GRÁFICOS EM MATLAB

Isabel Espírito Santo

Universidade do Minho

2011

Conteúdo

Capítulo 1

Gráficos

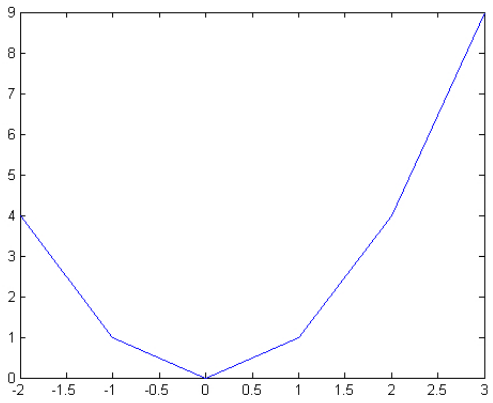
1.1 Gráficos 2d

1.1.1 Representação de conjuntos de pontos

Para se representar um conjunto de pontos a duas dimensões, utiliza-se o comando `PLOT`. Este pode ser usado com vectores da mesma dimensão ou com uma função. A sintaxe é `PLOT(X,Y)`, que desenha o gráfico de um vector `Y` em função de um vector `X`. Se `X` ou `Y` for uma matriz, então é feito o gráfico de um vector *versus* as linhas ou colunas da matriz, de acordo com o que tiver maior dimensão. Se `X` for um escalar e `Y` um vector, são criados objectos discretos e desenhados como pontos verticalmente em `X`.

Exemplo 1.1.1 *Comando* `PLOT`

```
>> x=[-2:3];
>> y=[4 1 0 1 4 9];
>> plot(x,y)
```



`PLOT(Y)` desenha as colunas de `Y` em função dos seus índices. Se `Y` for complexo, então `PLOT(Y)` é equivalente a `PLOT(real(Y),imag(Y))`. Nos restantes casos do uso da função `plot` a parte imaginária dos números complexos é ignorada.

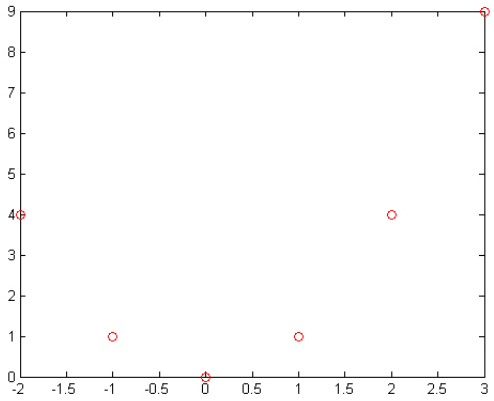
Podem obter-se vários tipos de linhas, símbolos e cores com a sintaxe `PLOT(X,Y,S)`, em que `S` é uma *string* formada por um elemento de qualquer uma ou todas as seguintes 3 colunas.

cor		símbolo		linha	
b	azul	.	ponto	-	sólida
g	verde	o	círculo	:	ponteado
r	vermelho	x	cruz	-.	traço e linha
c	cyan	+	mais	–	tracejado
m	magenta	*	estrela	(nada)	sem linha
y	amarelo	s	quadrado		
k	preto	d	losango		
w	branco	v	triângulo (para baixo)		
		^	triângulo (para cima)		
		<	triângulo (para a esquerda)		
		>	triângulo (para a direita)		
		p	pentágono		
		h	hexágono		

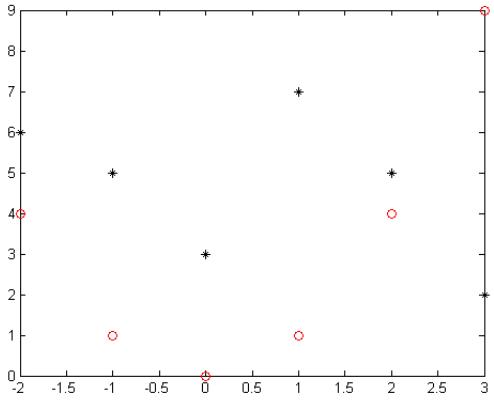
`PLOT(X1,Y1,S1,X2,Y2,S2,X3,Y3,S3,...)` - combina os gráficos definidos pelos conjuntos triplos `(X,Y,S)`, em que `X` e `Y` são vectores ou matrizes e `S` são *strings*.

Exemplo 1.1.2 *Alteração de linhas ou marcas*

```
>> plot(x,y,'or')
```




```
>> hold on
>> z=[6 5 3 7 5 2];
>> plot(x,z,'*k')
```



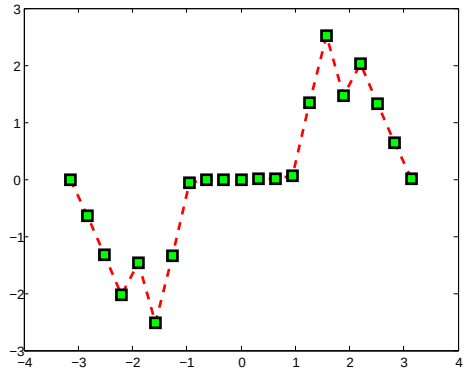
Se não for especificada nenhuma cor, o comando `PLOT`, faz uso automático das cores especificadas pela propriedade dos eixos `ColorOrder`. Por defeito, o comando `PLOT` usa as cores da propriedade `ColorOrder` de forma cíclica. Para sistemas monocromáticos, o comando `PLOT` usa a propriedade dos eixos `LineStyleOrder` de forma cíclica.

Se não for especificado nenhum tipo de marca, a função `PLOT` não usa nenhuma. Se não for especificado nenhum tipo de linha, a função `PLOT` usa uma linha sólida.

Os pares `X,Y`, ou os trios `X,Y,S` podem ser seguidos por parâmetros ou valores para especificar propriedades adicionais às linhas ou marcas.

Exemplo 1.1.3 *Propriedades das linhas e marcas*

```
>> x = -pi:pi/10:pi;
>> y = tan(sin(x)) - sin(tan(x));
>> plot(x,y,'--rs','LineWidth',2,...
                    'MarkerEdgeColor','k',...
                    'MarkerFaceColor','g',...
                    'MarkerSize',10)
```



1.1.2 Manipulação de gráficos

`>> xlabel('...')` - título do eixo dos xx

`>> ylabel('...')` - título do eixo dos yy

`>> title('...')` - título do gráfico

`>> text(x,y,'...')` - texto nas coordenadas (x,y)

`>> gtext('...')` - o local do texto é indicado com o rato

`>> grid on` - coloca uma grelha

`>> axis equal` - coloca as duas escalas iguais

`>> axis square` - o gráfico fica quadrado

`>> axis([xmin xmax ymin ymax])` - fixa os limites dos eixos

`>> axis normal` - anula as opções anteriores dos eixos

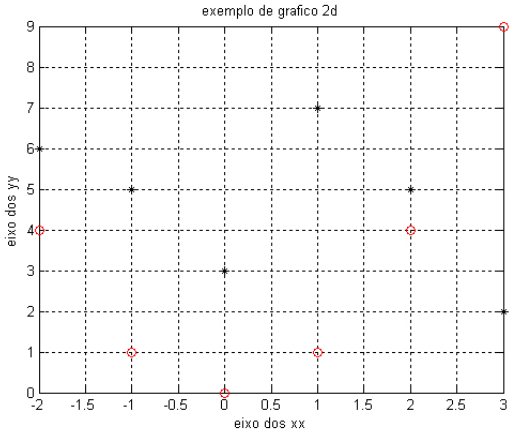
`>> hold on` - mantém o gráfico em espera na janela

`>> hold off` - descarta o gráfico anterior

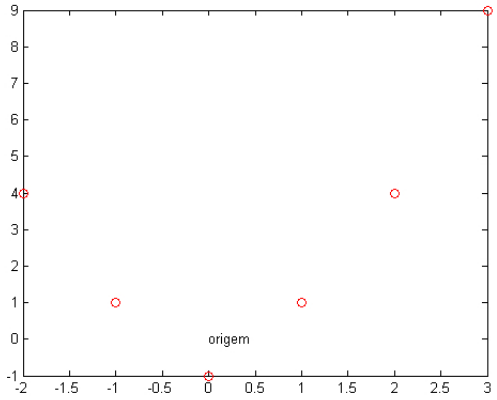
Todas estas opções podem ser realizadas na própria figura uma vez editadas as propriedades do gráfico (`edit ->figure properties`).

Exemplo 1.1.4 *Introdução de informação nos gráficos*

```
>> xlabel('eixo dos xx')
>> ylabel('eixo dos yy')
>> title('exemplo de grafico 2d')
>> grid on
```




```
>> x=[-2:3];
>> y=[4 1 0 1 4 9];
>> plot(x,y,'or')
>> text(0,0,'origem')
```



1.1.3 Outros gráficos 2d

Há alguns comandos que permitem produzir outros tipos de gráfico a duas dimensões, diferentes do produzido por `PLOT`.

>> `area` - pinta a área abaixo do gráfico

>> `bar` - gráfico de barras verticais

>> `barh` - gráfico de barras horizontais

>> `hist` - histograma

>> `pie` - sectores

>> `rose` - diagrama polar

>> `stairs` - gráfico de degraus

>> `stem` - sequência de dados discretos

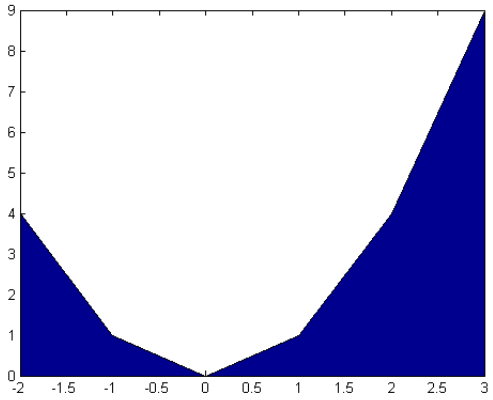
>> `loglog` - escala logaritmica em ambos os eixos

>> `semilogx` - com escala logaritmica no eixo dos xx

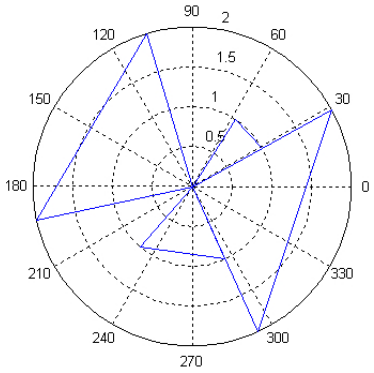
>> `semilogy` - com escala logaritmica no eixo dos yy

Exemplo 1.1.5 *Outros gráficos 2d*

```
>> x=[-2:3];
>> y=[4 1 -0 1 4 9];
>> area(x,y)
```




```
>> rose(x,y)
```

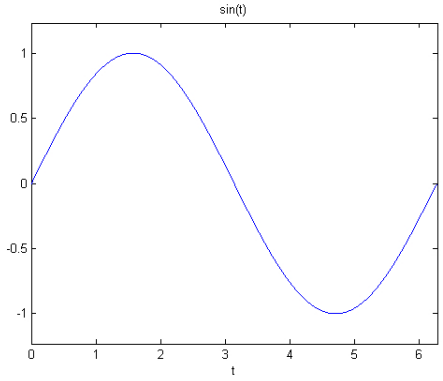


1.1.4 Representação de funções num intervalo

A função `EZPLOT` (easy to use function plotter) faz o gráfico de uma função explícita de `X` no domínio por defeito -2π <`X`< 2π. Para se alterar este intervalo, basta acrescentá-lo como segundo argumento de entrada da função.

Exemplo 1.1.6 *Comando* `EZPLOT`

```
>> ezplot('sin(t)',[0 2*pi])
```

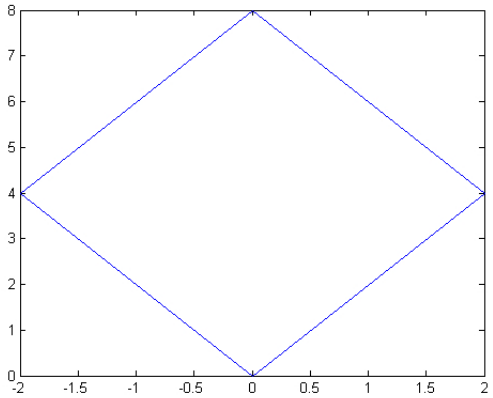


1.1.5 Gráficos de polígonos

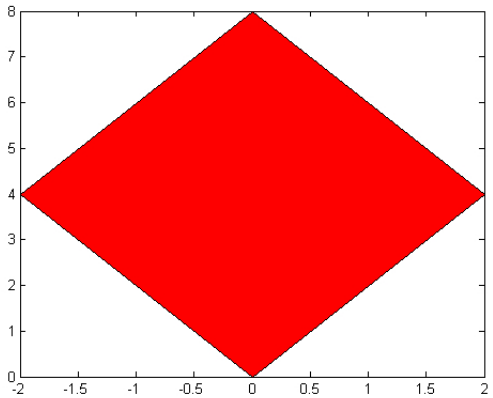
É possível, com o comando `PLOT` desenhar polígonos, sendo as coordenadas os vértices do mesmo. Se se pretender um polígono a cheio, usa-se o comando `FILL` em vez do comando `PLOT`.

Exemplo 1.1.7 *Polígonos*

```
>> x=[-2 0 2 0 -2];
>> y=[4 8 4 0 4];
>> plot(x,y)
```




```
>> fill(x,y,'r')
```



1.1.6 Gráficos paramétricos

Quando y não é dado explicitamente em função de x mas ambos são dados como funções de um certo parâmetro, podem o MATLAB permite recorrer a gráficos paramétricos.

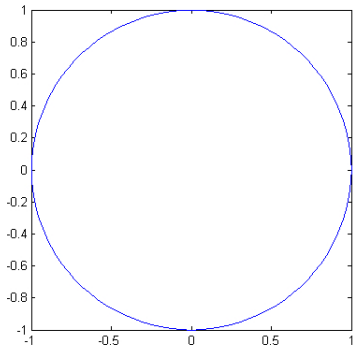
Exemplo 1.1.8 *Gráfico paramétrico de uma circunferência*

Desenhar o círculo de raio 1 centrado em $(0,0)$. *A fórmula paramétrica desta circunferência é*

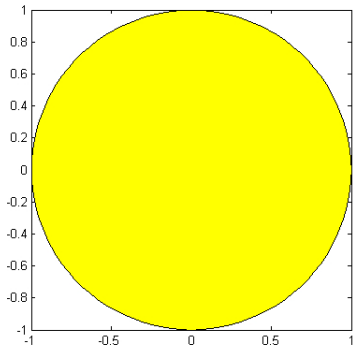
$$\begin{cases} x = \cos(2\pi t) \\ y = \sin(2\pi t), \end{cases}$$

com t entre 0 e 1.

```
>> t=0:0.01:1;
>> plot(cos(2*pi*t),sin(2*pi*t))
>> axis square
```




```
>> fill(cos(2*pi*t),sin(2*pi*t),'y')
>> axis square
```



Em alternativa, para a primeira figura pode usar-se o comando:

```
>> ezplot('cos(t)','sin(t)',[0 2*pi]);
>> axis square
```

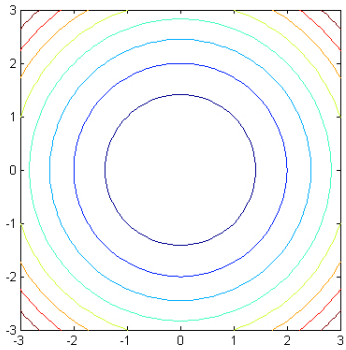
1.1.7 Curvas de nível

Muitas vezes é conveniente desenhar as curvas de nível de uma determinada função bidimensional no plano $x - y$ onde esta função assuma valores constantes.

Exemplo 1.1.9 *Curvas de nível*

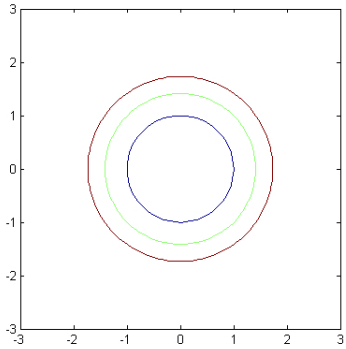
As curvas de nível da função x^2+y^2 são os círculos centrados na origem. Produz-se uma grelha de pontos com o comando `'meshgrid'` *e de seguida desenham-se as curvas com o comando* `'contour'`.

```
>> [x y]=meshgrid(-3:0.1:3,-3:0.1:3);
>> contour(x,y,x.^2+y.^2)
>> axis square
```



Exemplo 1.1.10 *Curvas de nível para valores da função específicos Para representar os círculos de raio* 1, $\sqrt{2}$ *e* $\sqrt{3}$*:*

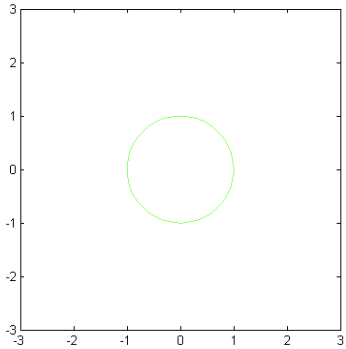
```
>> contour(x,y,x.^2+y.^2,[1 2 3])
>> axis square
```



Exemplo 1.1.11 *Curva de nível para um único valor de função*

O último argumento tem que conter pelo menos dois elementos, mas se se pretender a circunferência de raio 1 centrada na origem:

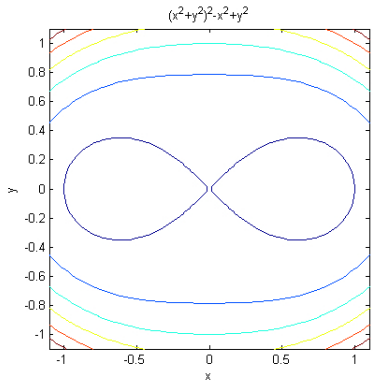
```
>> contour(x,y,x.^2+y.^2,[1 1])
```



Pode usar-se também o commando `EZCONTOUR` para representar curvas de nível de funções. Neste caso não é necessário recorrer a `MESHGRID`. No entanto, para se obter apenas uma curva tem que se recorrer a `CONTOUR`. `EZCONTOUR(FUN)` desenha as curvas de nível da função `FUN` usando a função `CONTOUR`. As curvas são desenhadas por defeito no domínio $-2\pi < x < 2\pi$, $-2\pi < y < 2\pi$ e usando uma grelha de 60×60. Para se alterar o domínio onde se pretende representar a função `FUN` basta usar `EZCONTOUR(FUN,[XMIN,XMAX,YMIN,YMAX])`, em que `XMIN`$< x <$`XMAX` e `YMIN`$< x <$`YMAX`, ou `EZCONTOUR(FUN,[A,B])`, em que `A`$< x <$`B` e `A`$< x <$`B`. Para se alterar a grelha deve usar-se `EZCONTOUR(...,N)`, definindo este comando uma grelha `N`$\times$`N`.

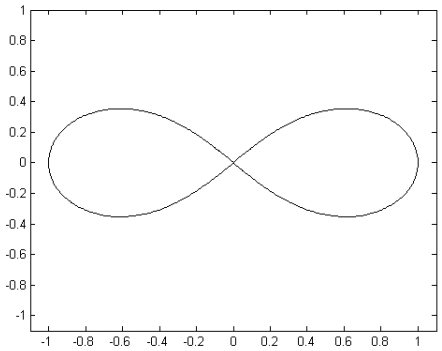
Exemplo 1.1.12 *Curvas de nível com o comando* `EZCONTOUR`

```
>> ezcontour('(x^2+y^2)^2-x^2+y^2',[-1.1 1.1 -1.1 1.1]);
>> axis square
```



Exemplo 1.1.13 `>> [x,y]=meshgrid(-1.1:0.01:1.1,-1.1:0.01:1);`

```
>> z=(x.^2+y.^2).^2-x.^2+y.^2;
>> contour(x,y,z,[0 0],'k')
```

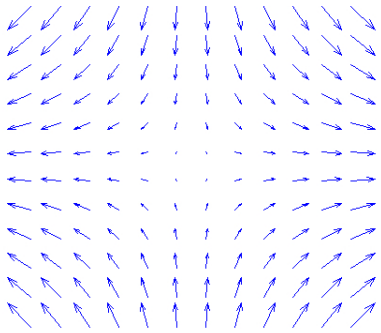


1.1.8 Campos de vectores

A função `quiver` é usada para representar campos de vectores ou *arrays* de setas. As setas podem localizar-se m pontos específicos igualmente espaçados no plano caso as coordenadas x e y não sejam dadas explicitamente, ou podem ser colocadas em localizações específicas.

Exemplo 1.1.14 `>> [x,y]=meshgrid(-1.1:.2:1.1,-1.1:.2:1.1);`

```
>> quiver(x,-y);
>>axis equal;
>>axis off
```



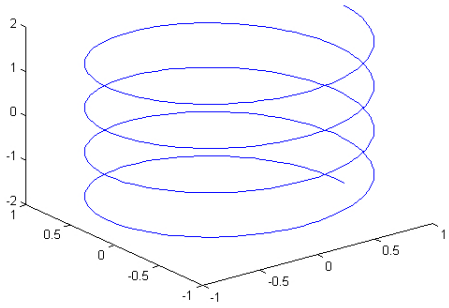
1.2 Gráficos 3d

1.2.1 Curvas em espaço tridimensional

Para desenhar curvas em 3 dimensões, usa-se o comando básico `plot3`, que é semelhante ao `plot`. A diferença é que usa 3 coordenadas (x, y, z) em vez de 2 (x, y).

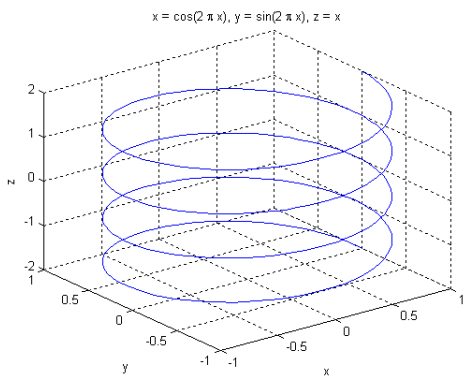
Exemplo 1.2.1 *Para desenhar uma hélice,*

```
>> t=-2:0.01:2;
>> plot3(cos(2*pi*t),sin(2*pi*t),t)
```



ou

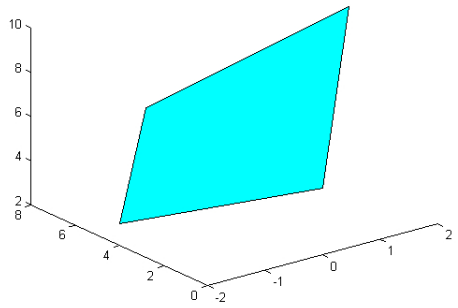
```
>> ezplot3('cos(2*pi*x)', 'sin(2*pi*x)' ,'x',[-2,2])
```



1.2.2 Polígono a 3 dimensões

Para representar um polígono a 3 dimensões usa-se o comando `fill3` de forma similar ao `fill`, mas neste caso têm-se 4 argumentos, em que o quarto é a cor.

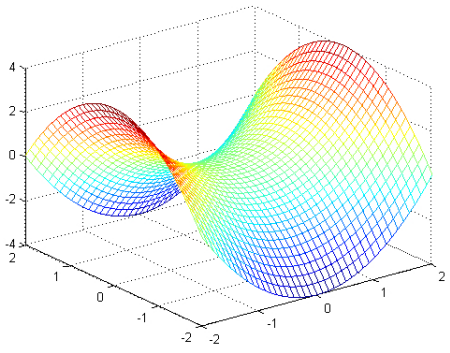
Exemplo 1.2.2
```
>> x=[-2 0 2 0 -2];
>> y=[4 8 4 0 4];
>> z=[3 5 10 5 3];
>> fill3(x,y,z,'c') %c representa a cor cyan
```



1.2.3 Superfícies em espaço tridimensional

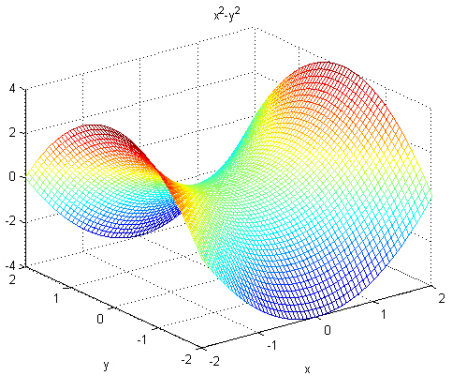
Os comandos básicos para desenhar superfícies 3d são `mesh` e `surf`. O primeiro produz uma superfície transparente e o segundo uma opaca. Podem ser usadas de duas formas. A coordenada z é dada como função de x e y, ou z não pode ser representado explicitamente em função de x e y e as coordenadas, x, y e z, podem ser dadas como superfícies paramétricas em função de outros 2 parâmetros. Em qualquer dos casos tem que se começar por definir uma grelha através do comando `meshgrid`. Em alternativa podem usar-se os comandos `ezmesh` e `ezsurf`.

Exemplo 1.2.3
```
>> [x,y]=meshgrid(-2:0.1:2,-2:0.1:2);
>> z=x.^2-y.^2;
>> mesh(x,y,z)
```



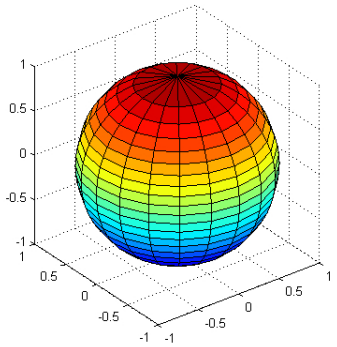
ou

```
>>ezmesh('x^2-y^2',[-2,2],[-2,2])
```



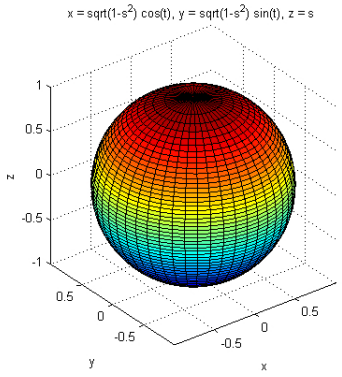
Exemplo 1.2.4 *Pretende representar-se a esfera de equação* $x^2 + y^2 + z^2 = 1$. *Uma forma de representar esta esfera é tomando como parâmetro a coordenada vertical* z *e a coordenada polar* θ *no plano* $x - y$. *Se* r *representar a distância ao eixo do* z, *a equação da esfera vem* $r^2 + z^2 = 1$ *ou* $r = \sqrt{1 - z^2}$. *Logo,* $x = \sqrt{1 - z^2}\cos\theta$ *e* $y = \sqrt{1 - z^2}\sin\theta$.

```
>> [theta,z]=meshgrid((0:0.1:2)*pi,(-1:0.1:1));
>> x=sqrt(1-z.^2).*cos(theta);
>> y=sqrt(1-z.^2).*sin(theta);
>> surf(x,y,z);
>> axis square
```



ou

```
>>ezsurf('sqrt(1-s^2)*cos(t)','sqrt(1-s^2)*sin(t)','s',[-1,1,0,2*pi]);
>> axis equal
```



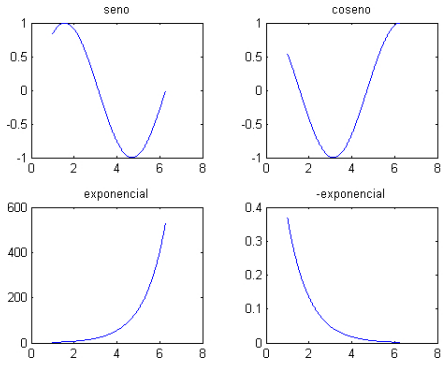
1.3 Efeitos especiais

1.3.1 Partição de uma figura em várias janelas

Uma figura pode dividir-se em m partições horizontais e em n verticais, de forma a que cada subdivisão tenha os seus próprios eixos. É utilizada, para este efeito, a função `subplot(m,n,p)`, em que p representa a subdivisão activa.

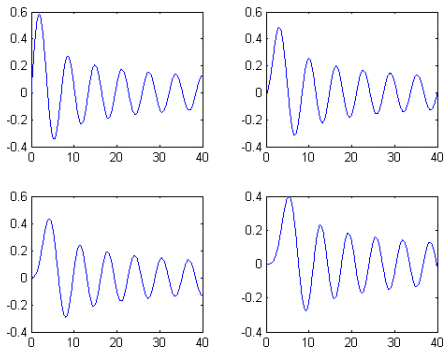
Exemplo 1.3.1 `>> x=1:0.01:2*pi;`

```
>> y1=sin(x);
>> y2=cos(x);
>> y3=exp(x);
>> y4=exp(-x);
>> subplot(2,2,1),plot(x,y1),title('seno')
>> subplot(2,2,2),plot(x,y2),title('coseno')
>> subplot(2,2,3),plot(x,y3),title('exponencial')
>> subplot(2,2,4),plot(x,y4),title('-exponencial')
```



Exemplo 1.3.2 *Para construir uma figura com as quatro primeiras funções de Bessel,*

```
>> x=[0:0.05:40];
>> for j=1:4
subplot(2,2,j)
plot(x,besselj(j*ones(size(x)),x))
end
```



```
>> subplot(1,1,1) %para se voltar a uma única janela
```

1.3.2 Animações

A forma mais simples de criar uma figura animada é com o comando `comet`, que produz uma figura paramétrica de uma curva, tal como o comando `plot`, mas vê-se a curva a ser traçada ao longo do tempo.

Exemplo 1.3.3
```
>> t=0:0.01*pi:2*pi;
>> figure;axis equal;axis([-1 1 -1 1]); hold on
>> comet(cos(t),sin(t))
```

Neste caso vê-se um movimento circular.

Para animações mais complexas podem usar-se os comandos `getframe` e `movie`. O primeiro capta a janela da figura activa para uma *frame* do filme e o segundo mostra o resultado.

Exemplo 1.3.4
```
>> x=[0:0.01:1];
>> for j=0:50
plot(x,sin(j*pi/5)*sin(pi*x)),axis([0,1,-2,2])
M(j+1)=getframe;
end
>> movie(M)
```

1.3.3 Mudar o aspecto do fundo de uma figura

Exemplo 1.3.5 *Criar um tabuleiro de xadrez:*

```
>> white=[1 1 1]; %RGB - branco 100% de vermelho,verde e azul
>> gray=0.7*white;
>> a=[0 1 1 0];
>> b=[0 0 1 1];
>> c=[1 1 1 1];
>> figure;
>> hold on
>> for k=0:1
for j=0:2:6
fill(a'*c+c'*(0:2:6)+k,b'*c+j+k,gray)
end
end
>> plot(8*a',8*b','k')
>> set(gca,'XTickLabel',[],'YTickLabel',[]) %gca - get current axes
>> set(gcf,'Color',white) % estrutura com conjunto de propriedades
>> axis square
```