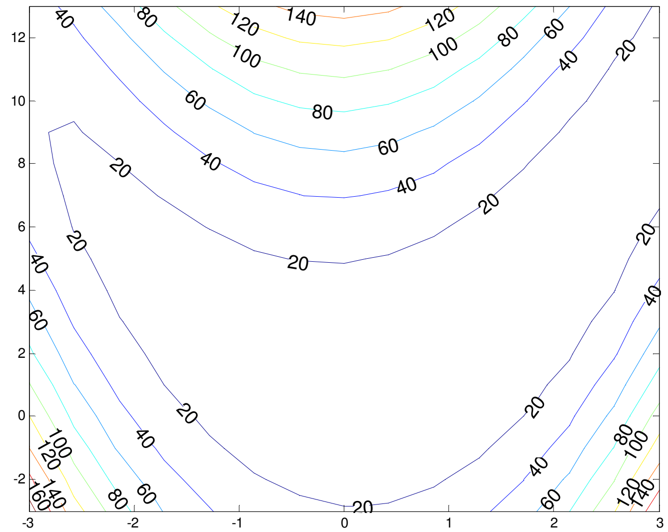



```
[x,y,z] = SurfExample;
contour(x, y, z, 4)
```

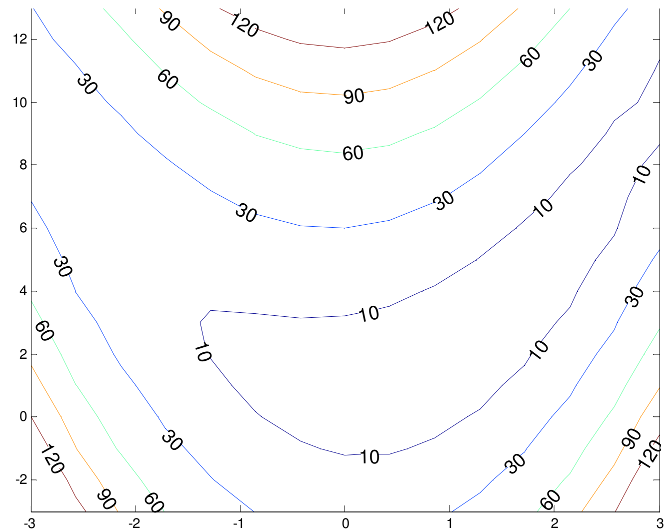


Ejemplos de contour y clabel

```
[x,y,z] = SurfExample;
[C, h] = contour(x, y, z);
clabel(C, h)
```




```
[x,y,z] = SurfExample;
v= [10, 30:30:120];
[C, h] = contour(x, y, z, v);
clabel(C, h, v)
```



Gráficos de contornos 3D

- Para obtener los contornos de superficies en 3D, se usa

  `contour3`(x, y, z, v)

  donde

  x, *y*, *z* son las coordenadas de los puntos de la superficie

  v, si es un escalar, es el número de niveles de contornos a visualizar y, si es un vector de valores, los contornos de la superficie en los valores de *z*. El uso de *v* es opcional

  Para etiquetar los contornos se usa

  [C, h] = `contour3`(x, y, z, v)

  `clabel`(C, h, v)

Gráficos de contornos 3D

- Para rellenar la region entre contornos 2D con diferentes colores se usa

  `contourf`(x, y, z, v)

  los valores de los colores se pueden identificar usando

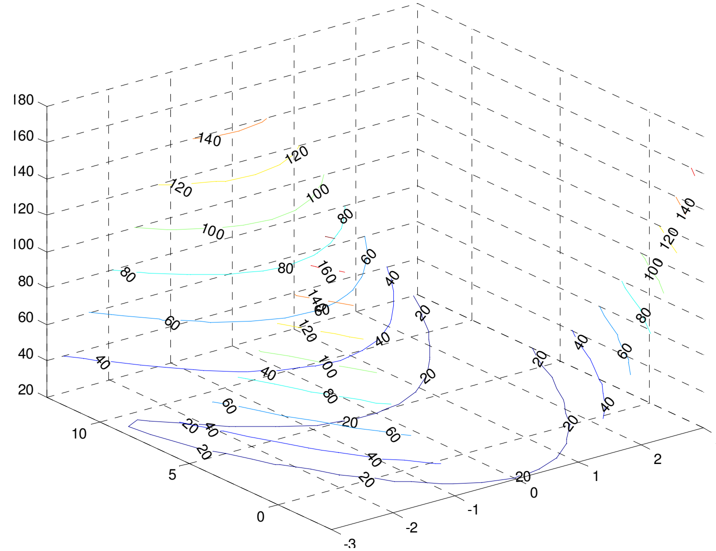
  `colorbar`(s)

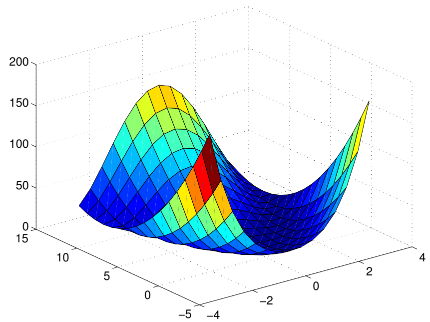
  que coloca una barra de colores y sus correspondientes valores numéricos adyacente a la figura

  La cantidad *z* es un string igual a 'horiz' o 'vert' para indicar la orientación de la barra. El valor por defecto es 'vert'

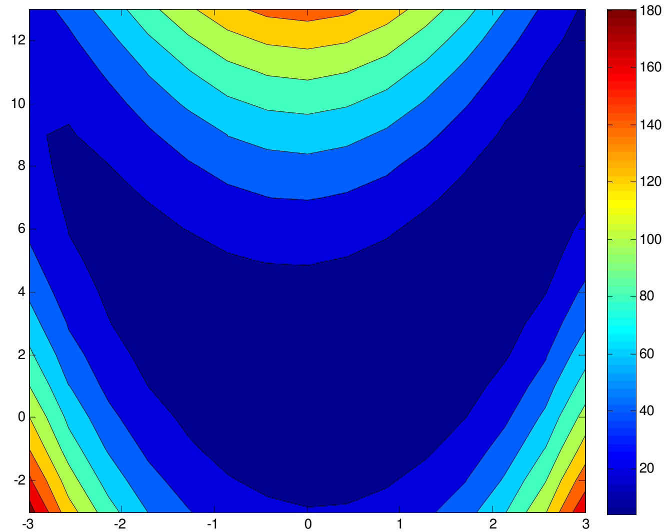
Ejemplos de contour3, contourf y colorbar

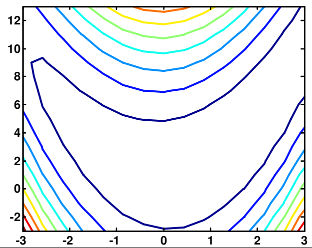
```
[x,y,z] = SurfExample;
[C, h] = contour3(x, y, z);
clabel(C, h)
```





```
[x,y,z] = SurfExample;
[C, h] = contourf(x, y, z);
colorbar
```

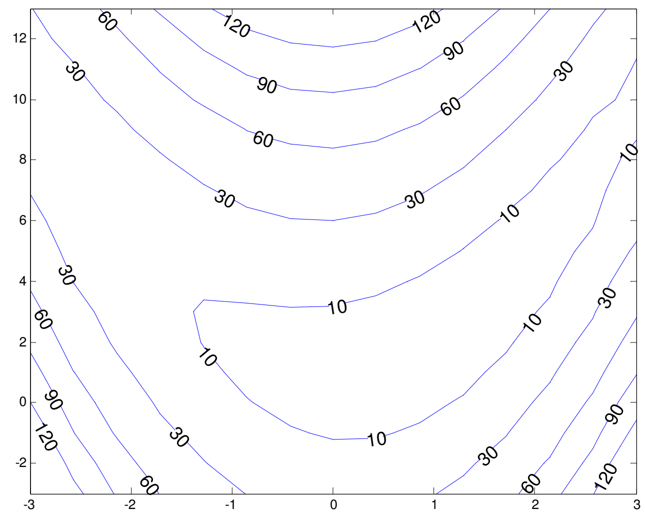




Gráficos de contornos 3D

- Las propiedades de las líneas y etiquetas se pueden modificar de forma similar que para `plot`
- Por ejemplo, para cambiar el tamaño de las etiquetas creadas con `contour` a 14 puntos y las líneas del contorno azules, se siguen los pasos

```
[x, y, z] = SurfExample;
[C, h] = contour(x, y, z, v)
g = clabel(C, h, v);
set(g, 'Fontsize', 14)
set(h, 'LineColor', ‘b')
```



Superficies cilíndricas, esféricas y elipsoidales

- Se puede usar una curva 2D como generador para crear superficies de revolución usando

  [x, y, z] = `cylinder`(r, n)

  que retorna las coordenadas x, y, z de una superficie cilíndrica utilizandoel vector r para definir una curva perfil

  La función `cylinder` trata cada elemento en r como un radio en n puntos equiespaciados alrededeor de su circunferencia. Si se omite n se considera el valor 20

Ejemplo de superficie cilíndrica

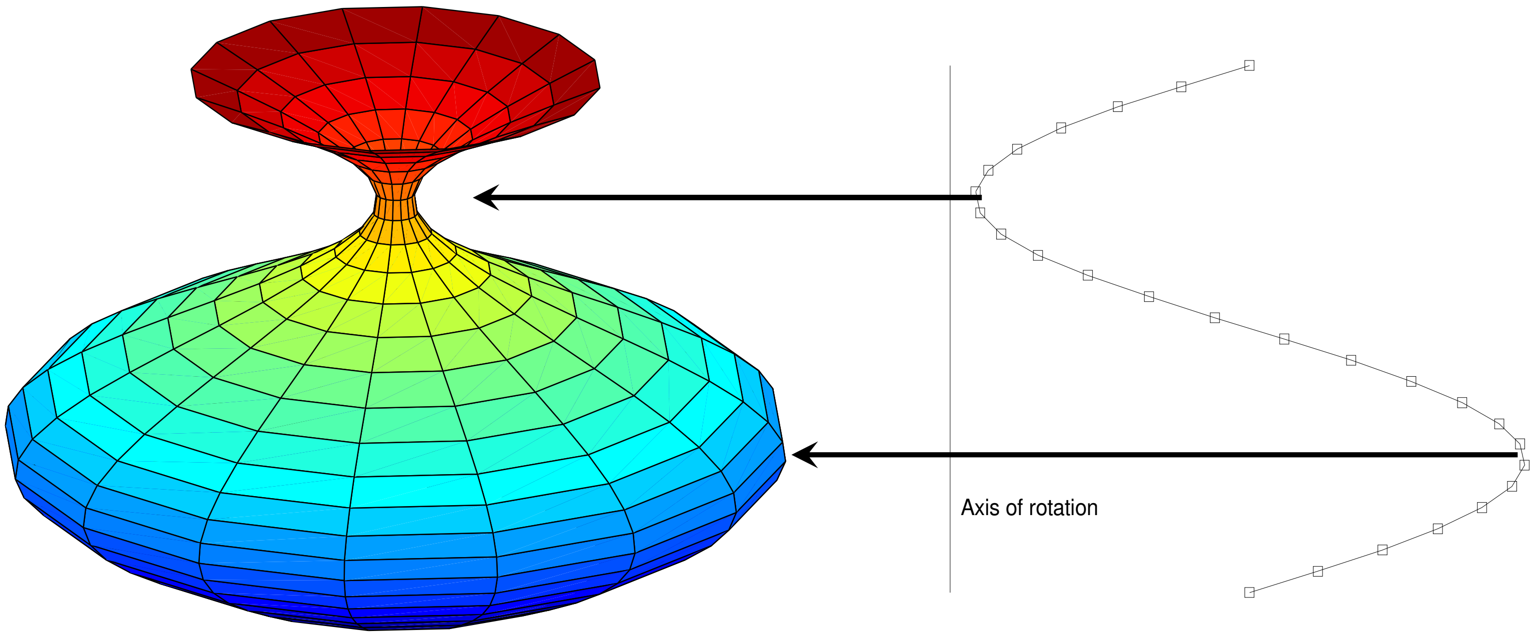
- Para la curva

$$r = 1.1 + \sin(z) \qquad 0 \geq z \leq 2\pi$$

  que se rota 360° alrededor del eje-*z*
  Se usa 26 intervalos equiespaciados en la dirección *z* y 16 intervalor equiespaciados en la dirección circunferencial
- El script para graficar la superficie cilíndrica es

```
zz = linspace(0, 2*pi, 26);
[x, y, z] = cylinder(1.1+sin(zz), 16);
surf(x, y, z)
axis off
```

Ejemplo de superficie cilíndrica

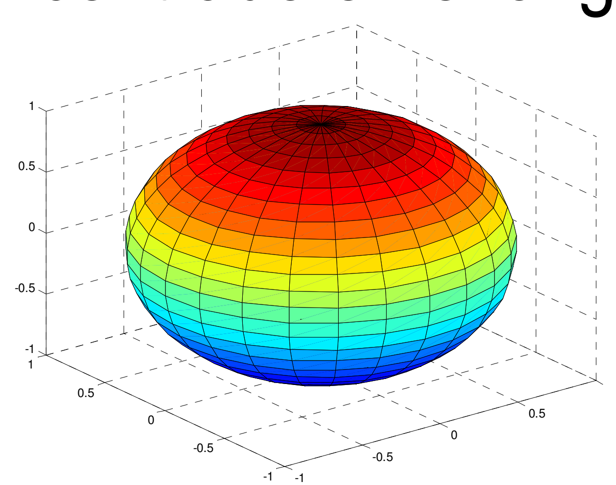


Superficies cilíndricas, esféricas y elipsoidales

- Para crear una esfera, se puede usar

```
[x, y, z] = sphere(n);
axis equal
surf(x, y, z)
```

donde n es el número de n x n elementos que comprende la esfera de radio 1 centrado en el origen. Si n se omite se toma $n = 20$



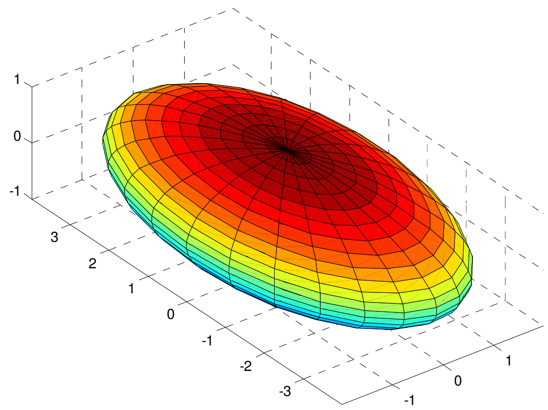
Superficies cilíndricas, esféricas y elipsoidales

- Para crear una elipsoide, se puede usar

  [x, y, z] = `ellipsoid`(xc, yc, zc, xr, yr, zr, n);

  `axis equal`

  `surf`(x, y, z)



en (*xc*, *yc*, *zc*) con longitud de semi-ejes en las direcciones *x*, *y*, *z* respectivamente, de *xr*, *yr*, y *zr* . *n* es el número de *n* x *n* elementos que comprende el elipsoide. Si *n* se omite se toma *n* = 20

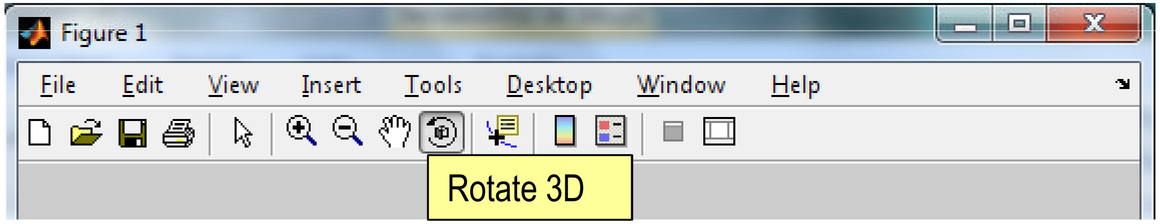
Angulo de visión

- En ocasiones se desea cambiar el ángulo de vista por defecto de los gráficos 3D porque
  - No se muestra las características de interés
  - Varias vistas diferentes deben mostrarse usando subplot
  - La exploración de la superficie desde varias vistas es deseable antes de decidir la orientación final
- Para determinar el azimuth (*a*) y ángulo de elevación de la vista (*e*), se usa

  [a, e] = `view`

Angulo de visión

- Para orientar el objeto se usa el icono *Rotate 3D* en la ventana de la figura y se orienta el objeto hasta obtener una orientación satisfactoria. Se mostrará los valores de azimuth y elevación mientras se rota



- Esos valores se pueden ingresar en la expresión

  `view`(an, en)

  para crear la orientación deseada cuando se ejecuta un script

Sombreado (shading)

- Las superficies creadas con surf usan la propiedad de sombreado por defecto llamada 'faceted'.
- La función que cambia el sombreado es

  `shading` s

  donde *s* es un string igual a

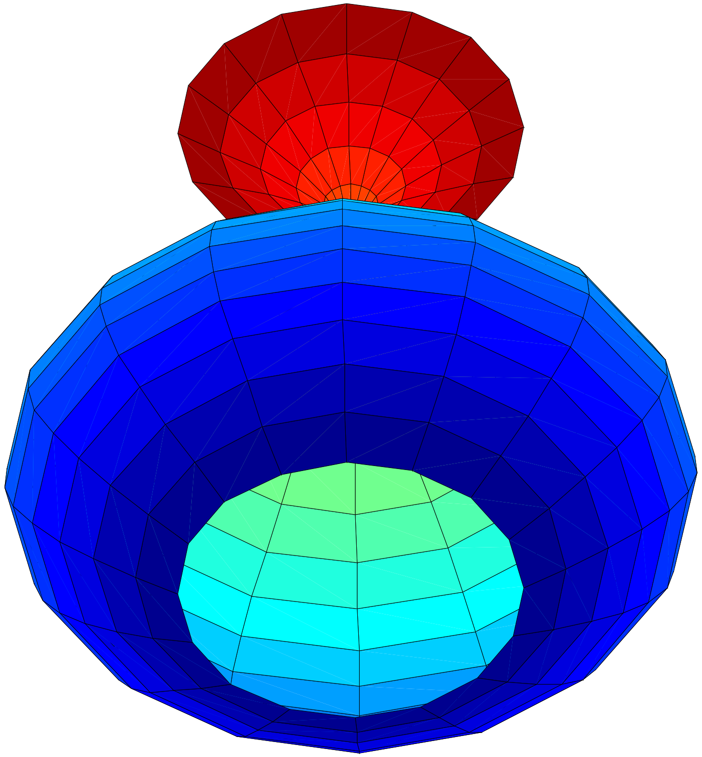
    faceted   % Default

    flat

    interp

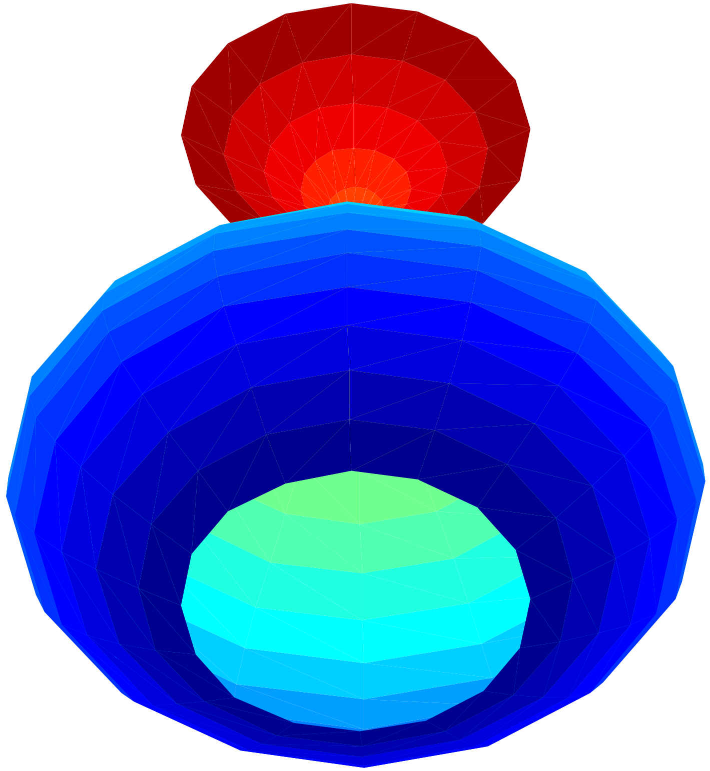
Ejemplo de view y shading

```
zz = linspace(0, 2*pi, 26);
r=1.1+sin(zz);
[x, y, z] = cylinder(r, 16);
surf(x, y, z)
view(-88.5, -48)
shading faceted
axis off vis3d
```



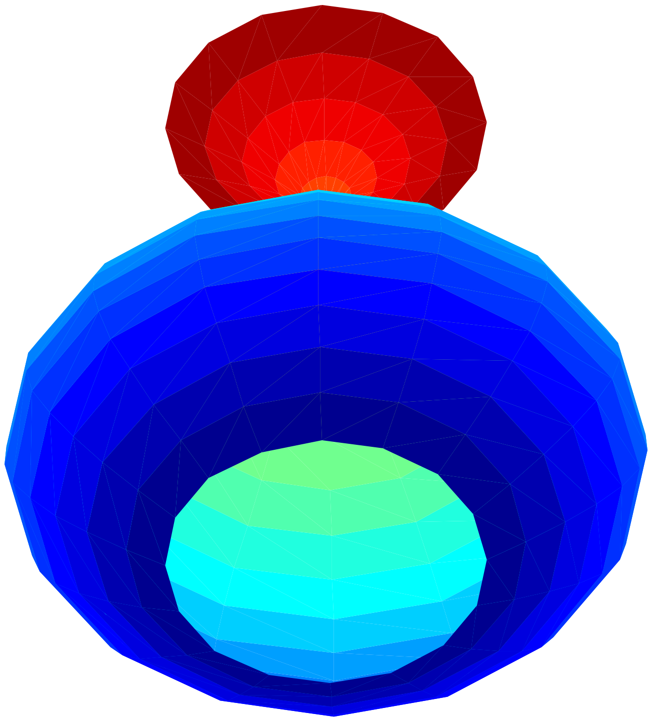
Ejemplo de view y shading

```
zz = linspace(0, 2*pi, 26);
r=1.1+sin(zz);
[x, y, z] = cylinder(r, 16);
surf(x, y, z)
view(-88.5, -48)
shading flat
axis off vis3d
```



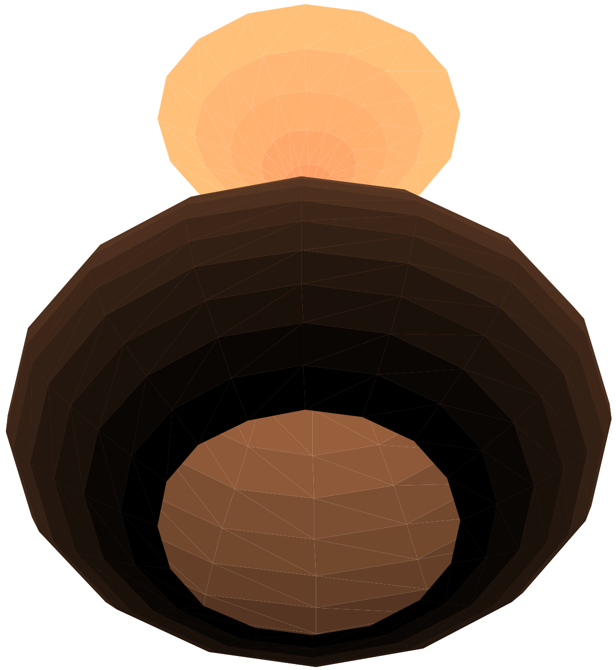
Ejemplo de view y shading

```
zz = linspace(0, 2*pi, 26);
r=1.1+sin(zz);
[x, y, z] = cylinder(r, 16);
surf(x, y, z)
view(-88.5, -48)
shading interp
axis off vis3d
```



Ejemplo de view y shading

```
r = 1+sin(zz);
[x, y, z] = cylinder(r, 16);
surf(x, y, z)
view(-88.5, -48)
shading interp
colormap(copper)
axis off vis3d
```



Transparencia

- Las superficies creadas con surf puede tener su opacidad alterada asignando un valor numérico al keyword 'FaceAlpha'
- El efecto de este keyword en la superficie resultante es dependiente del tipo del sombreado seleccionado
- Para ilustrar la opción de transparencia, se crea una función que genera los valores numéricos para la superficie dada por

$$x = a^{v}\cos v(1+\cos u)$$
$$y = -a^{v}\sin v(1+\cos u)$$
$$z = -ba^{v}(1+\sin u)$$

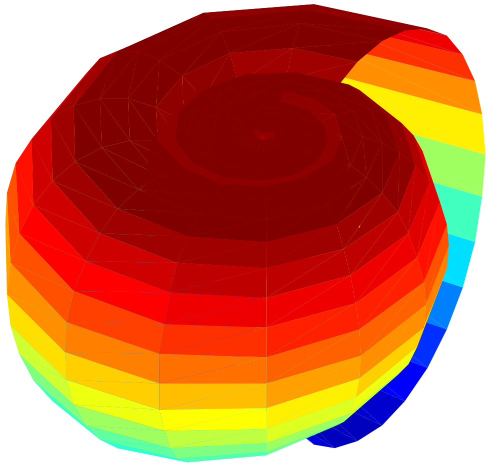
Transparencia

- Si se asume que *a* = 1.13 y *b* = 1.14, la función fichero m para esta superficie es

```
function [x, y, z] = Transparency
a = 1.13;   b = 1.14;
uu = linspace(0, 2*pi, 30);
vv = linspace(-15, 6, 45);
[u, v] = meshgrid(uu, vv);
x = a.^v.*cos(v).*(1+cos(u));
y = -a.^v.*sin(v).*(1+cos(u));
z = -b*a.^v.*(1+sin(u));
```

Ejemplo de transparencia

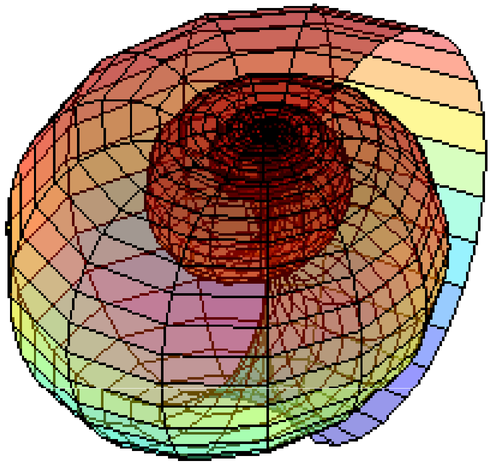
```
[x, y, z] = Transparency;
surf(x, y, z)
shading interp
axis vis3d off
equal
view([-35 38])
```



```
[x, y, z] = Transparency;
h = surf(x, y, z)
set(h, 'FaceAlpha', 0.4)
shading interp
axis vis3d off equal
view([-35 38])
```

Ejemplo de transparencia

```
[x, y, z] = Transparency;
h = surf(x, y, z)
set(h, 'FaceAlpha', 0.4)
axis vis3d off equal
view([-35 38])
```



Nota: se omite `shading`

Ejemplo: coloreado de cajas

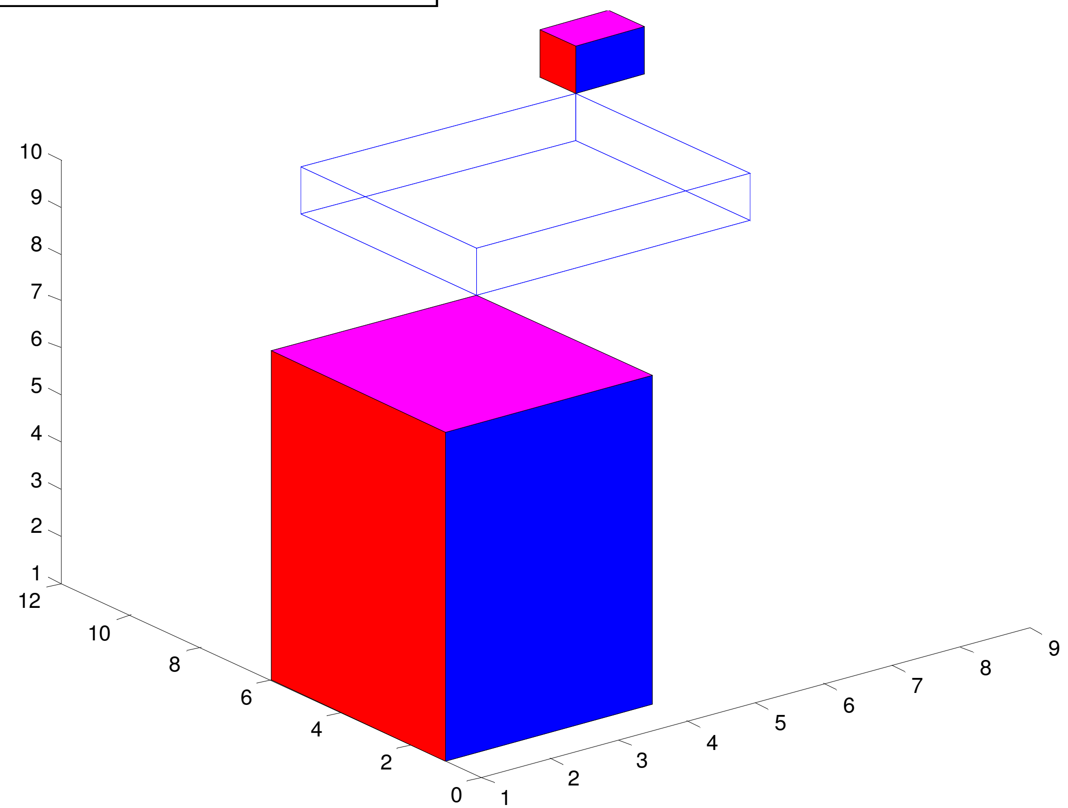
- Modificación de fichero m *BoxPlot3* para que las seis superficies representadas por los rectángulos se rellene con un color diferente
- La modificación se consigue usando `fill3`
- La versión revisada de *BoxPlot3* renombrada como *BoxPlot3C* es

Ejemplo: coloreado de cajas

```
function BoxPlot3C(xo, yo, zo, Lx, Ly, Lz, w)
% w = 0, wire frame;  w = 1, rectangles are colored
x = [xo     xo      xo      xo      xo+Lx   xo+Lx   xo+Lx   xo+Lx];
y = [yo     yo      yo+Ly   yo+Ly   yo      yo      yo+Ly   yo+Ly];
z = [zo     zo+Lz   zo+Lz   zo      zo      zo+Lz   zo+Lz   zo    ];
index = zeros(6,5);
index(1,:) = [1 2 3 4 1];
index(2,:) = [5 6 7 8 5];
index(3,:) = [1 2 6 5 1];
index(4,:) = [4 3 7 8 4];
index(5,:) = [2 6 7 3 2];
index(6,:) = [1 5 8 4 1];
c = 'rgbcmy';
for k = 1:6
  if w~=0
    fill3(x(index(k,:)), y(index(k,:)), z(index(k,:)), c(k))
  else
    plot3(x(index(k,:)), y(index(k,:)), z(index(k,:)))
  end
  hold on
end
```

Ejemplo: coloreado de cajas

```
BoxPlot3C(1, 1, 1, 3, 5, 7, 1)
BoxPlot3C(4, 6, 8, 4, 5, 1, 0)
BoxPlot3C(8, 11, 9, 1, 1, 1, 1)
```



Ejemplo: intersección de un cilindro y una esfera y resaltado de su intersección

- La curva que resulta de la intersección de una esfera de radio 2*a* centrada en el origen y un cilindro circular de radio *a* centrado en (*a*, 0) es dado por las ecuaciones paramétricas

$$x = a(1+\cos\varphi)$$
$$y = a\sin\varphi$$
$$z = 2a\sin(\varphi/2)$$

donde $0 \le \varphi \le 4\pi$

- Para crear una esfera de radio 2*a*, se multiplica cada coordenada de `sphere` por 2*a*.

Ejemplo: intersección de un cilindro y una esfera y resaltado de su intersección

- Las coordenadas de `cylinder` se modifican con la transformación:

$$x \to ax + a$$

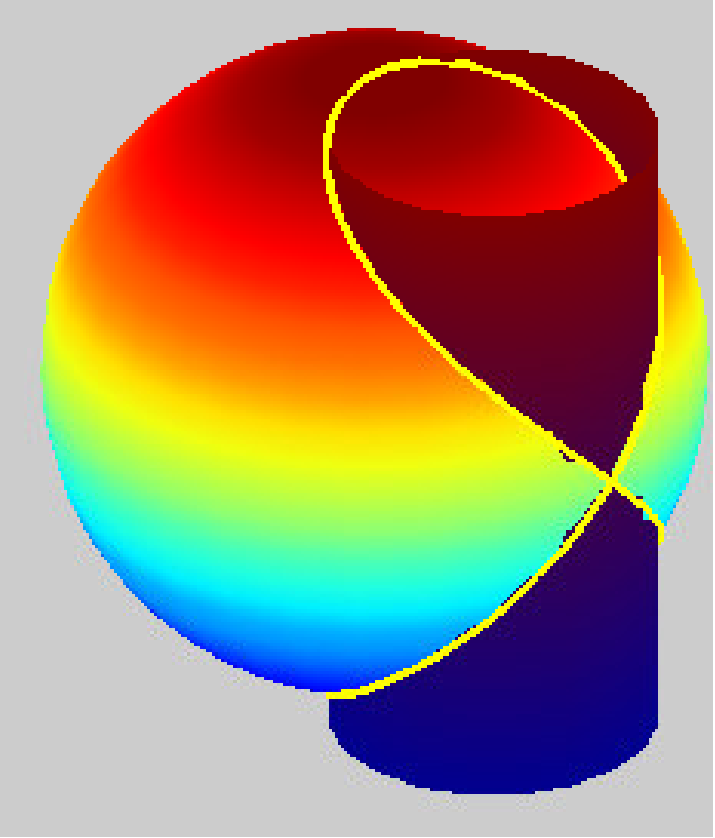
$$y \to ay$$

$$z \to 4az - 2a$$

- Se asume que $a = 1$. El script es

Ejemplo: intersección de un cilindro y una esfera y resaltado de su intersección

```
a = 1;
[xs, ys, zs] = sphere(30);
surf(2*a*xs, 2*a*ys, 2*a*zs)
hold on
[x, y, z] = cylinder;
surf(a*x+a, a*y, 4*a*z-2*a)
shading interp
t = linspace(0, 4*pi, 100);
x = a*(1+cos(t));
y = a*sin(t);
z = 2*a*sin(t/2);
plot3(x, y, z, 'y-', 'Linewidth', 2.5);
axis equal off
view([45, 30])
```



Ejemplo: mejora de gráficos 2D con objetos 3D

- Para una esfera de radio *a* y un elipsoide con su eje mayor en la dirección *x* igual a 2*a*, eje menor en la dirección *y* igual a 2*b*, y un eje menor en la dirección *z* igual a 2*c*, la proporción del volumen de un elipsoide con relación al volumen de una esfera es

$$V = \frac{V_{ellipse}}{V_{sphere}} = \left(\frac{b}{a}\right)\left(\frac{c}{a}\right)$$

- Se crea el siguiente programa para mejorar la comprensión de un gráfico de *V* como función de *b*/*a* para varios valores de *c*/*a*

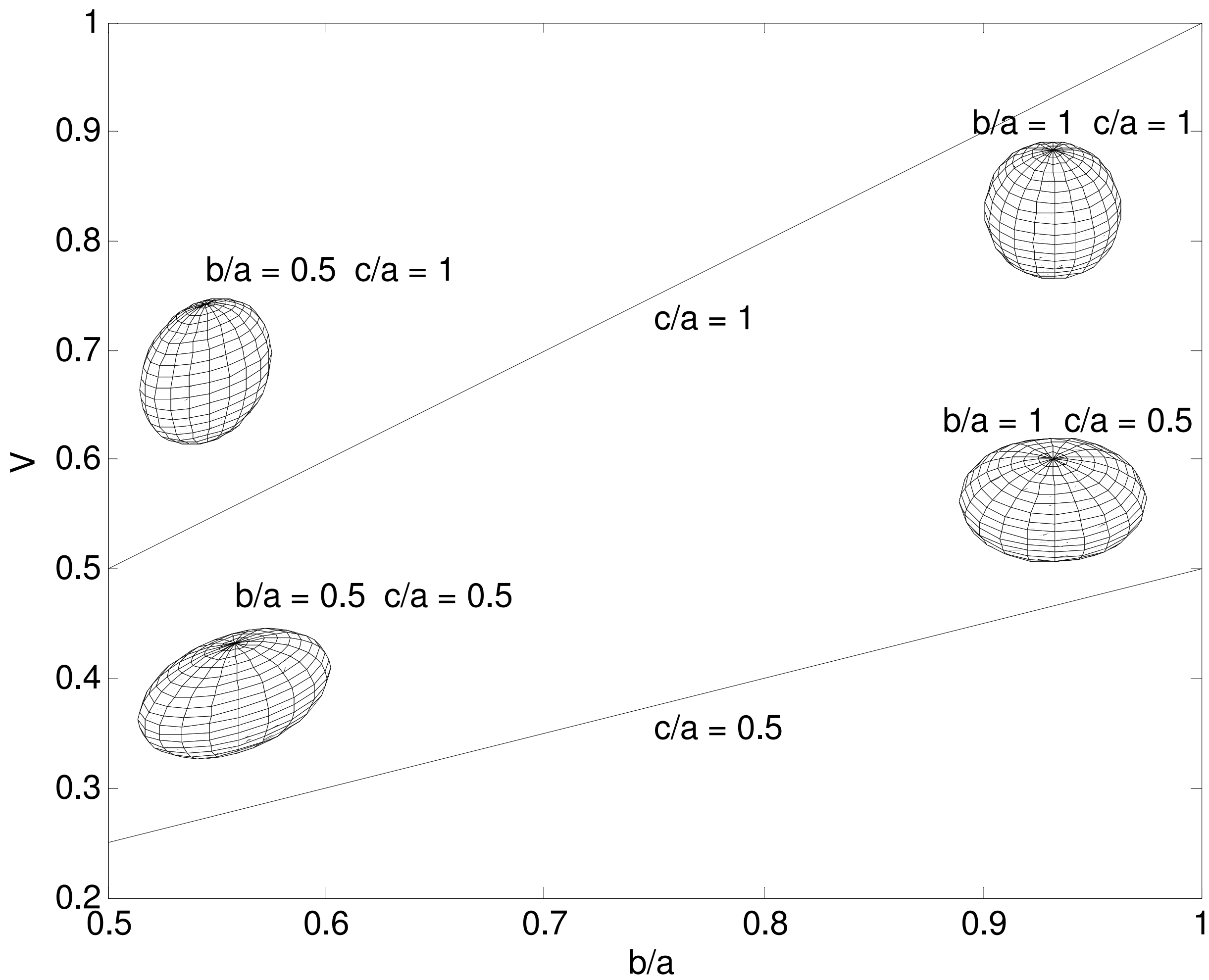
Ejemplo: mejora de gráficos 2D con objetos 3D

```
b = [0.5, 1];   c = b;
for k = 1:2
  plot(b, b*c(k), 'k-')
  text(0.75, (b(1)*c(k)+b(2)*c(k))/2-0.02, ['c/a = ' num2str(c(k))])
  hold on
end
xlabel('b/a')  ylabel('V')
for k = 1:4
  switch k
    case 1
      axes('position', [0.12, 0.2, 0.2, 0.2])
      [xs, ys, zs] = ellipsoid(0, 0, 0, 1, b(1), c(1), 20);
      mesh(xs, ys, zs)
      text(0, 0, 1, ['b/a = ' num2str(b(1)) '  c/a = ' num2str(c(1))])
 case 2
      axes ('position', [0.1, 0.5, 0.2, 0.2])
      [xs, ys, zs] = ellipsoid(0, 0, 0, 1, b(1), c(2), 20);
      mesh (xs, ys, zs)
      text (0, 0, 1.5, ['b/a = ' num2str(b(1)) ' c/a = ' num2str(c(2))])
```

Ejemplo: mejora de gráficos 2D con objetos 3D

```
    case 3
      axes ('position', [0.7, 0.65, 0.2, 0.2])
      [xs, ys, zs] = ellipsoid(0, 0, 0, 1, b(2), c(2), 20);
      mesh (xs, ys, zs)
      text (-1.5, 0, 2, ['b/a = ' num2str(b(2)) ' c/a = ' num2str(c(2))])
    case 4
      axes ('position', [0.7, 0.38, 0.2, 0.2])
      [xs, ys, zs] = ellipsoid(0, 0, 0, 1, b(2), c(1), 20);
      mesh (xs, ys, zs)
      text (-1.5, 0, 1.5, ['b/a = ' num2str(b(2)) ' c/a = ' num2str(c(1))])
  end
  colormap([0 0 0])
  axis equal off
end
```

Ejemplo: mejora de gráficos 2D con objetos 3D



Rotación y traslación de objetos 3D: ángulos de Euler

- La rotación y traslación de un punto $p(x,y,z)$ a otra posición $P(X,Y,Z)$ es determinado por

$$X = L_x + a_{11}x + a_{12}y + a_{13}z$$

$$Y = L_y + a_{21}x + a_{22}y + a_{23}z$$

$$Z = L_z + a_{21}x + a_{22}y + a_{23}z$$

donde L_x, L_y, y L_z son los componentes x, y, z de la traslación, respectivamente, y a_{ij}, i, j = 1, 2, 3, son los elementos de

$$a = \begin{bmatrix} \cos\psi\cos\chi & -\cos\psi\sin\chi & \sin\psi \\ \cos\varphi\sin\chi + \sin\varphi\sin\psi\cos\chi & \cos\varphi\cos\chi - \sin\varphi\sin\psi\sin\chi & -\sin\varphi\cos\psi \\ \sin\varphi\sin\chi - \cos\varphi\sin\psi\cos\chi & \sin\varphi\cos\chi + \cos\varphi\sin\psi\sin\chi & \cos\varphi\cos\psi \end{bmatrix}$$

Rotación y traslación de objetos 3D: ángulos de Euler

- Las cantidades ϕ, ψ, y χ son los ángulos de rotación ordenados (ángulos de Euler) del sistema de coordenadas alrededor del origen

  ϕ alrededor del eje *x*

  ψ alrededor del eje *y*

  χ alrededor del eje *z*

- En general, (*x*,*y*,*z*) pueden ser escalares, vectores de la misma longitud, o matrices del mismo orden

- Se crea la función *EulerAngles*

Rotación y traslación de objetos 3D: ángulos de Euler

```
function [Xrt, Yrt, Zrt] = EulerAngles(psi, chi, phi, Lx, Ly, Lz, x, y, z)
a = [cos(psi)*cos(chi), -cos(psi)*sin(chi), sin(psi); ...
     cos(phi)*sin(chi)+sin(phi)*sin(psi)*cos(chi), ...
     cos(phi)*cos(chi)-sin(phi)*sin(psi)*sin(chi), ...
     -sin(phi)*cos(psi); ...
     sin(phi)*sin(chi)-cos(phi)*sin(psi)*cos(chi), ...
     sin(phi)*cos(chi)+cos(phi)*sin(psi)*sin(chi), ...
     cos(phi)*cos(psi)];
Xrt = a(1,1)*x+a(1,2)*y+a(1,3)*z+Lx;
Yrt = a(2,1)*x+a(2,2)*y+a(2,3)*z+Ly;
Zrt = a(3,1)*x+a(3,2)*y+a(3,3)*z+Lz;
```

$$\boxed{\begin{aligned} X &= L_x + a_{11}x + a_{12}y + a_{13}z \\ Y &= L_y + a_{21}x + a_{22}y + a_{23}z \\ Z &= L_z + a_{21}x + a_{22}y + a_{23}z \end{aligned}}$$

Rotación y traslación de objetos 3D: generación de Toro

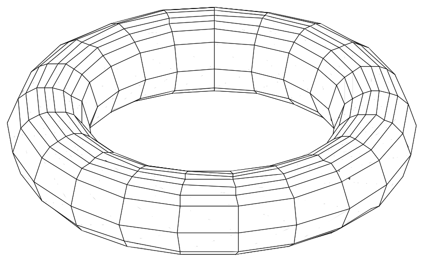
- Las ecuaciones para generar un toro son

$$x = r\cos\theta$$

$$y = r\sin\theta$$

$$z = \pm\sqrt{a^2 - \left(\sqrt{x^2 + y^2} - b\right)^2}$$

Torus



donde $b - a \leq r \leq b + a$, $0 \leq \theta \leq 2\pi$, y $b > a$

Se crea la función *Torus* para obtener las coordenadas del toro que usa las función `real` para eliminar la parte imaginaria debida a redondeos numéricos

Rotación y traslación de objetos 3D: generación de Toro

```
function [X, Y, Z] = Torus(a, b)
r = linspace(b-a, b+a, 10);
th = linspace(0, 2*pi, 22);
x = r'*cos(th);
y = r'*sin(th);
z = real(sqrt(a^2-(sqrt(x.^2+y.^2)-b).^2));
X = [x x];
Y = [y y];
Z = [z -z];
```

Rotación y traslación de objetos 3D: generación de Toro

- Se obtendrá cuatro gráficas del toro:
  - Sin rotación
  - Rotado 60° alrededor del eje *x* (ϕ = 60°) y comparado con el toro original
  - Rotado 60° alrededor del eje *y* (ψ = 60°) y comparado con el toro original
  - Rotado 60° alrededor del eje *x* (ϕ = 60°), rotado 60° alrededor del eje *y* (ψ = 60°) y comparado con el toro original
- Se asume que *a* = 0.2 y *b* = 0.8 y se usa `colormap` para producir una malla de líneas

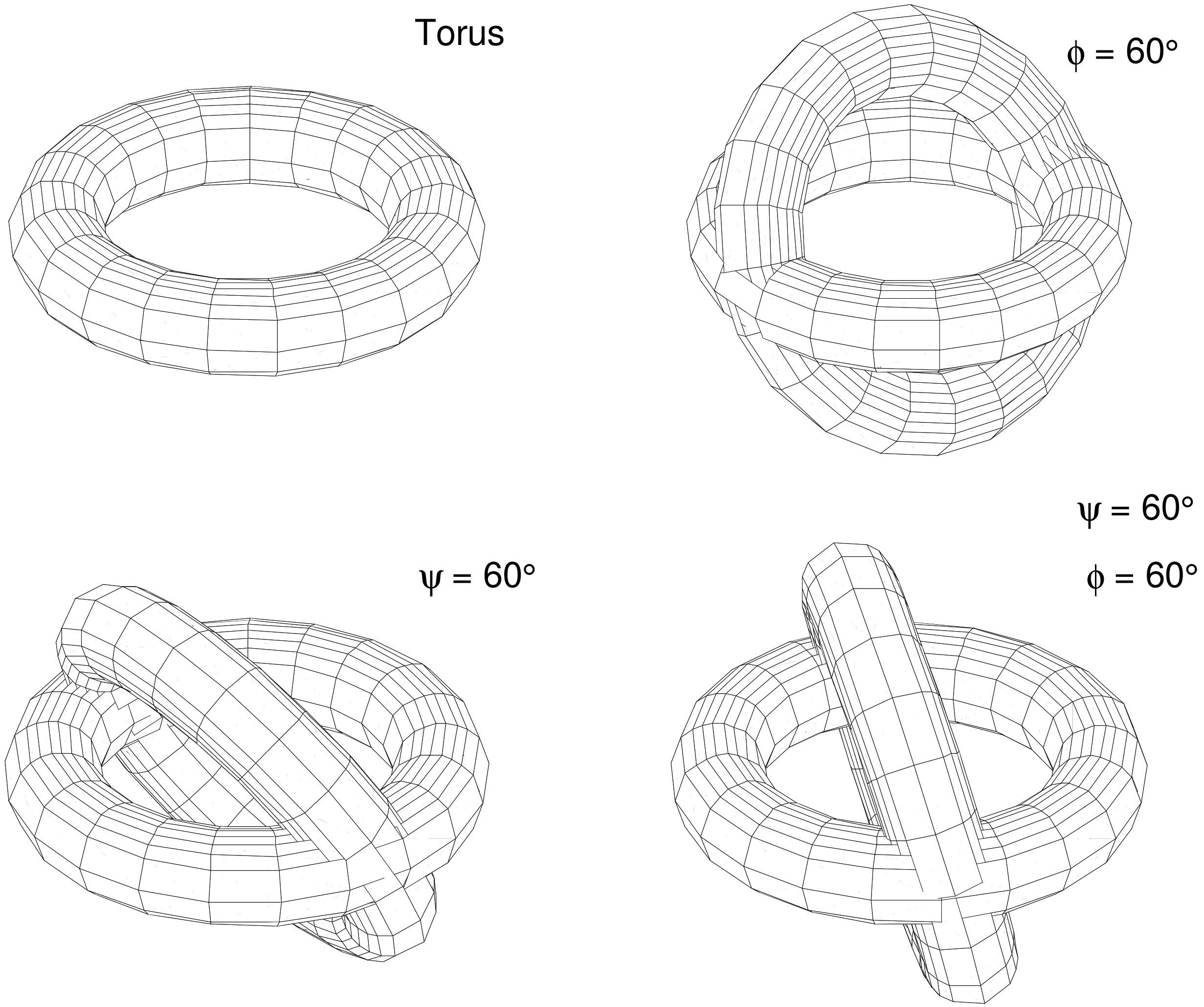
Rotación y traslación de objetos 3D: generación de Toro

```
[X, Y, Z] = Torus(0.2, 0.8);
psi = [0, pi/3, pi/3];  chi = [0, 0, 0];  phi = [pi/3,0, pi/3];
Lx = 0; Ly = 0; Lz = 0;
for k = 1:4
   subplot(2,2,k)
   if k==1
      mesh(X, Y, Z)
   else
      mesh(X, Y, Z)
      hold on
      [Xr Yr Zr] = EulerAngles(psi(k-1), chi(k-1), …
                      phi(k-1), Lx, Ly, Lz, X, Y, Z);
      mesh(Xr, Yr, Zr)
  end
```

Rotación y traslación de objetos 3D: generación de Toro

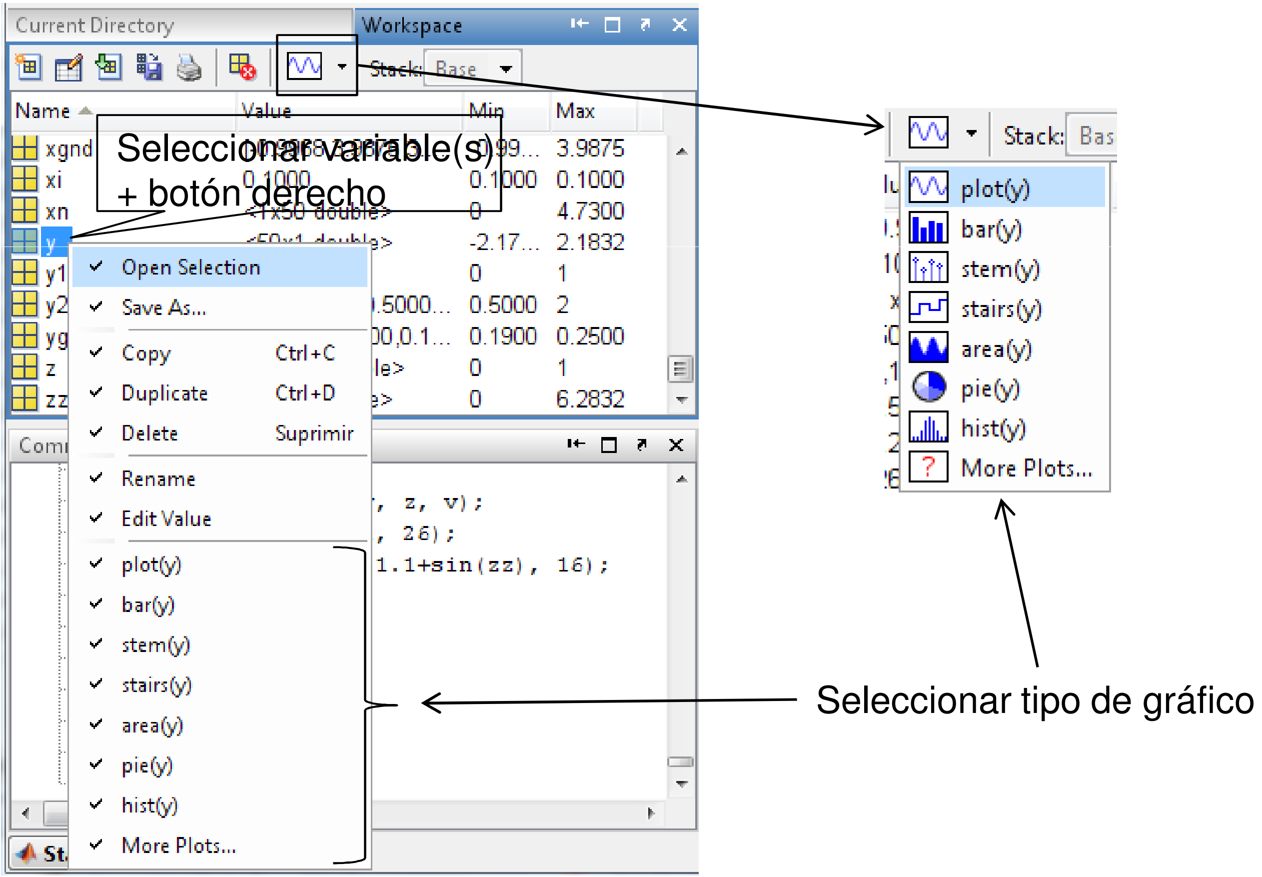
```
switch k
  case 1
    text(0.5, -0.5, 1, 'Torus')
  case 2
    text(0.5, -0.5, 1,'\phi = 60\circ')
  case 3
   text(0.5,-0.5,1,'\psi = 60\circ')
  case 4
    text(0.5, -0.5, 1.35,'\psi = 60\circ')
    text(0.55, -0.5, 1,'\phi = 60\circ')
 end
 colormap([0 0 0])
 axis equal off
 grid off
end
```

Rotación y traslación de objetos 3D: generación de Toro



Creación de gráficos interactivamente

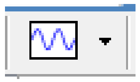
- El entorno Matlab permite crear gráficas interactivamente de varias maneras

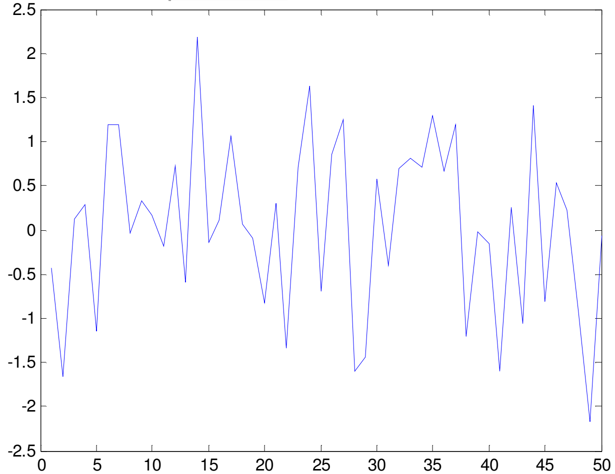


Creación de gráficos interactivamente

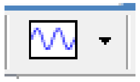
- Se introducen los siguientes comandos:

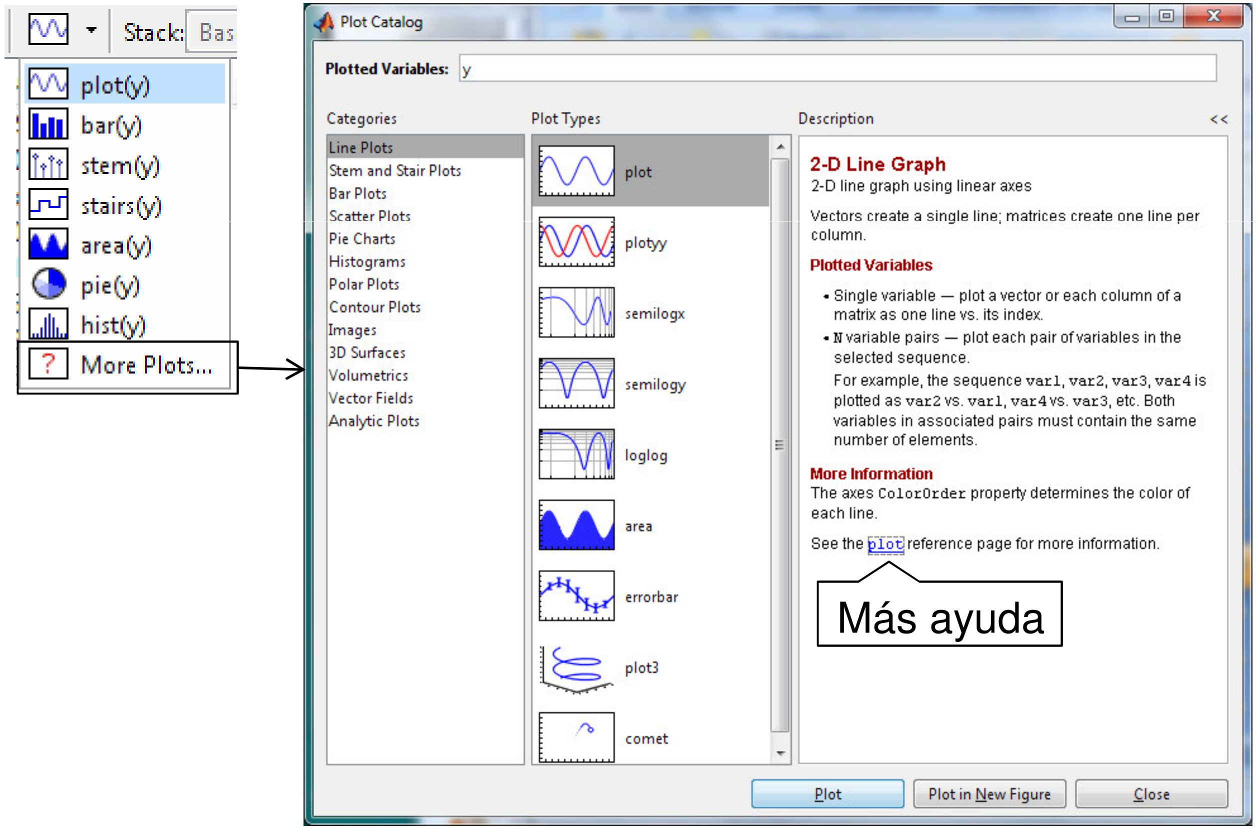
```
>> N=50;
>> y=randn(N,1);
>> y2=filter([1 1]/2,1,y);
```

- Se pulsa sobre la variable *y* en el Workspace y se pulsa sobre el icono . Se obtiene el gráfico



Creación de gráficos interactivamente

- Se puede modificar el tipo de gráfico desplegando el menú  para obtener la descripción



Creación de gráficos interactivamente

- Cuando se selecciona un tipo de gráfico se genera el comando correspondiente en la ventana de comandos

Command Window

New to MATLAB? Watch this Video, see Demos, or read Getting Started.

```
>> plot(y, 'DisplayName', 'y', 'YDataSource', 'y'); figure(gcf)
>> bar(y, 'DisplayName', 'y', 'YDataSource', 'y'); figure(gcf)
>> stem(y, 'DisplayName', 'y', 'YDataSource', 'y'); figure(gcf)
>> 
```

Creación de gráficos interactivamente

- En la ventana de figura se puede modificar el gráfico, generar el código y guardarlo para ser invocado