Universidade Estadual de Campinas
Faculdade de Engenharia Mecânica
Departamento de Projeto Mecânico
Laboratório de Dinâmica de Estruturas e Máquinas

Guia Prático de Matlab

Capítulo II

Aspectos Práticos

Docente Responsável:

Prof. Dr. Milton Dias Jr.

Autores:

Hugo Heidy Miyasato
Leonardo Gimenes
Vinícius Gabriel Segala Simionatto

Campinas, 2009

Seção 2.1 – Plotar gráficos

Com uma ferramenta como o Matlab em mãos é possível adquirir, gerar e processar dados de forma simples e eficiente. No dia-a-dia profissional, seja como carreira acadêmica, industrial, ou mesmo estudante, é preciso representar estes dados para nós mesmos, ou outras pessoas, de forma que se possa tirar conclusões rapidamente sobre conjuntos enormes de informações.
Para isso, o Matlab possui funções que podem representar gráficos bi e tridimensionais, sejam os dados em forma de superfície, curva, pontos discretos, volume, etc.
Estas utilidades pertencentes a este ambiente serão apresentadas neste capítulo.

- ***Plot 2D***

– Plot

Com o comando *plot(dataX,dataY)* é possível plotar pontos a partir de um conjunto de dados nos eixos x e y (*dataX* e *dataY*, respectivamente). É importante ressaltar que os vetores que representam os dados em x e em y devem ter o mesmo tamanho, pois juntos, eles representam um conjunto de pontos (x,y).

Exemplo:

Plotando senóide de 1Hz: $y = \sin(2\pi t)$ de 0 a 10 s.

```
clear all
close all
clc

t0 = 0;
dt = 0.001;     %intervalo de amostragem
tf = 10;
t = t0:dt:tf; %crio vetor de tempo de 0 a 10 s

%crio sinal de seno a 1 Hz
y = sin(2*pi*1*t);
plot(t,y)
```

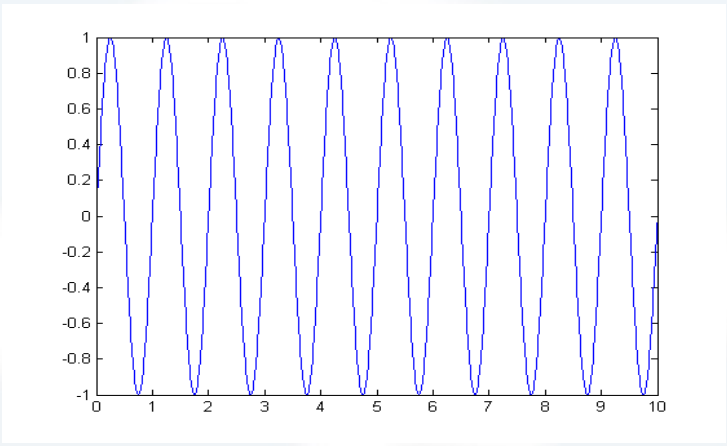


Figura 36: Gráfico plotado em tela

– Colocando título e nomeando eixos do gráfico

O gráfico pode ter um título através do comando *title('seu titulo')*, recebendo como entrada uma *string*. Com os comandos *xlabel('eixo x')* e *ylabel('eixo y')* é possível nomear os eixos x e y com o texto desejado.

Exemplo:

Plotando senóide de 1 Hz: $y = \sin(2\pi t)$ de 0 a 10 s nomeando os eixos.

```
clear all
close all
clc

t0 = 0;
dt = 0.001;     %intervalo de amostragem
tf = 10;
t = t0:dt:tf; %crio vetor de tempo de 0 a 10 s

%crio sinal de seno a 1 Hz
y = sin(2*pi*1*t);
plot(t,y)               %ploto grafico em janela
xlabel('tempo (s)')     %nomeio eixo x
ylabel('amplitude')     %nomeio eixo y
title('Figura Teste')   %titulo do grafico
```

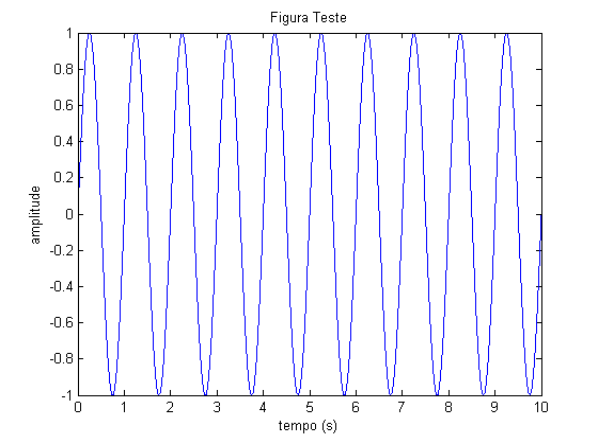


Figura 37: Exemplo de gráfico com cor e marcadores selecionados

– Alterando cor, tipo de linha ou marcador do gráfico

A codificação de cores no Matlab é feita por meio do padrão RGB. A sigla RGB vem do inglês (Red, Green, Blue), e com uma mistura destas três cores básicas, pode-se definir qualquer outra cor. Neste ambiente, a cor é definida por um vetor linha, de três colunas, cada uma especificando respectivamente a proporção de vermelho, verde e azul que se deseja na cor final. Para cada uma das cores básicas, a sua proporção na cor final deve variar de 0 a 1.
Porém, definir as cores a cada vez que se precisa delas é um processo lento e ineficaz. Por isso, o Matlab possui algumas cores padrão, que podem ser referidas por uma string, ao invés de um trio

RGB. Estas strings são chamadas de ColorSpec, e têm duas versões: o nome curto (“Short name”) e o longo (“Long name”)

A seguir temos o ColorSpec (*Color Specification*) do Matlab:

Cor	Nome curto	Nome longo	Trio RGB
azul	b	blue	[0 0 1]
verde	g	green	[0 1 0]
vermelho	r	red	[1 0 0]
ciano	c	cyan	[0 1 1]
magenta	m	magenta	[1 0 1]
amarelo	y	yellow	[1 1 0]
preto	k	black	[0 0 0]
Branco	w	white	[1 1 1]

Além disso, é possível colocar marcadores em posições calculadas do gráfico. Isto pode ajudar quando se deseja exibir algum evento que acontece em uma região específica dos dados, por exemplo um ponto de máximo ou mínimo. Pode-se também utilizar os marcadores para desenhar o gráfio, ao invés de desenhá-lo por meio de linhas. Esta é uma maneira simples de representar dados discretos (conjunto de pontos).

A seguir são mostrados os tipos de marcadores:

Caractere	Marcador
.	ponto
o	circulo
x	marca x
+	mais (cruz)
*	estrela
s	quadrado
d	diamante
v	triangulo (para baixo)
^	triangulo (para cima)
<	triangulo (para esquerda)
>	triangulo (para direita)
p	pentágono (estrela de 5 pontas)
h	hexágono

Se necessário for, ainda é possível alterar o tipo de linha a ser traçada no gráfico.

Aqui são exibidos os tipos de linha disponíveis no Matlab:

Caractere	Tipo de Linha
-	sólida
:	pontilhada
-.	traço-ponto
--	tracejada
(none)	nenhuma linha

No comando *plot*, pode-se passar um terceiro argumento (uma string) que identificará a cor e a forma (tipo de linha ou marcador) do seu gráfico. A sintaxe desta string é muito simples. Ela deve ser iniciada pela *ColorSpec* que identifica a cor desejada (caso a cor seja padrão), e logo em seguida, deve vir o caractere, ou o conjunto de caracteres que define a forma do gráfico, seja ele um marcador ou um tipo de linha

Exemplos:

```
clear all
close all
clc

x = -2.5:0.2:4.5; % Inicializo as coordenadas x dos pontos
y = 0.25*x.^3-0.75*x.^2-1.13*x+2.65; % Inicializo as coordenadas y dos pontos

plot(x,y,'ro') % O gráfico será feito com bolas (o) vermelhas (r)
title('Meu Gráfico')
xlabel('X')
ylabel('Y')
```

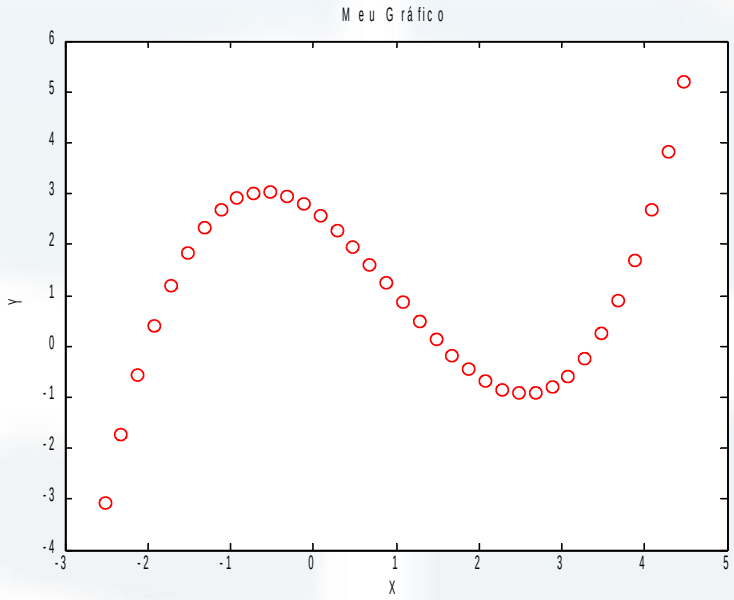


Figura 38: Exemplo de gráfico com cor e marcadores selecionados

```
clear all
close all
clc

x = -2.5:0.2:4.5; % Inicializo as coordenadas x dos pontos
y = 0.25*x.^3-0.75*x.^2-1.13*x+2.65; % Inicializo as coordenadas y dos pontos

plot(x,y,'kp') % O gráfico será feito com pentágonos(p) pretos(k)
title('Meu Gráfico')
xlabel('X')
ylabel('Y')
```

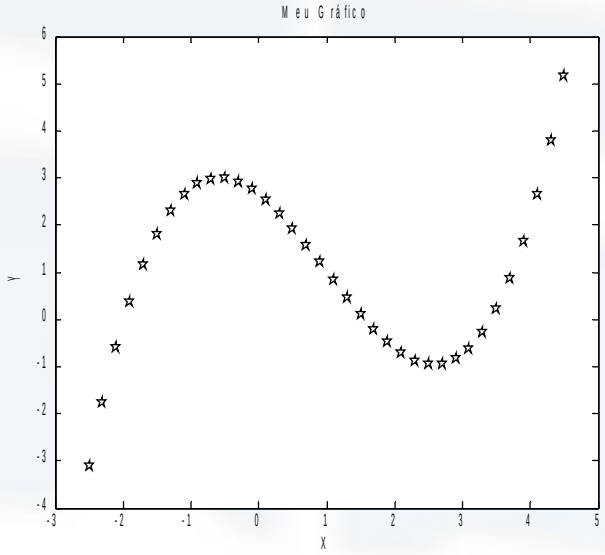


Figura 39: Exemplo de gráfico com cor e marcadores selecionados

```
clear all
close all
clc

x = -2.5:0.2:4.5; % Inicializo as coordenadas x dos pontos
y = 0.25*x.^3-0.75*x.^2-1.13*x+2.65; % Inicializo as coordenadas y dos pontos

plot(x,y,'m*') % O gráfico será feito com estrelas(*) magenta(m)
title('Meu Gráfico')
xlabel('X')
ylabel('Y')
```

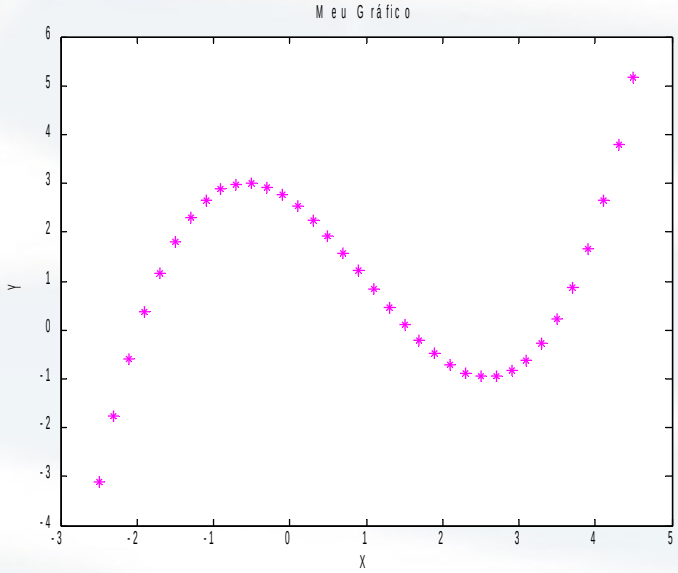


Figura 40: Exemplo de gráfico com cor e marcadores selecionados

```
clear all
close all
clc

x = -2.5:0.2:4.5; % Inicializo as coordenadas x dos pontos
y = 0.25*x.^3-0.75*x.^2-1.13*x+2.65; % Inicializo as coordenadas y dos pontos

plot(x,y,'g.-') % O gráfico será feito com linha traço-ponto(.-) verde(g)
title('Meu Gráfico')
xlabel('X')
ylabel('Y')
```

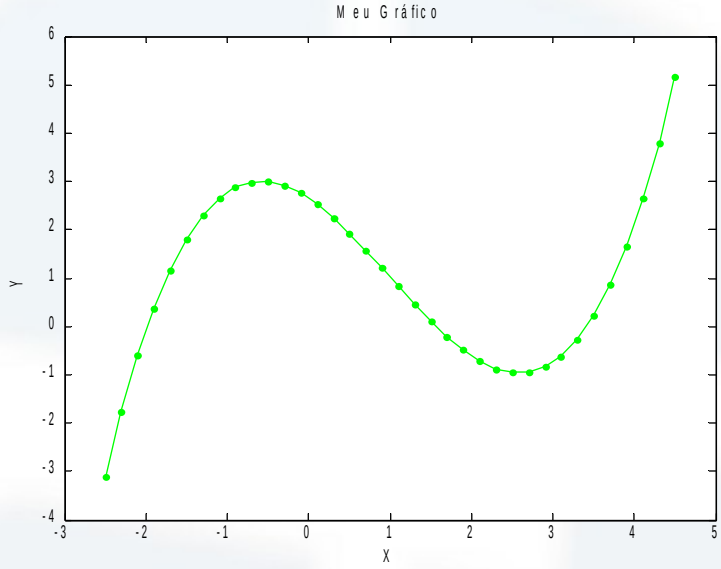


Figura 41: Exemplo de gráfico com cor e linha selecionados

Caso essa string tenha só uma das informações, o Matlab escolhe a outra como padrão. A cor padrão para gráficos no Matlab é o azul, e a forma ou tipo de linha padrão é a linha lisa.

Exemplos:

Definindo somente a cor do gráfico

```
clear all
close all
clc

x = -2.5:0.2:4.5; % Inicializo as coordenadas x dos pontos
y = 0.25*x.^3-0.75*x.^2-1.13*x+2.65; % Inicializo as coordenadas y dos pontos

plot(x,y,'g') % O gráfico será feito com linha verde(g)
title('Meu Gráfico')
xlabel('X')
ylabel('Y')
```

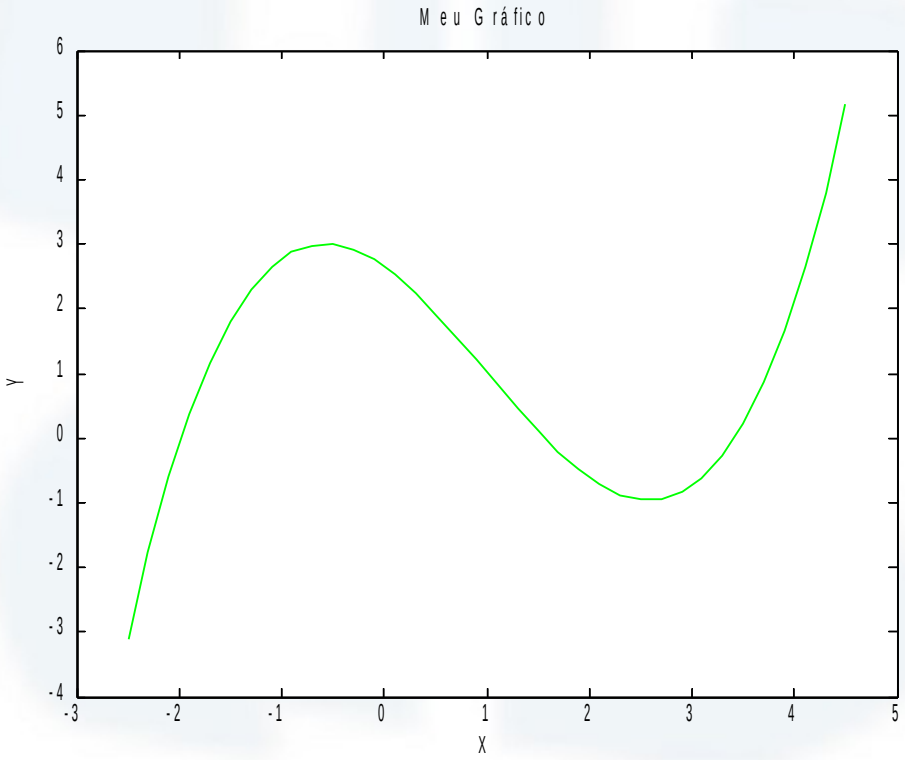


Figura 42: Exemplo de gráfico com cor selecionada

– Definindo somente a forma da linha:

```
clear all
close all
clc

x = -2.5:0.2:4.5; % Inicializo as coordenadas x dos pontos
y = 0.25*x.^3-0.75*x.^2-1.13*x+2.65; % Inicializo as coordenadas y dos pontos

plot(x,y,'.-') % O gráfico será feito com linha traço-ponto(.-) azul
title('Meu Gráfico')
xlabel('X')
ylabel('Y')
```

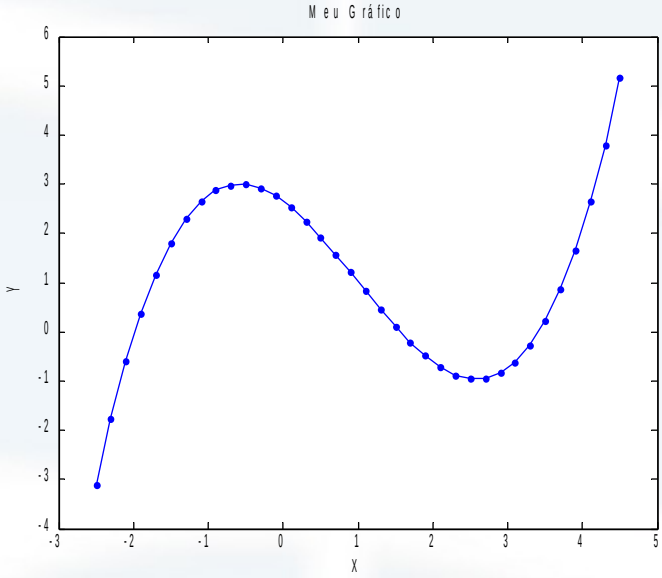


Figura 43: Exemplo de gráfico com tipo de linha selecionada

No Matlab, também é possível plotar duas curvas no mesmo gráfico. Com isto, é possível mostrar conjuntos de dados diferentes sob os mesmos eixos, para comparação, ou ainda salientar informações sobre o próprio gráfico que está sendo mostrado.
Para plotar dois ou mais conjuntos de dados, utilizando a função *plot*, deve-se passar os dados em trios, da forma:

plot(x1, y1, css1, x2, y2, css2, ..., xn, yn, cssn)

Onde:

- x1, ..., xn são os vetores x dos conjuntos de pontos 1, …,n ;
- y1, ..., yn são os vetores y dos conjuntos de pontos 1, …,n ;
- css1, ..., cssn são as strings com a colorspec e a forma de linha dos conjuntos de pontos 1, …,n ;

Exemplos:

Mostrando um polinômio e sua derivada no mesmo gráfico:

```
clear all
close all
clc

x = -2.5:0.2:4.5; % Inicializo as coordenadas x dos pontos do polinômio
y = 0.25*x.^3-0.75*x.^2-1.13*x+2.65; % Inicializo as coordenadas y dos pontos
do polinômio

dx = -2.5:0.2:4.5; % Inicializo as coordenadas x dos pontos da derivada do
polinômio
dy = 3*0.25*x.^2-2*0.75*x.^1-1.13; % Inicializo as coordenadas y dos pontos da
derivada do polinômio

plot(x,y,'b',dx,dy,'r') % O gráfico do polinômio será azul e o de sua derivada
será vermelho
title('Meu Gráfico')
xlabel('X')
ylabel('Y')
```

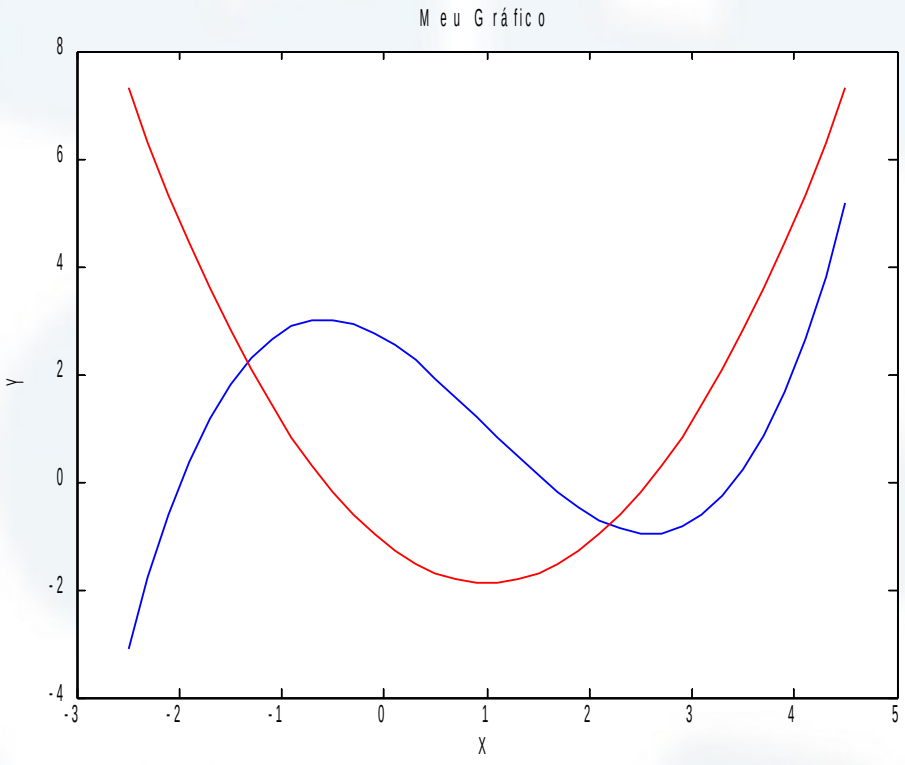


Figura 44: Exemplo mostrando um polinômio e sua derivada no mesmo gráfico

Mostrando os pontos de máximo e mínimo locais do polinômio:

```
clear all
close all
clc

x = -2.5:0.2:4.5; % Inicializo as coordenadas x dos pontos do polinômio
y = 0.25*x.^3-0.75*x.^2-1.13*x+2.65; % Inicializo as coordenadas y dos pontos
do polinômio

% coeficientes da derivada deste polinômio:

a = 3*0.25;
b = -2*0.75;
c = -1.13;

% Calculo onde a derivada é nula:

x1  = (-b + sqrt(b^2-4*a*c))/(2*a);
x2  = (-b - sqrt(b^2-4*a*c))/(2*a);

% Calculo o valor do polinômio nestes pontos

y1 = 0.25*x1.^3-0.75*x1.^2-1.13*x1+2.65;
y2 = 0.25*x2.^3-0.75*x2.^2-1.13*x2+2.65;

plot(x,y,'b',x1,y1,'ro',x2,y2,'ro') % O gráfico do polinômio será azul e
                                    % os pontos de máximo e mínimo locais
                                    % serão bolas vermelhas.
title('Meu Gráfico')
xlabel('X')
ylabel('Y')
```

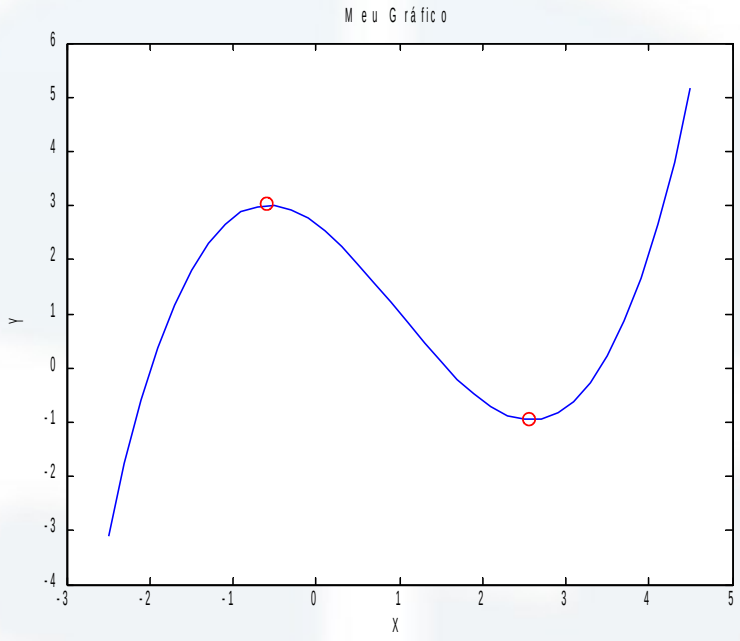


Figura 45: Exemplo mostrando os pontos de máximo e mínimo locais do polinômio

– Hold

Outra maneira de se plotar várias curvas no mesmo gráfico é utilizando o comando *hold*. No Matlab, o último eixo (de gráficos) gerado, é o eixo sobre o qual serão executados os comandos de eixo, como por exemplo o *hold*.
O fato é que, se executarmos o comando *plot*, e não houver nenhuma janela aberta, ele criará uma nova janela (no Matlab, chamada de *figure*), e nela criará novos eixos, onde desenhará seu gráfico.
Caso a janela já exista, e executemos o comando *plot* novamente, ele apagará tudo o que está na janela, criará novos eixos e desenhará o novo gráfico. Nesta etapa, não é isto o que queremos. Queremos, na verdade, que o último eixo criado antes deste comando permaneça na janela, e que outro gráfico seja desenhado sobre ele.
Para que isso seja feito, temos que digitar *"hold on"* depois do comando *plot*. Assim, o último eixo criado não será apagado, e quando o comando *plot* for chamado novamente, ele desenhará o novo gráfico sobre os eixos antigos.
Depois que desenhamos todos os gráficos que desejávamos sobre os mesmos eixos, devemos digitar o comando *"hold off"* para que possamos gerar novas janelas, onde colocaremos novos eixos e novos gráficos.

Exemplos:

Utilizando o comando *plot* duas vezes sem utilizar o hold

```
clear all
close all
clc

x = -2.5:0.01:4.5;
y = 2*atan(5*x)/pi;
z = sign(x);


plot(x,y,'b') % O gráfico da função atan será azul

plot(x,z,'r') % O gráfico da função sign será vermelho

title('Meu Gráfico')
xlabel('X')
ylabel('Y')
```

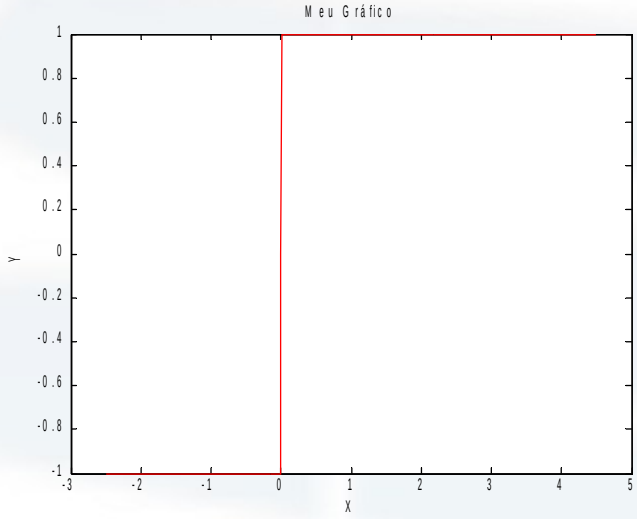


Figura 46: Exemplo utilizando o comando plot duas vezes sem utilizar o hold

Note que somente o gráfico da função *sign* ficou na janela. O gráfico da função *atan* foi apagado para que este fosse desenhado.

Agora, utilizando *hold on* e *hold off* :

```
clear all
close all
clc

x = -2.5:0.01:4.5;
y = 2*atan(5*x)/pi;
z = sign(x);

plot(x,y,'b') % O gráfico da função atan será azul
hold on
plot(x,z,'r') % O gráfico da função sign será vermelho
hold off

title('Meu Gráfico')
xlabel('X')
ylabel('Y')
```

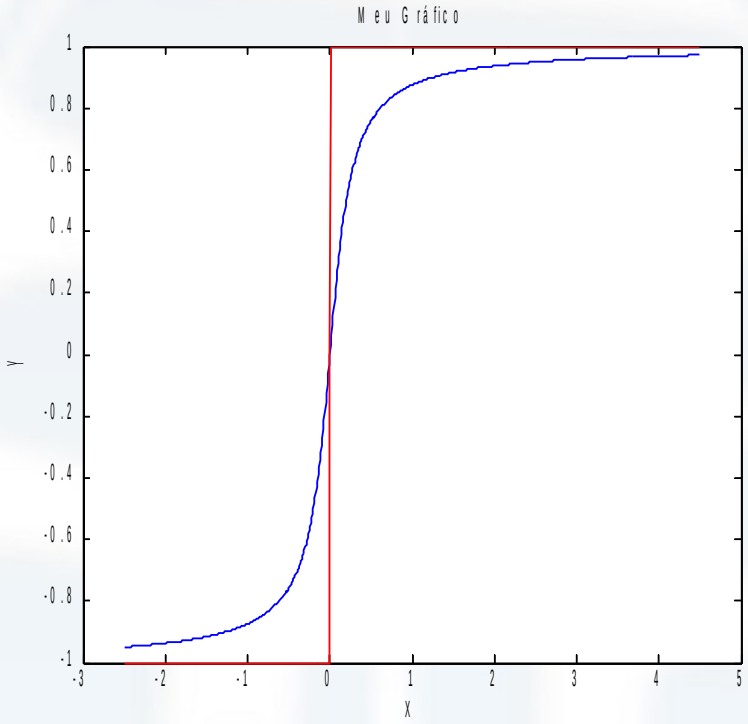


Figura 47: Exemplo utilizando utilizando *hold on* e *hold off*

Agora, plotando múltiplas funções com o comando *hold:*

```
clear all
close all
clc

x = -2.5:0.01:4.5; % Inicializo as coordenadas x dos pontos a plotar
y1 = 2*atan(2*x)/pi; % Inicializo as coordenadas y dos pontos de cada curva
y2 = 2*atan(5*x)/pi;
y3 = 2*atan(10*x)/pi;
y4 = 2*atan(20*x)/pi;
y5 = 2*atan(50*x)/pi;

plot(x,y1,'b') % curva azul

hold on % Comando hold on somente após a criação do eixo desejado,
        % criado pelo primeiro plot.

plot(x,y2,'g') % curva verde
plot(x,y3,'r') % curva vermelha
plot(x,y4,'k') % curva preta
plot(x,y5,'m') % curva magenta

hold off % Comando hold off depois que já foram desenhados os gráficos.

title('Meu Gráfico')
xlabel('X')
ylabel('Y')
```

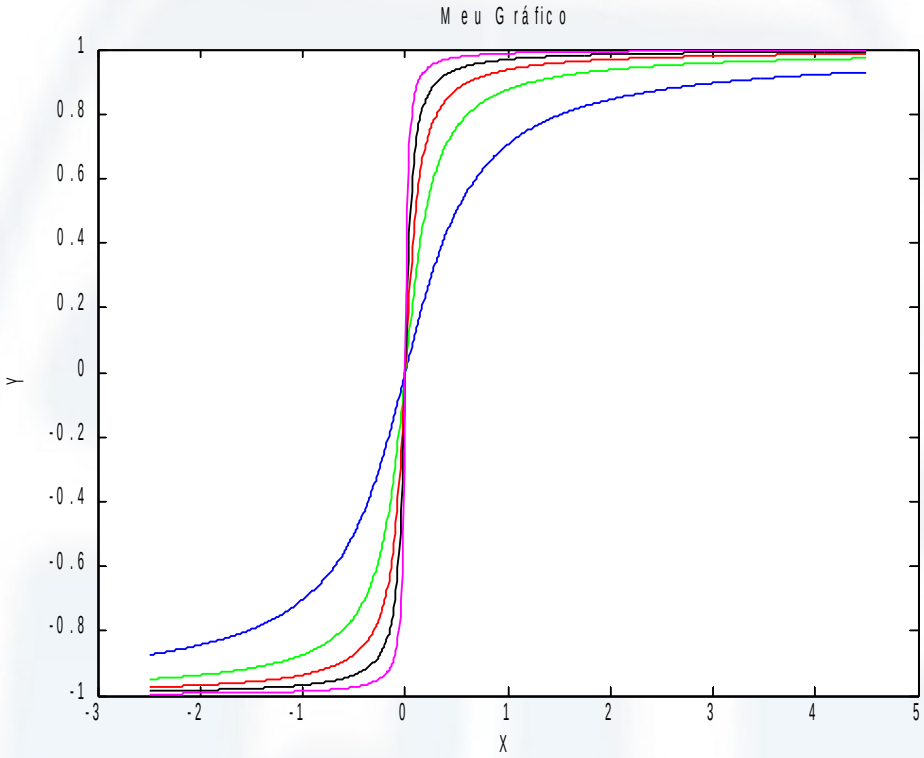


Figura 48: Exemplo utilizando utilizando *hold on* e *hold off*

IMPORTANTE: Observe que como o eixo *x* é comum a todas as curvas, foi utilizado somente um vetor *x* e cinco vetores *y*, um para cada curva.

Lembre-se: Evitar criar vetores com informações redundantes, como no exemplo acima, pode ajudar a economizar a memória disponível de seu computador, evitando que seu Matlab pare de funcionar por falta de memória, e fazendo com que seu programa não desperdice tempo executando comandos desnecessários.

– xlim, ylim

Quando plotamos um gráfico, o Matlab escolhe os limites dos eixos do gráfico, de forma que a visualização de todos os conjuntos de dados que pretendemos exibir fique adequada, ou seja, que todos sejam exibidos na tela por inteiro. Porém, estes limites podem não ser os melhores para a apresentação que se pretende fazer. Pode-se desejar uma visão mais aproximada dos dados, ou mais distante.
Para alterar os limites dos eixos do gráfico, usamos as funções *xlim* e *ylim*. Em ambas, devemos passar como argumento um vetor linha de dias colunas. A primeira indicará o limite mínimo do eixo em questão, e a segunda, o máximo.

Exemplo:

Plotando uma onda quadrada:

```
clear all
close all
clc

x = 0:0.001:10;
y = square(2*pi*x,50);

plot(x,y)

title('Meu Gráfico')
xlabel('X')
ylabel('Y')
```

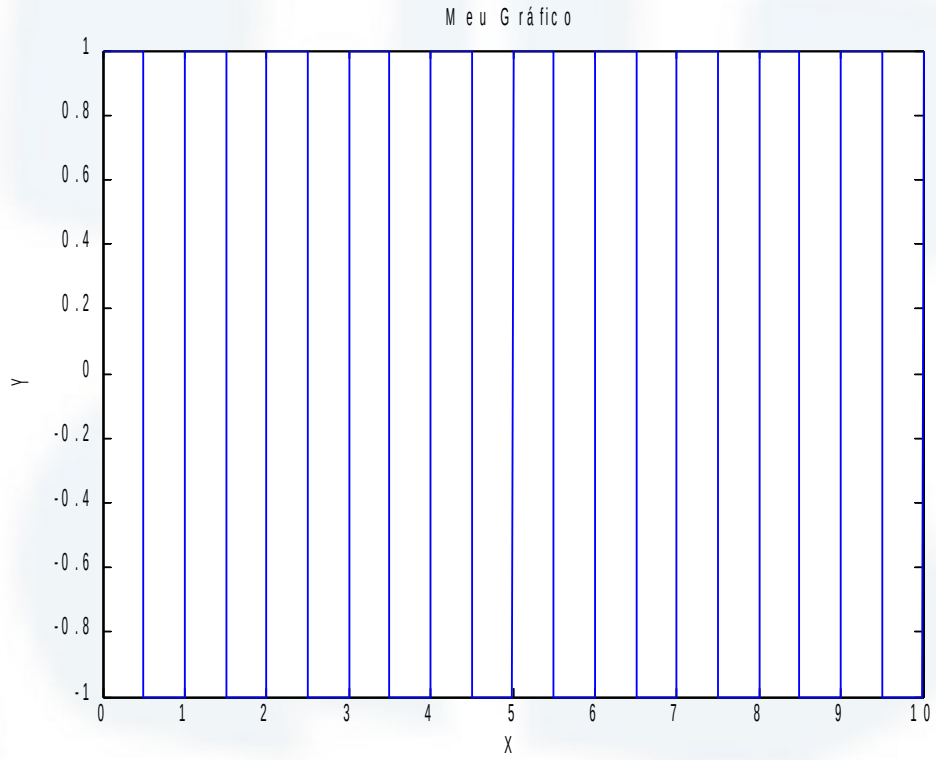


Figura 49: Plotando onda quadrada

Como podemos observar, os limites do eixo *y* escolhidos pelo Matlab permitem a visualização do sinal por completo, mas de maneira comprometida.

Alterando os limites do gráfico:

```
clear all
close all
clc

x = 0:0.001:10;
y = square(2*pi*x,50);

plot(x,y)

title('Meu Gráfico')
xlabel('X')
ylabel('Y')
ylim([-1.5 1.5]) % Aumento os limites do eixo Y
```

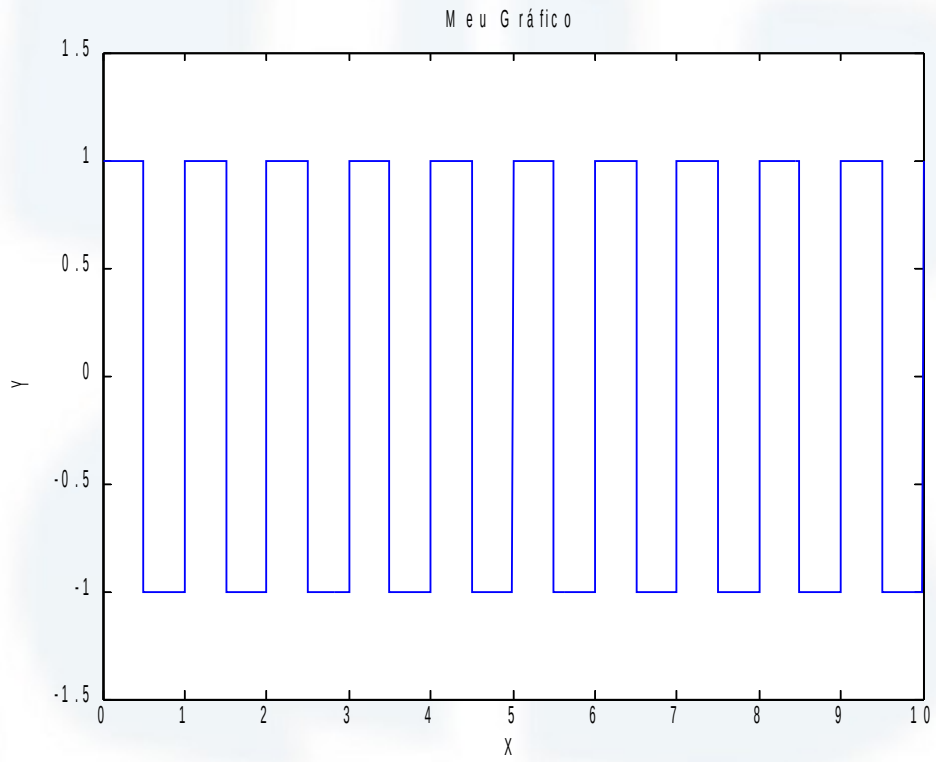


Figura 50: Plotando onda quadrada alterando limites do gráfico

Observe como a visibilidade da curva melhorou com a mudança introduzida.

Agora, aproximando a vista da parte central do gráfico:

```
clear all
close all
clc

x = 0:0.001:10;
y = square(2*pi*x,50);

plot(x,y)

title('Meu Gráfico')
xlabel('X')
ylabel('Y')
xlim([3.75 5.75])% Reduzo os limites do eixo X para aproximar
ylim([-1.5 1.5]) % Aumento os limites do eixo Y
```

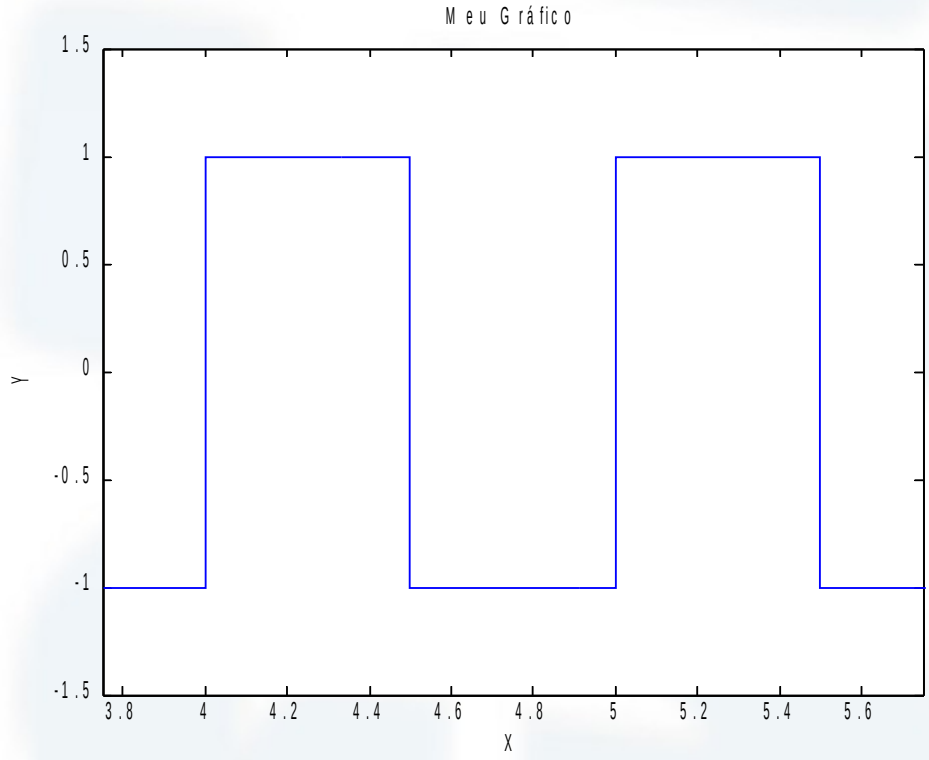


Figura 51: Plotando onda quadrada aproximando a vista da parte central do gráfico

– Grid

De maneira simples, porém muito útil, também está disponível uma grade (ou *grid*) para que se possa ler adequadamente os dados dos gráficos. Para que se possa acionar o *grid* no gráfico, o comando é *"grid on"*. Para retirar o *grid,* o comando é *"grid off"*.

Exemplo:

```
clear all
close all
clc

x = 0:0.001:10;
y = sin(pi*x./2);

plot(x,y)

grid on            % Ligando o grid
title('Meu Gráfico')
xlabel('X')
ylabel('Y')
```

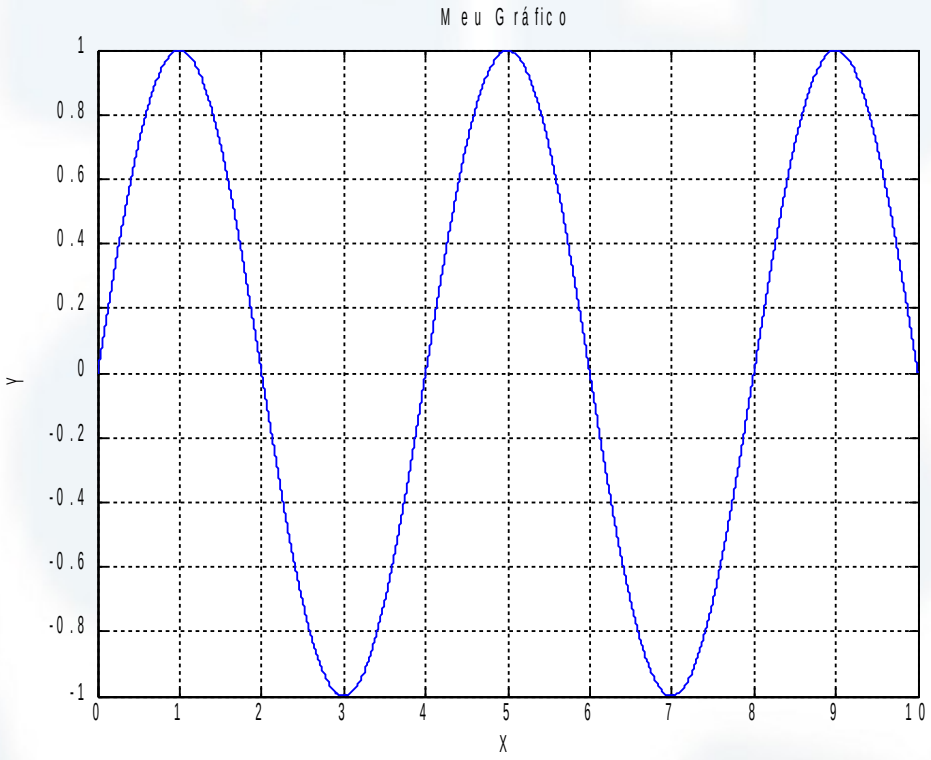


Figura 52: Plotando onda quadrada mostrando as linhas de grid

– Legend

Quando se plota muitas curvas no mesmo gráfico, é interessante que se possa inserir legendas no mesmo. O Matlab também tornou isto possível. Com o comando *legend,* basta passar como argumentos strings com o nome dos sinais, na ordem em que eles foram desenhados no gráfico, que a legenda surgirá automaticamente.

Exemplos:

Usando o comando *hold*

```
clear all
close all
clc

x = -2.5:0.2:4.5; % Inicializo as coordenadas x dos pontos a plotar
y1 = 2*atan(2*x)/pi; % Inicializo as coordenadas y dos pontos de cada curva
y2 = 2*atan(5*x)/pi;
y3 = 2*atan(10*x)/pi;
y4 = 2*atan(20*x)/pi;
y5 = 2*atan(50*x)/pi;

plot(x,y1,'b.') % curva azul

hold on % Comando hold on somente após a criação do eixo desejado,
        % criado pelo primeiro plot.

plot(x,y2,'g-.') % curva verde
plot(x,y3,'ro') % curva vermelha
plot(x,y4,'k+') % curva preta
plot(x,y5,'m.-') % curva magenta

hold off % Comando hold off depois que já foram desenhados os gráficos.

title('Meu Gráfico')
xlabel('X')
ylabel('Y')

legend('Azul','Verde','Vermelha','Preta','Magenta')
```

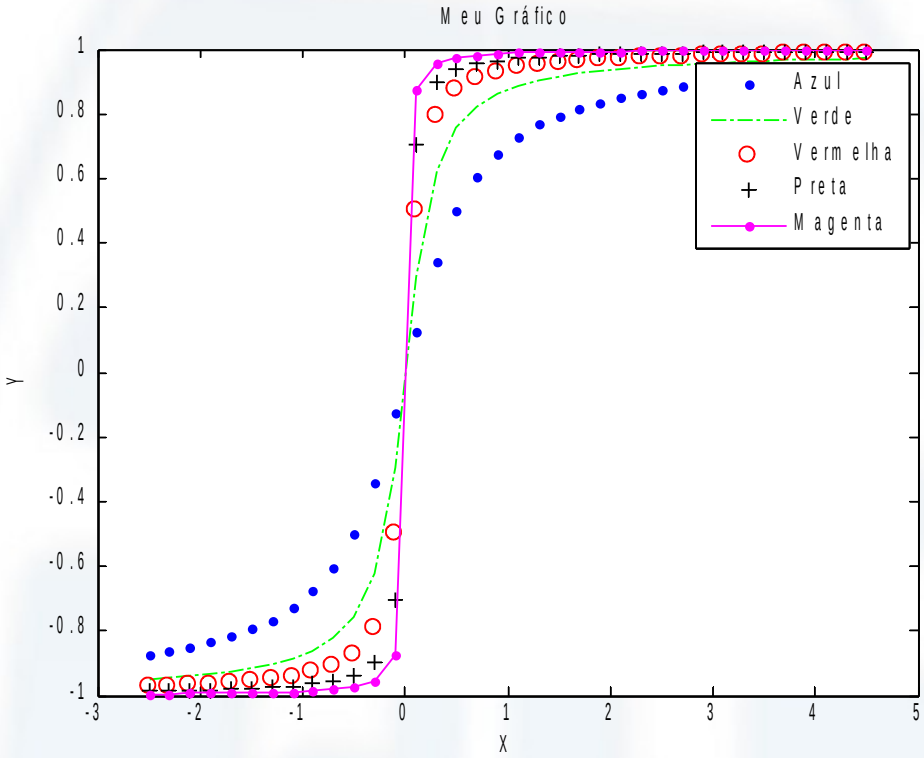


Figura 53: Exemplo acrescentando legenda ao gráfico, plotando com hold

Usando somente o comando *plot:*

```
clear all
close all
clc

x = -2.5:0.2:4.5; % Inicializo as coordenadas x dos pontos a plotar
y1 = 2*atan(2*x)/pi; % Inicializo as coordenadas y dos pontos de cada curva
y2 = 2*atan(5*x)/pi;
y3 = 2*atan(10*x)/pi;
y4 = 2*atan(20*x)/pi;
y5 = 2*atan(50*x)/pi;

plot(x,y1,'b.',x,y2,'g-.',x,y3,'ro',x,y4,'k+',x,y5,'m.-') % Todas as curvas

title('Meu Gráfico')
xlabel('X')
ylabel('Y')

legend('Azul','Verde','Vermelha','Preta','Magenta')
```

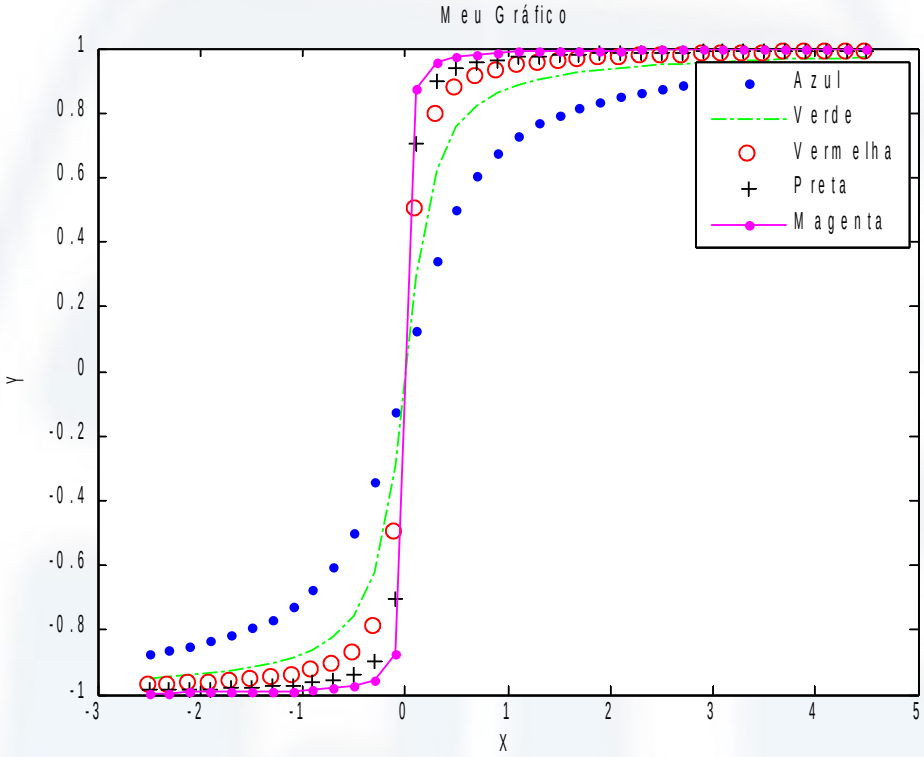


Figura 54: Exemplo acrescentando legenda ao gráfico, plotando com plot

– Subplot

Normalmente, há informações que vem aos pares, mas que não possuem as mesmas unidades, como por exemplo amplitude e fase de um sinal de resposta em frequência, ou velocidade e deslocamento de um grau de liberdade.
Uma maneira simples de apresentar estas informações, seria colocar estes gráficos na mesma janela, mas em eixos distintos. Isto pode ser feito com a ajuda do comando *subplot.*
Este comando possui uma sintaxe simples, que torna seu uso muito flexível, e por isso vamos explorá-la com cuidado para podermos extrair deste comando toda a potencialidade que ele possui.
Basicamente, este comando divide a janela (*figure*) em *l* linhas e em *c* colunas, e na posição desejada, ele cria um eixo de coordenadas, onde o gráfico será desenhado.
A seleção da posição onde se deseja desenhar o gráfico é feita por meio de um número inteiro. Quando se define o número de linhas e colunas da matriz de divisão da *figure* , as posições desta matriz são enumeradas da esquerda para a direita e de cima para baixo, conforme a figura abaixo:

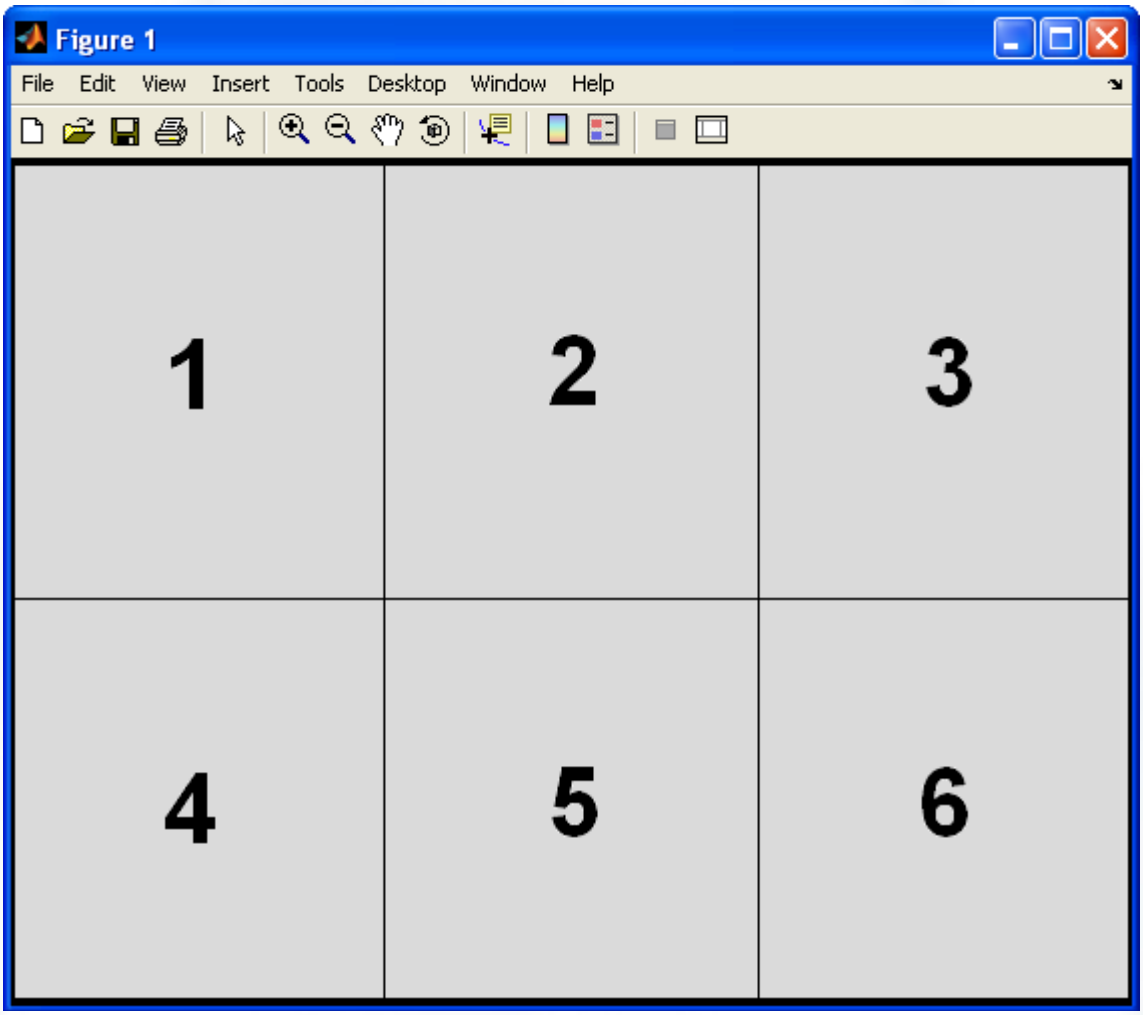


Figura 55: Número de linhas e colunas da matriz de divisão da figure

Neste caso, foi escolhido dividir a *figure* em duas linhas e três colunas. Os números em cada subdivisão da janela são referentes à indexação de cada posição, que iremos acessar adiante para inserir os gráficos na posição desejada.

A sintaxe deste comando é a seguinte:

subplot(l,c,n)

Onde:

- *l* é o número de linhas em que a figure será dividida
- *c* é o número de colunas em que a figure será dividida
- *n* é a posição onde deve ser criado o gráfico

Assim, para plotar vários gráficos na mesma janela, teremos que utilizar o comando *subplot* antes de cada comando *plot*. Isto se deve ao fato de a divisão que o *subplot* executa na *figure* ser válida somente para o comando *plot* subsequente, e não para todos os outros *plot* que serão executados nesta janela. Isto permite que se possa plotar gráficos de tamanhos diferentes na mesma janela.

Exemplos:

Utilizando o mesmo número de linhas e colunas em todos os *subplot:*

```
clear all
close all
clc

x = -2.5:0.2:4.5;
y1 = 2*atan(2*x)/pi;
y2 = 2*atan(5*x)/pi;
y3 = 2*atan(10*x)/pi;
y4 = 2*atan(20*x)/pi;
y5 = 2*atan(50*x)/pi;
y6 = 2*atan(500*x)/pi;

subplot(2,3,1) % Divido a Figura em uma matriz 2x3 e acesso a posição 1
plot(x,y1,'ro')
title('Bolas Vermelhas')
subplot(2,3,2) % Divido a Figura em uma matriz 2x3 e acesso a posição 2
plot(x,y2,'go')
title('Bolas Verdes')
subplot(2,3,3) % Divido a Figura em uma matriz 2x3 e acesso a posição 3
plot(x,y3,'ko')
title('Bolas Pretas')
subplot(2,3,4) % Divido a Figura em uma matriz 2x3 e acesso a posição 4
plot(x,y4,'b.-')
title('Ponto-Traço Azul')
subplot(2,3,5) % Divido a Figura em uma matriz 2x3 e acesso a posição 5
plot(x,y5,'m-.')
title('Traço-Ponto Magenta')
subplot(2,3,6) % Divido a Figura em uma matriz 2x3 e acesso a posição 6
plot(x,y6,'c*')
title('Estrelas Ciano')
```

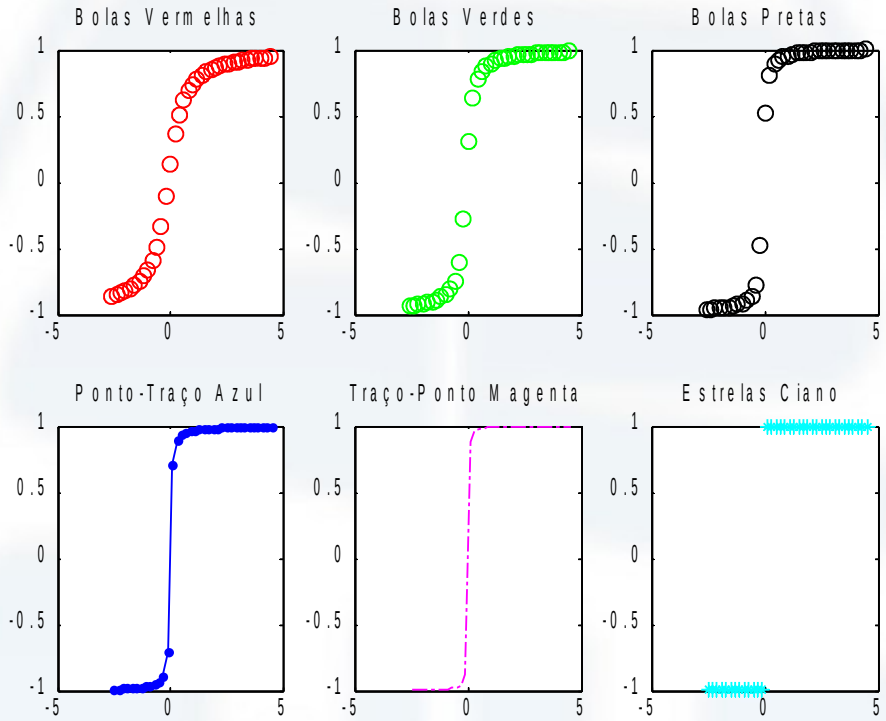


Figura 56: Utilizando o mesmo número de linhas e colunas em todos os subplot

Utilizando divisões diferentes em cada subplot:

```
clear all
close all
clc

x = -2.5:0.2:4.5;
y1 = 2*atan(2*x)/pi;
y2 = 2*atan(5*x)/pi;
y3 = 2*atan(10*x)/pi;
y4 = 2*atan(20*x)/pi;

subplot(1,2,1) % Divido a Figura em uma matriz 1x2 e acesso a posição 1
plot(x,y1,'ro')
title('Bolas Vermelhas')
subplot(2,2,2) % Divido a Figura em uma matriz 2x2 e acesso a posição 2
plot(x,y2,'go')
title('Bolas Verdes')
subplot(2,4,7) % Divido a Figura em uma matriz 2x4 e acesso a posição 7
plot(x,y3,'ko')
title('Bolas Pretas')
subplot(2,4,8) % Divido a Figura em uma matriz 2x4 e acesso a posição 8
plot(x,y4,'b.-')
title('Ponto-Traço Azul')
```

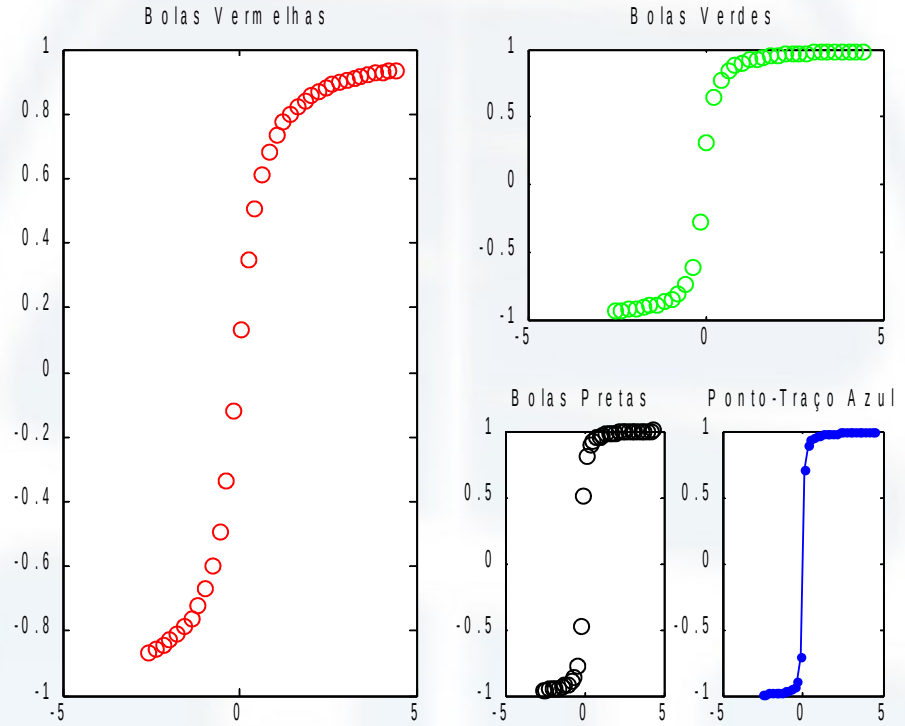


Figura 57: Utilizando divisões diferentes em cada subplot

Neste caso é possível observar que o comando *subplot* realiza as divisões na janela para que possamos inserir o gráfico no tamanho e posição desejados isto não interfere na posição dos outros gráficos já posicionados na janela.

- ***Figure***

Eventualmente, os programas desenvolvidos em Matlab geram muitos gráficos, e por isso, torna-se necessário exibir os gráficos em múltiplas janelas.
Uma questão técnica a se tratar é que, se utilizarmos o comando *plot* múltiplas vezes consecutivas, apenas uma janela será aberta, e os gráficos vão se sobrepor um a um na janela, de forma que só será possível visualizar o último dado que foi plotado.

Exemplo:

Uso do comando *plot* consecutivamente:

```
clear all
close all
clc

x = -2.5:0.1:4.5;
y1 = 2*atan(2*x)/pi;
y2 = 2*atan(5*x)/pi;
y3 = 2*atan(10*x)/pi;
y4 = 2*atan(20*x)/pi;
y5 = 2*atan(25*x)/pi;
y6 = 2*atan(50*x)/pi;

plot(x,y1,'co')
title('Bolas Ciano')
plot(x,y2,'go')
title('Bolas Verdes')
plot(x,y3,'ko')
title('Bolas Pretas')
plot(x,y4,'b.-')
title('Ponto-Traço Azul')
plot(x,y5,'m-.')
title('Traço-Ponto Magenta')
plot(x,y6,'r*')
title('Estrelas Vermelhas')
```

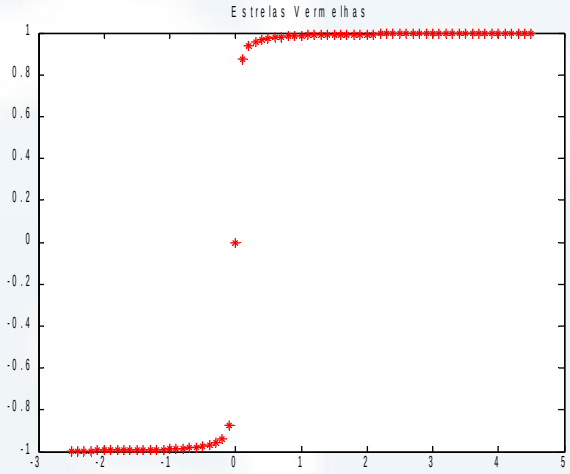


Figura 58: Uso do comando plot consecutivamente

Note que os 6 comandos *plot* foram sobrepostos, e somente o último ficou visível.

Para criar uma nova janela, utilizamos o comando *figure.* Este comando cria uma janela, onde será inserido o novo gráfico feito com a função *plot.*

IMPORTANTE: O comando *plot* gerará um gráfico sempre na última janela criada.

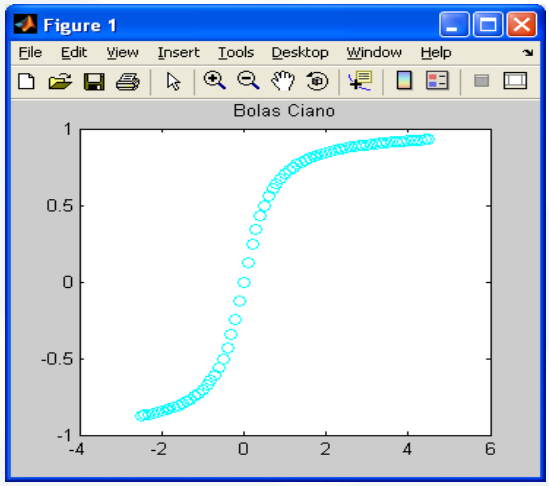
Exemplo:

```
clear all
close all
clc

x = -2.5:0.1:4.5;
y1 = 2*atan(2*x)/pi;
y2 = 2*atan(5*x)/pi;

figure % Crio uma nova janela
plot(x,y1,'co')
title('Bolas Ciano')

figure % Crio uma nova janela
plot(x,y2,'go')
title('Bolas Verdes')
```



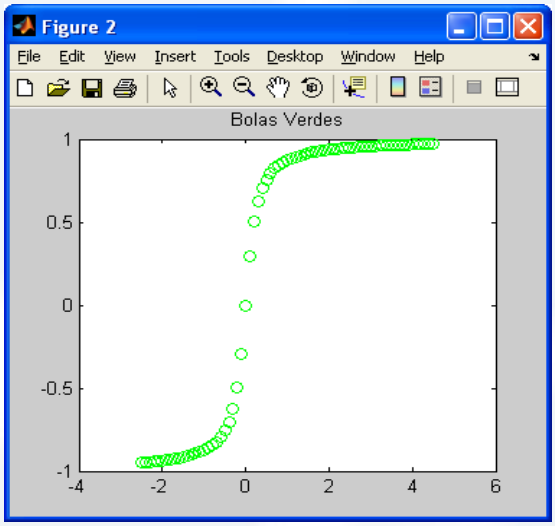


Figura 59 : Cada gráfico foi desenhado em uma janela diferente

Agora sim, cada gráfico foi desenhado em uma janela diferente, como desejávamos.

As *figures* têm muitas propriedades que podemos alterar. A maioria delas não surte efeitos diretos na plotagem dos gráficos, e por isso estas não serão detalhadas aqui. A sintaxe para se criar uma figure com a propriedades alteradas é a seguinte:

figure(prop1, val1, prop2, val2, ... ,propn, valn)

Onde:

- *prop1, ... , propn* são os nomes das propriedades a ser alteradas, de 1 a n;
- *val1, ... , valn* são os valores a ser colocados nas respectivas propriedades

Para a função *figure* é possível passar como argumento n pares propriedade-valor.

Há 3 propriedades que podem vir a ser muito úteis: *numbertitle, name* e *color.*

- NumberTitle

Esta propriedade tem dois valores possíveis: *on* ou *off*, e seu padão é *on*. Se ela estiver em *on*, o título da janela terá a string "Figure" mais o número da *figure* correspondente. Caso contrário, o título da janela não terá mais esta mensagem.

Exemplo:

```
clear all
close all
clc

figure

figure('numbertitle','off')
```

Numbertitle = on *Numbertitle = off*

Figura 60 : Numbertitle on/off

- Name

Esta propriedade representa o título que se deseja exibir na janela. Se *Numbertitle* estiver *off,* o título da janela será somente o valor passado para *name.* Caso contrário, o título da janela será a string "Figure" mais o número da janela, mais ": ", mais o nome desejado

Exemplo:

```
clear all
close all
clc

figure('name','Janela') % Neste caso, numbertitle está on

% Já neste ele será desligado
figure('numbertitle','off','name','Janela')
```

Numbertitle = on Numbertitle = off

Figura 61: Name na figure

– Color

Esta propriedade altera a cor de fundo da figure. O padrão do Matlab é o cinza, mas esta cor pode não ser a melhor para a apresentação dos resultados. Para setar as cores de fundo, podemos utilizar ColorSpecs ou trios RGB.

Exemplo:

```
clear all
close all
clc

% Cor de fundo: branco
figure('numbertitle','off','name','Janela Branca','color','w')
% Cor de fundo: vermelho
figure('numbertitle','off','name','Janela Vermelha','color','r')
% Cor de fundo: azul
figure('numbertitle','off','name','Janela Azul','color','b')
% Cor de fundo: verde
figure('numbertitle','off','name','Janela Verde','color','g')
```

Figura 62:Alterando cor da figure

Exemplo completo:

```
clear all
close all
clc

x = 0:0.001:5; % Inicializo o vetor x
y1 = exp(-0.5*x).*sin(2*pi*2.5*x); % Inicializo um dado de deslocamento y1
y1env1 = exp(-0.5*x); %Calculo sua envoltória superior
y1env2 = -y1env1; % E a envoltória inferior

% Calculo a derivada do deslocamento (velocidade)
dy1 = sqrt((2*pi*2.5)^2+(0.5)^2)*exp(-0.5*x).*cos(2*pi*2.5*x+atan(0.5 …
/(2*pi*2.5)));
%Calculo sua envoltória superior
dy1env1 = sqrt((2*pi*2.5)^2+(0.5)^2)*exp(-0.5*x);
% E a envoltória inferior
dy1env2 = -dy1env1;

%crio uma nova figura
figure('numbertitle','off','color','w','name','Velocidade e Aceleração')
subplot(2,1,1) % Divido ela em 2 linhas e 1 coluna e pego a primeira posição
plot(x,y1,'b') % Desenho ali o gráfico x por y1 azul
hold on        % Desejo desenhar mais curvas no mesmo gráfico
plot(x,y1env1,'k--') % desenho a primeira envoltória em linha preta tracejada
plot(x,y1env2,'k--') % desenho a segunda envoltória em linha preta tracejada
hold off       % Não desejo mais adicionar curvas a este gráfico
grid on        % Ligo as linhas de grade
% Coloco legenda nas curvas de deslocamento e envoltórias
legend('Deslocamento','Envoltórias')
title('Deslocamento')   % Coloco o título neste gráfico
xlabel('Tempo [s]')     % Coloco a legenda no eixo X
ylabel('Amplitude [m]') % Coloco a legenda no eixo Y

subplot(2,1,2)  % Divido ela em 2 linhas e 1 coluna e pego a segunda posição
plot(x,dy1,'r') % Desenho ali o gráfico x por dy1 vermelho
hold on         % Desejo desenhar mais curvas no mesmo gráfico
plot(x,dy1env1,'k--') % desenho a primeira envoltória em linha preta tracejada
plot(x,dy1env2,'k--') % desenho a segunda envoltória em linha preta tracejada
hold off        % Não desejo mais adicionar curvas a este gráfico
grid on         % Ligo as linhas de grade
% Coloco legenda nas curvas de velocidade e envoltórias
legend('Velocidade','Envoltórias')
title('Velocidade')       % Coloco o título neste gráfico
xlabel('Tempo [s]')       % Coloco a legenda no eixo X
ylabel('Amplitude [m/s]')   % Coloco a legenda no eixo Y
```

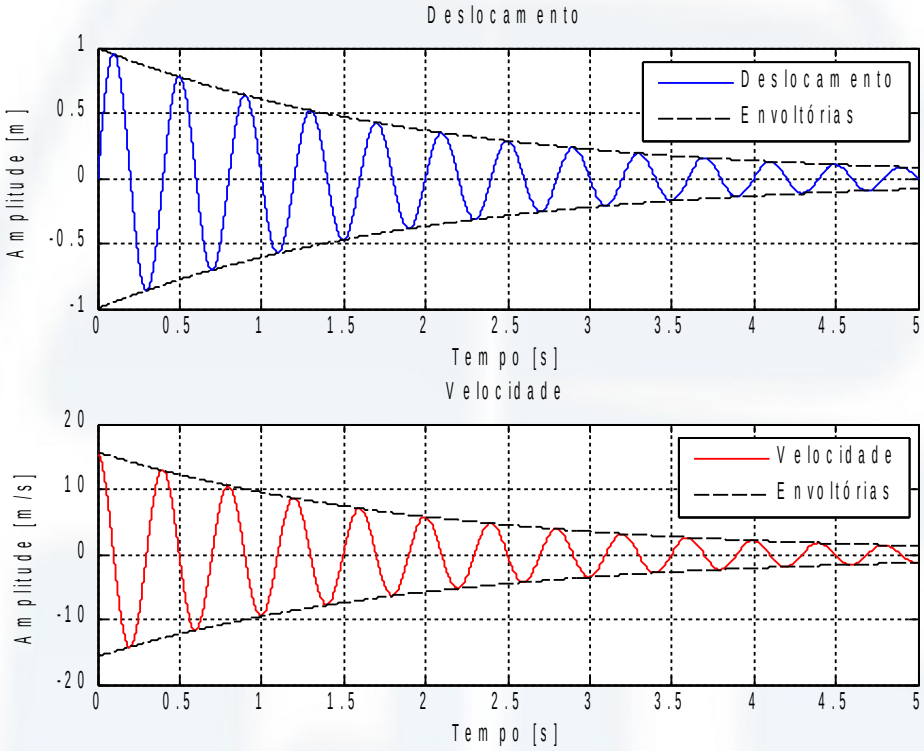


Figura 63:Exemplo Completo

Além destes comandos que estamos abordando aqui, o Matlab tem muitos outros comandos que editam outras propriedades, como propriedades de fonte e estilo das *labels* de eixos e título de gráfico, que podem ser encontradas no *help*.

Lembre-se: No Matlab você desenvolverá rotinas, que serão executadas muitas vezes. Quanto mais automatizado você tornar seu programa, menor será o seu trabalho no momento de exibir os gráficos, pois se você criar a rotina de plotagem corretamente, os gráficos sairão a seu gosto de maneira automática.

Mesmo assim, também há uma maneira de alterar todas as propriedades manualmente. Ao plotar seu gráfico você se deparará com esta janela:

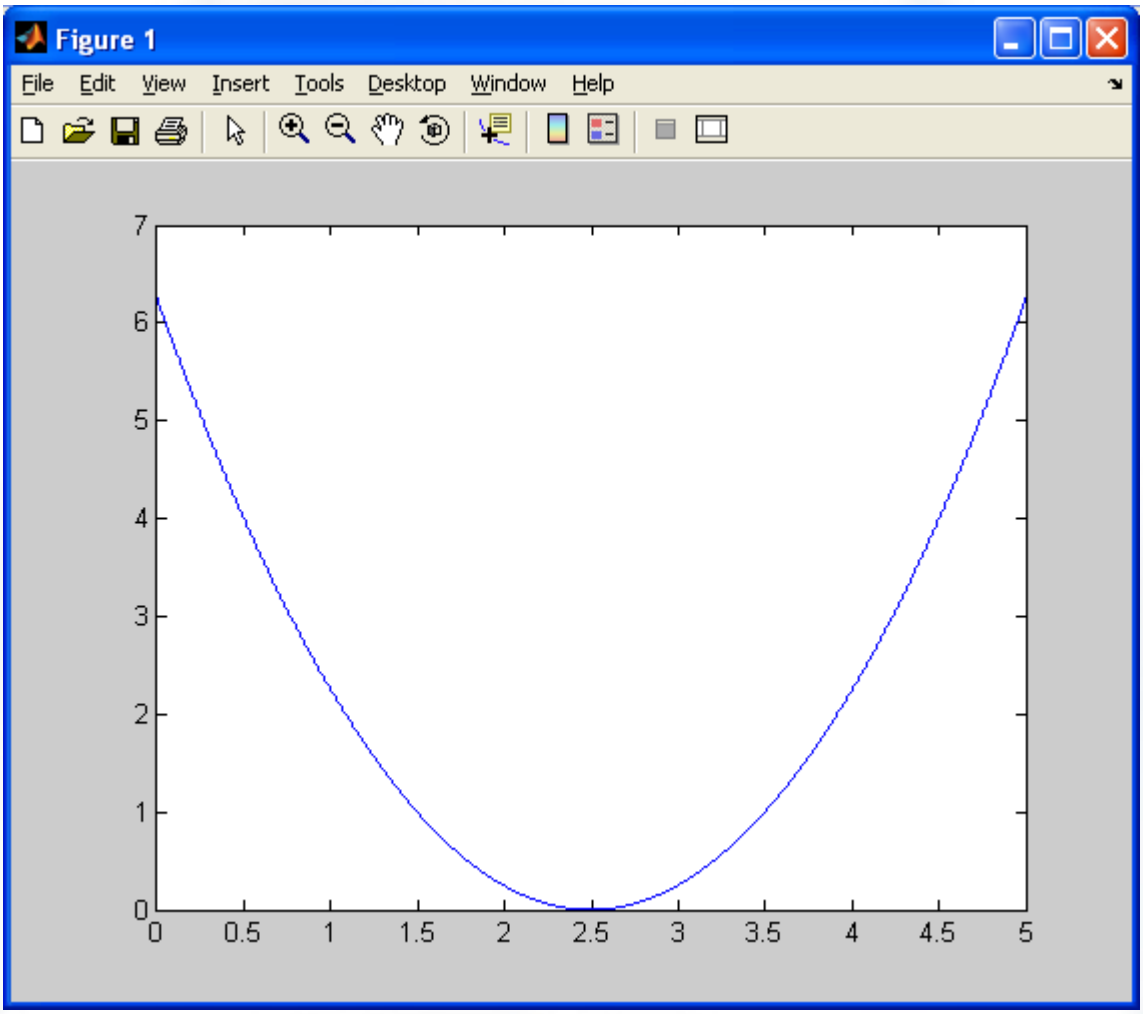


Figura 64:Janela do gráfico

Para poder editar as propriedades do gráfico manualmente, é preciso ter acesso às ferramentas de edição de gráfico, e isto é possível se clicarmos no ícone mais à direita da barra de ferramentas superior da janela, como mostrado na figura abaixo.

Figura 65:Ícone da janela

Ao clicar neste botão, a sua janela de gráfico ficará da seguinte forma:

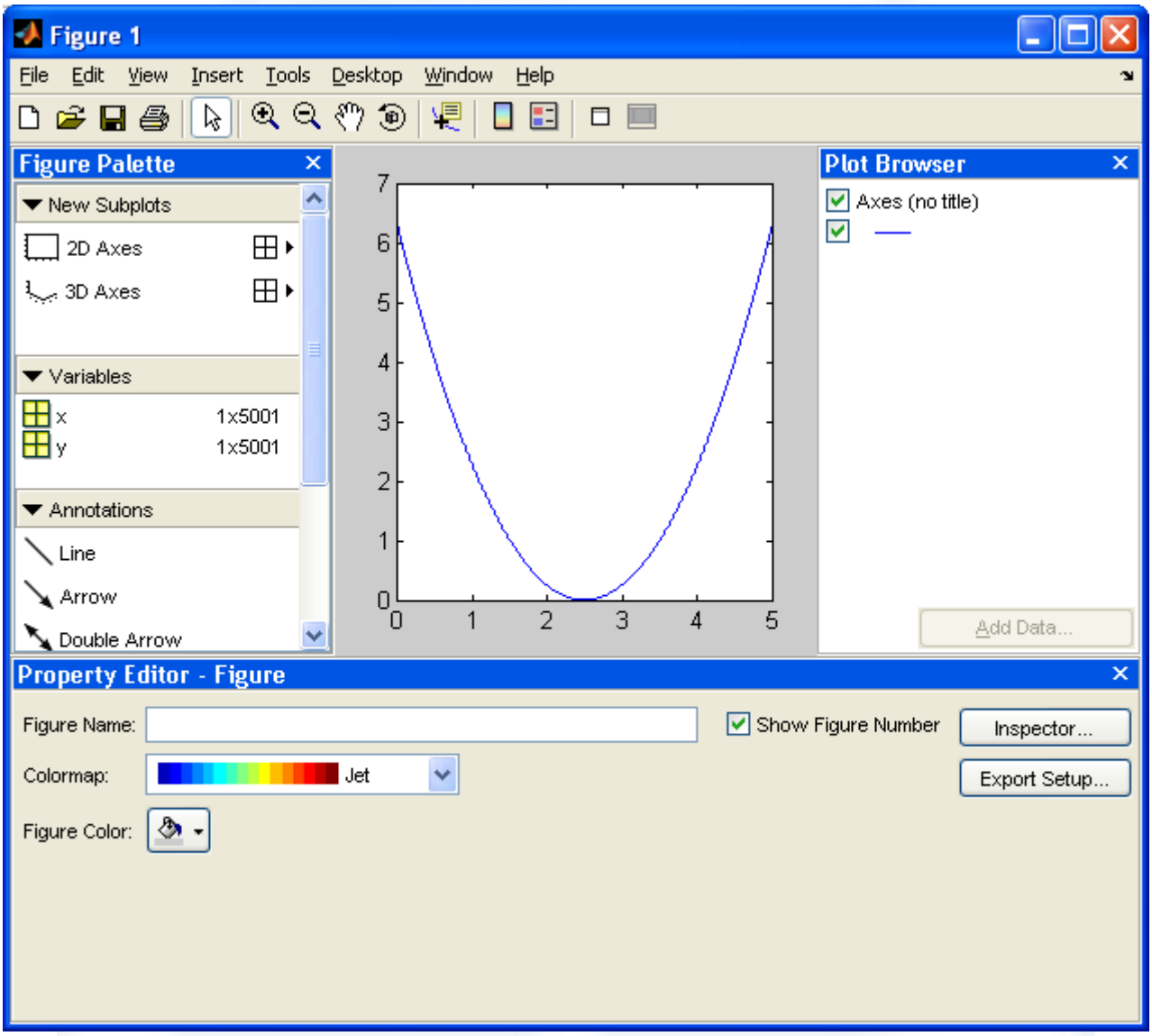


Figura 66:Janela do gráfico após clicar no ícone

A coluna Plot Browser, à direita, é o lugar onde vamos selecionar os elementos que desejamos alterar. A coluna Figure Palette, à esquerda, mostra todos os elementos que podemos adicionar no gráfico. A barra inferior, com título de Property Editor, é dinâmica, e mudará para cada elemento selecionado que desejemos alterar.

Quando nada está selecionado na Plot Browser, o Property Editor permitirá que alteremos características da própria figure, como seu título (com ou sem NumberTitle), a cor de fundo e o colormap, que não será usado por enquanto.

Ao selecionar "axes" no Plot Browser, a Property Editor ficará da seguinte forma:

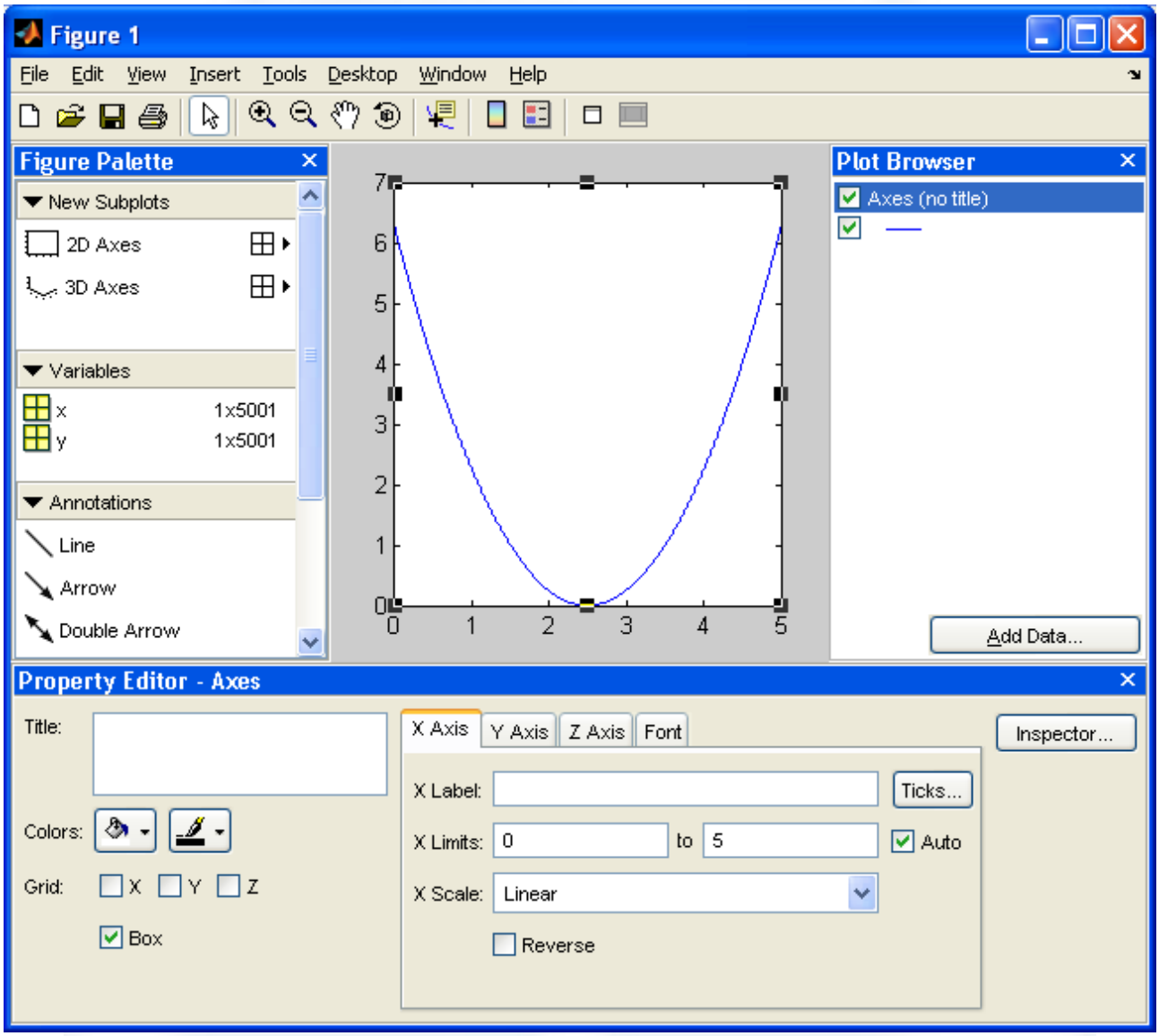


Figura 67:Axis selecionado

De modo que para cada eixo, poderemos mudar sua Label, seus limites, sua escala (linear ou log), seu sentido (reverso ou não). No gráfico, poderemos mudar o título, o grid, a cor dos eixos e do fundo do gráfico (não da figure).
Além disso, na aba Font, características relativas à fonte usada podem ser alteradas.
Se selecionarmos a linha azul, poderemos mudar características sobre a curva desenhada. Neste caso, a janela ficará da seguinte maneira:

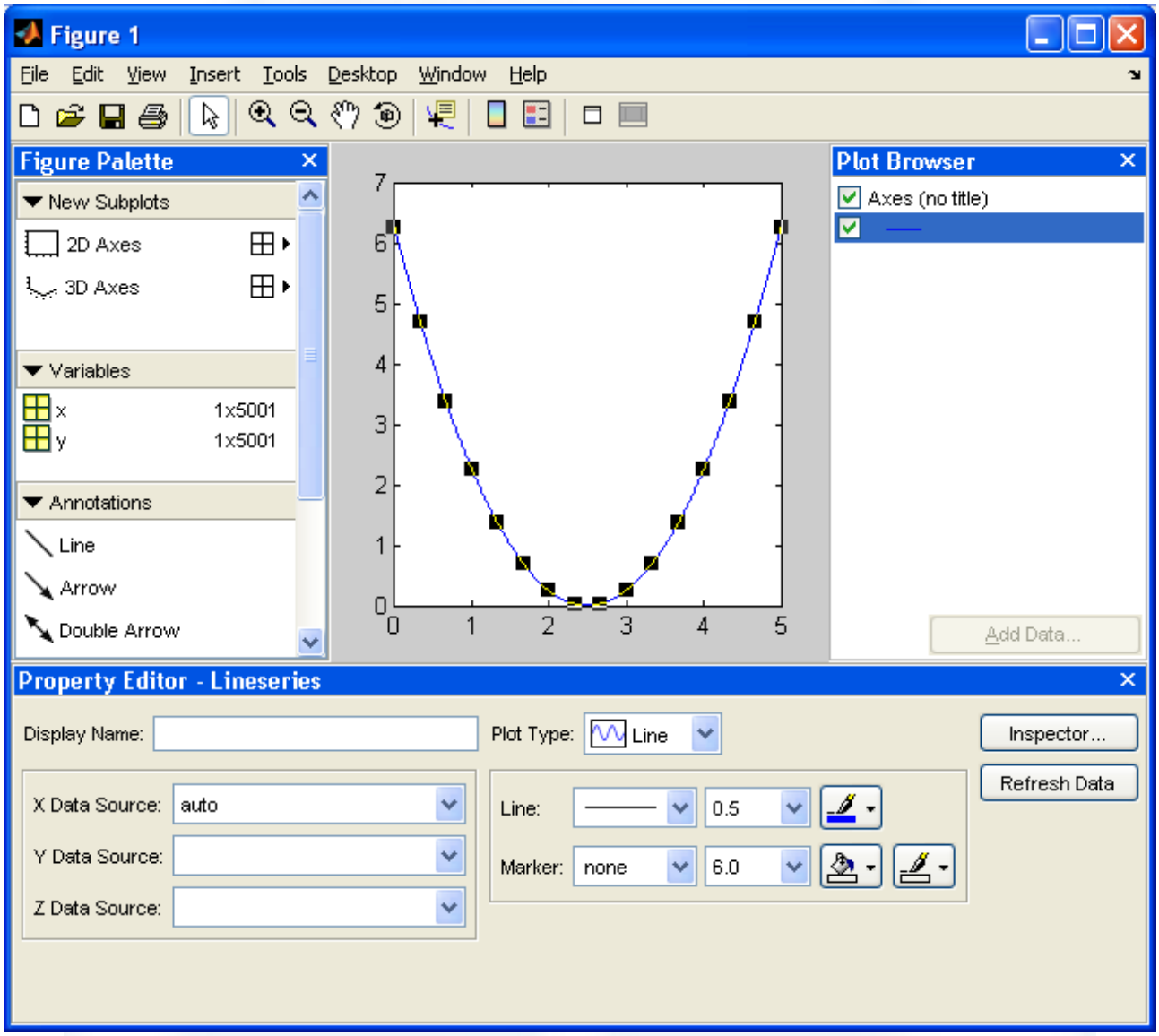


Figura 68:Linha selecionada

Podemos, neste caso, alterar forma, tamanho e cor da linha e dos marcadores, além de poder mudar a fonte dos dados em X, Y e Z e o nome da própria curva.
Além disso, podemos alterar o tipo de plot. Podemos plotar um conjunto de dados nas formas "Stem", "Line", "Bar", "Area" e "Stairs". Isto pode ser feito por meio desta janela, no menu Plot Type, ou por meio de linhas de código. Veremos a seguir cada um destes tipos de gráfico detalhadamente, e por linha de comando.

Tipos de Plot

Nem todos os dados que calculamos no Matlab representam dados contínuos, e por isso, a melhor representação para estes dados pode não ser o plot, onde todos os pontos são interligados por retas. Por isso, o Matlab disponibiliza outras formas de plotar os gráficos.

– Plot

O comando plot desenha os pontos definidos pelos vetores *x* e *y* interligados por retas, como visto anteriormente. No menu citado anteriormente, no Property Editor dos dados do gráfico, ele é referido como "Line".

Exemplo:

```
clear all
close all
clc

x = 0:0.5:5; % Inicializo o vetor x
y = (x-2.5).^2; % Inicializo o vetor y

plot(x,y)
```

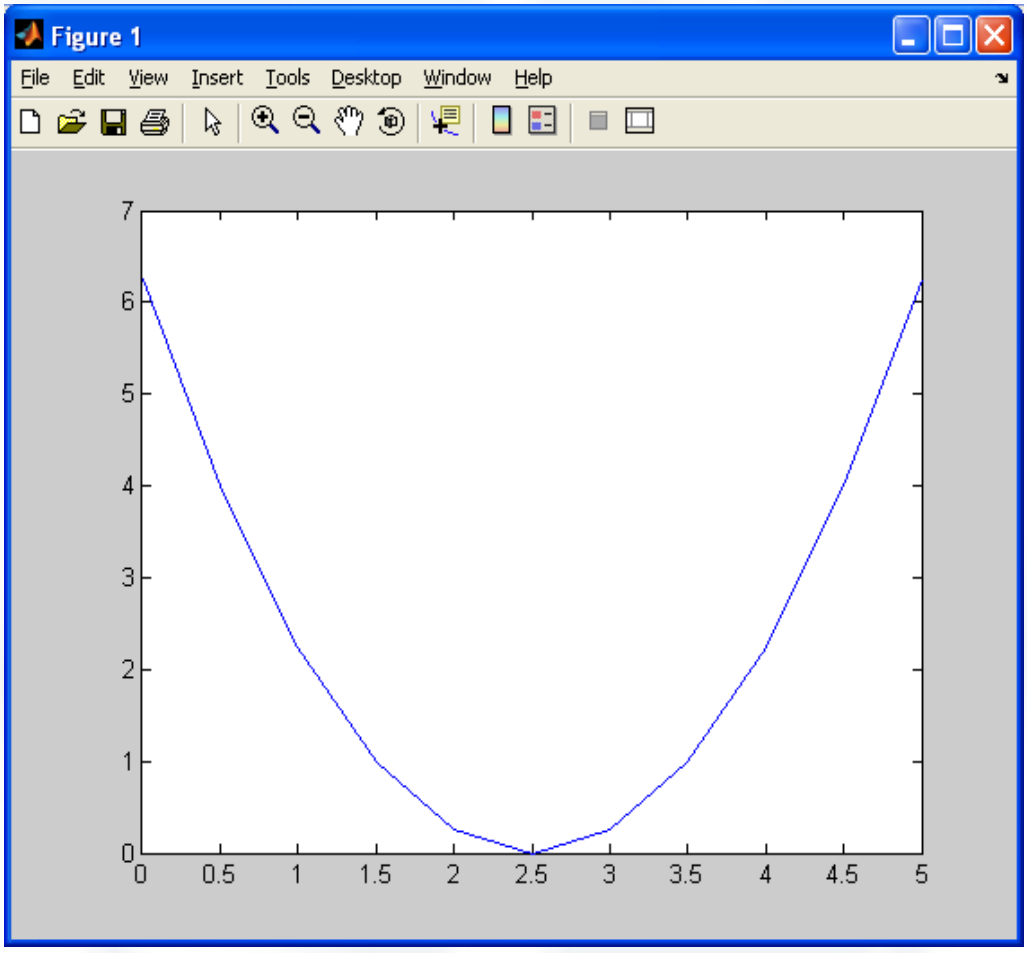


Figura 69:Tipo de plot - plot

– Stem

Para representar dados discretos no tempo, normalmente se usa o Stem. Este comando desenha os dados como círculos, ligados por retas até a linha vertical, que fica no nível zero.

Exemplo:

```
clear all
close all
clc

x = 0:0.5:5; % Inicializo o vetor x
y = (x-2.5).^2-2; % Inicializo o vetor y

stem(x,y)
```

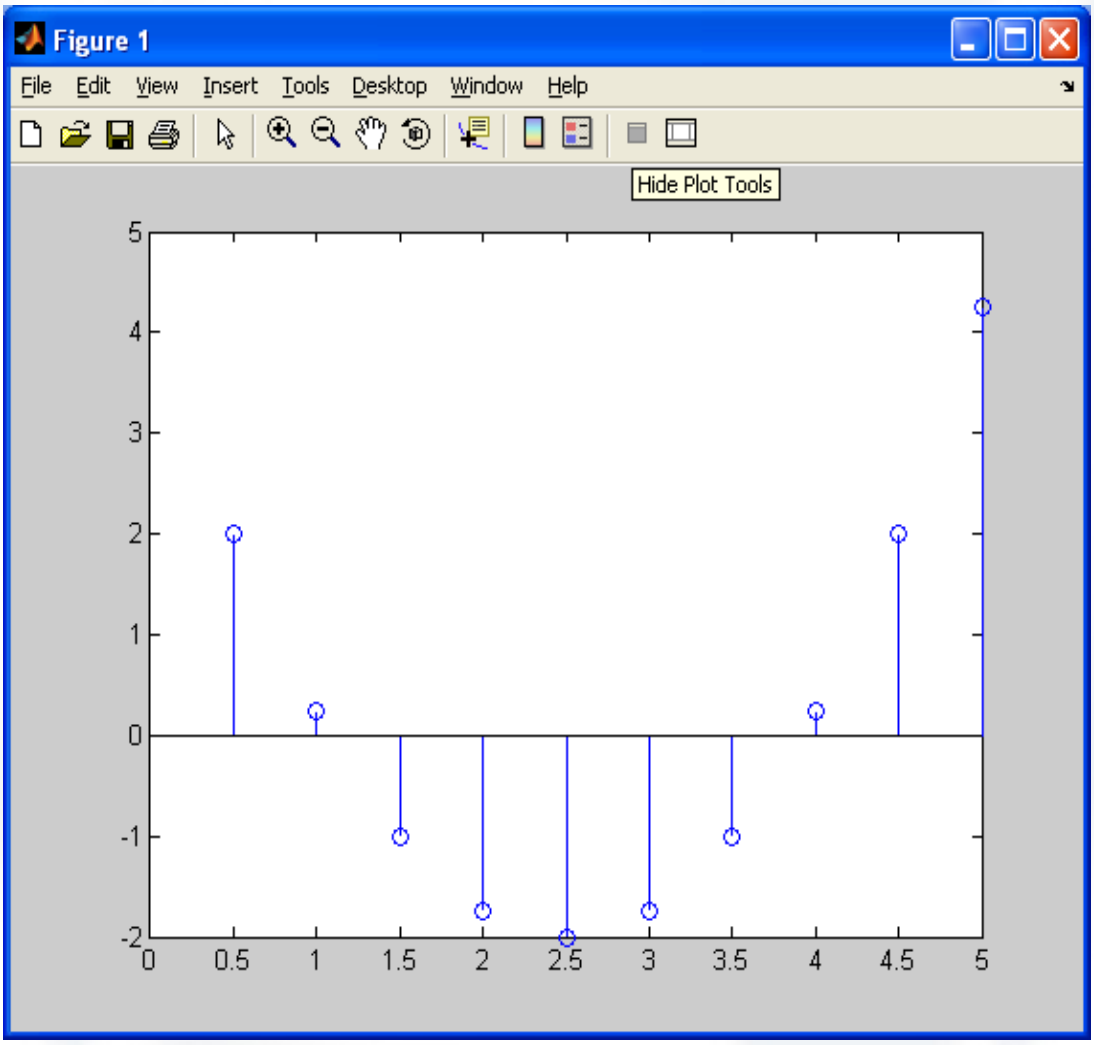


Figura 70:Tipo de plot - stem

– Bar

Outro tipo de gráfico muito utilizado para estatísticas é o gráfico de barras. Este pode ser feito através do comando *bar*.

Exemplo:

```
clear all
close all
clc

x = 0:0.5:5; % Inicializo o vetor x
y = (x-2.5).^2; % Inicializo o vetor y

bar(x,y)
```

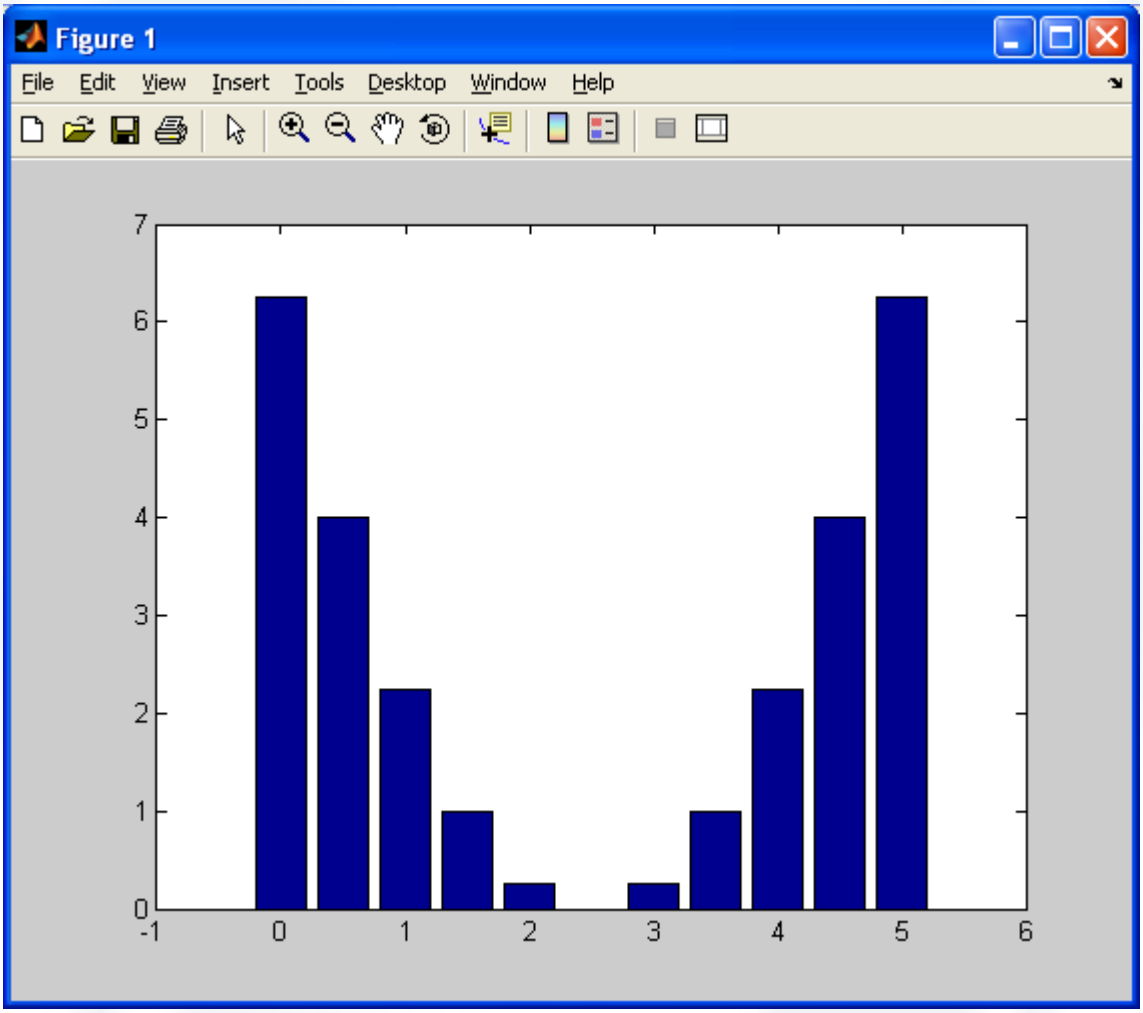


Figura 71: Tipo de plot - bar

– Area

Pode ser necessário também identificar a região que identifica a área sob a curva mostrada. Isto pode ser feito através do comando *area*.

Exemplo:

```
clear all
close all
clc

x = 0:0.5:5; % Inicializo o vetor x
y = (x-2.5).^2; % Inicializo o vetor y

area(x,y)
```

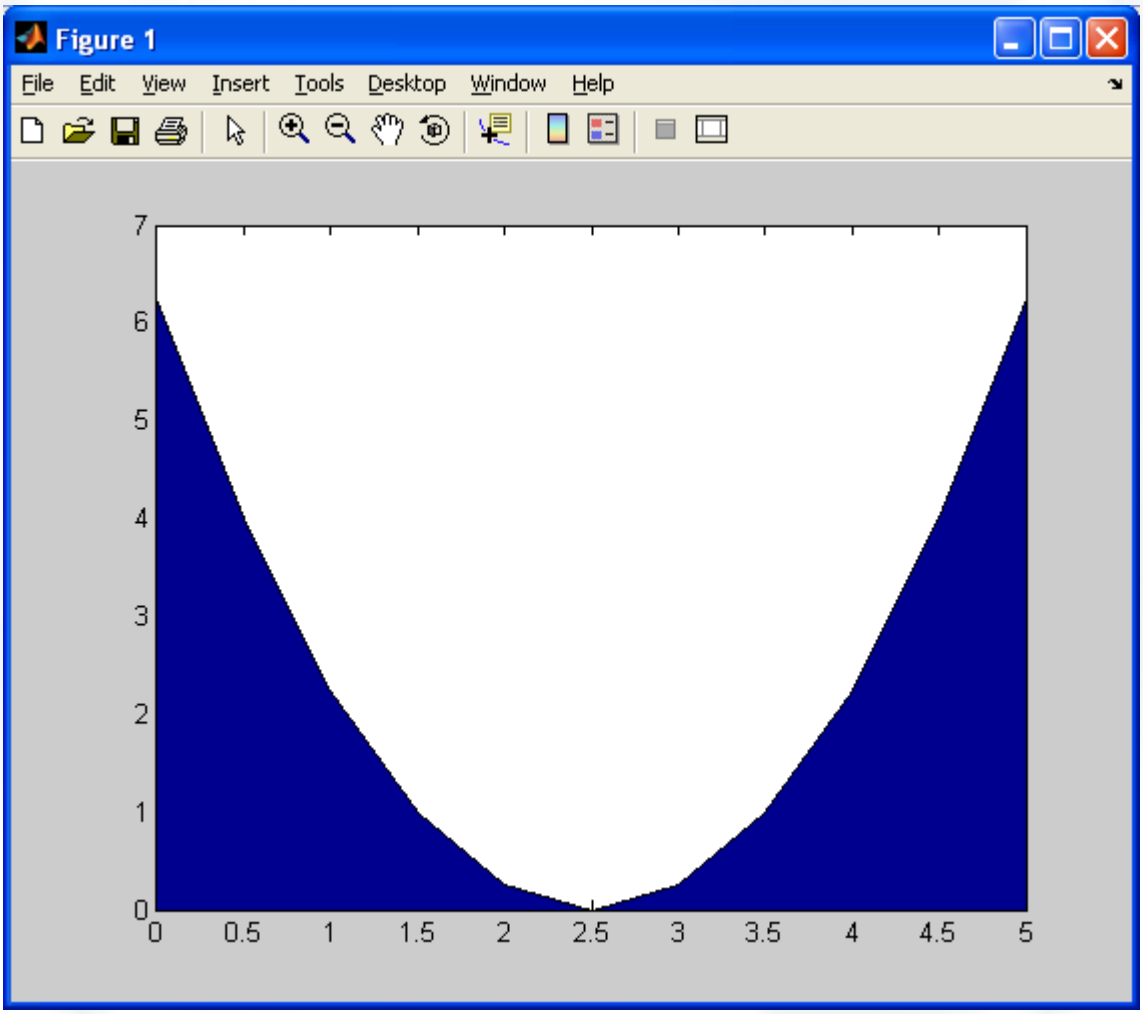


Figura 72: Tipo de plot - area

– Stairs

Na área de controle discreto, é comum desenhar os sinais na forma de escada, mostrando que o valor do sinal não muda até a próxima iteração do sistema. Isto é feito por meio da função *stairs*.

Exemplo:

```
clear all
close all
clc

x = 0:0.5:5; % Inicializo o vetor x
y = (x-2.5).^2; % Inicializo o vetor y

stairs(x,y)
```

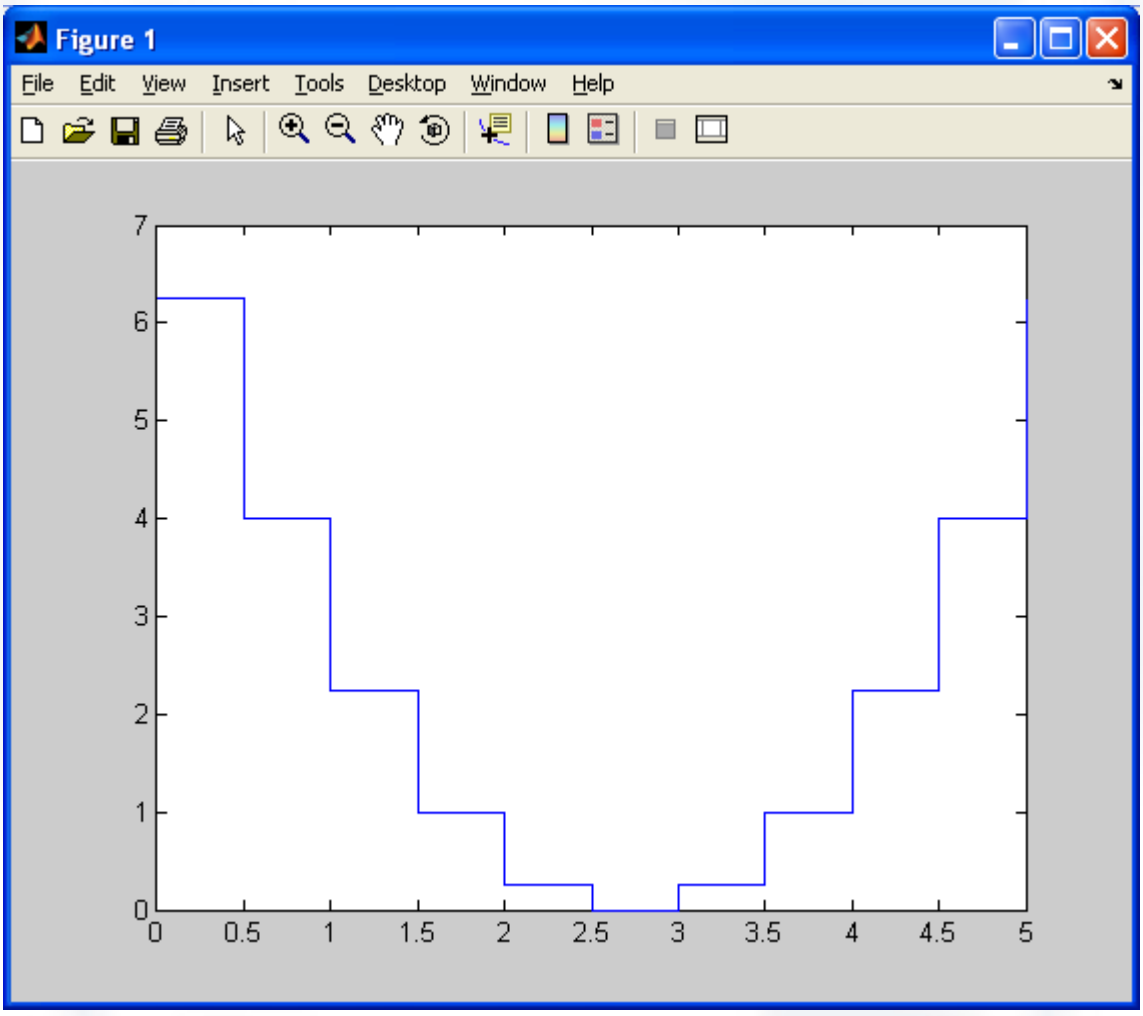


Figura 73: Tipo de plot - stairs

IMPORTANTE: As funções *stem* e *stairs,* pelo fato de plotarem linhas, possuem a mesma sintaxe do comando *plot* para alteração de cores e outros parâmetros da curva a ser desenhada.

Escalas

Finalmente, nos procedimentos experimentais e teóricos, para melhor visualização de gráficos, são utilizadas escalas diferentes da escala linear à qual estamos habituados. Podemos representar estes dados em escala semilog (em x ou y), ou em escala loglog. No Matlab, podemos fazer isto a partir das seguintes funções:

– semilogx

Neste caso, o gráfico será representado de forma que o eixo *x* esteja em escala logarítmica e o eixo *y* em escala linear.

Exemplo:

```
clear all
close all
clc

x = 0:0.001:5; % Inicializo o vetor x
y = x.^2; % Inicializo o vetor y

semilogx(x,y)
```

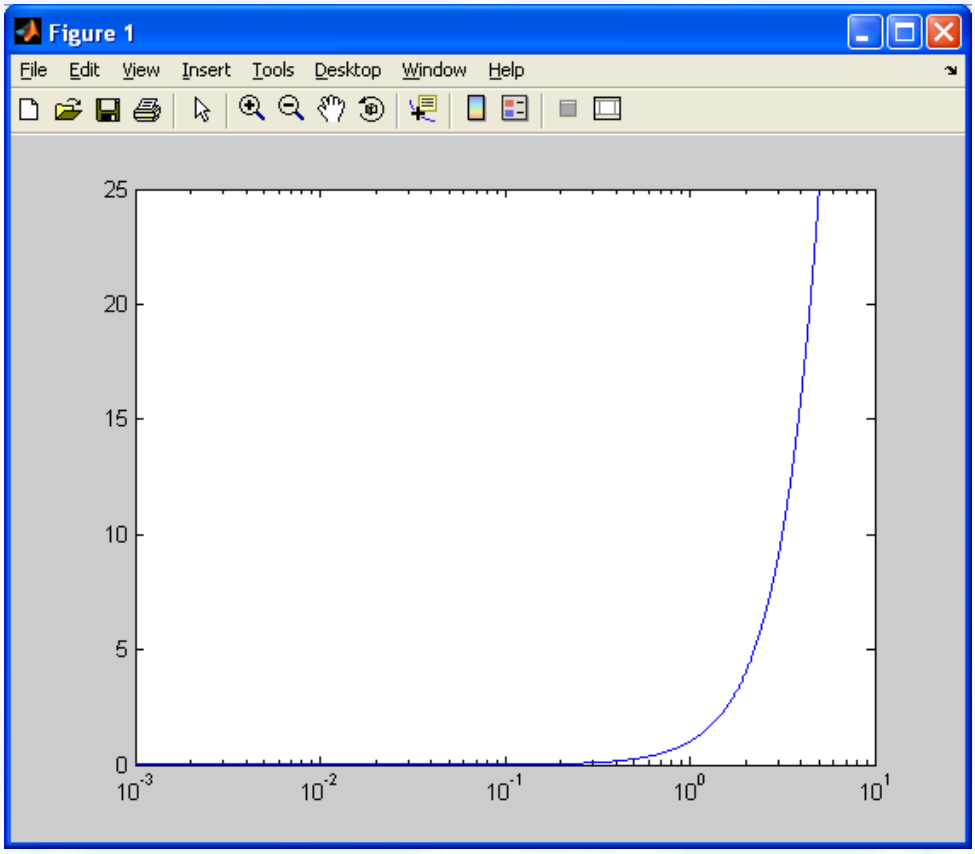


Figura 74: Tipo de plot - semilogx

– semilogy

Neste caso, o gráfico será representado de forma que o eixo *y* esteja em escala logarítmica e o eixo *x* em escala linear.

Exemplo:

```
clear all
close all
clc

x = 0:0.001:5; % Inicializo o vetor x
y = x.^2; % Inicializo o vetor y

semilogy(x,y)
```

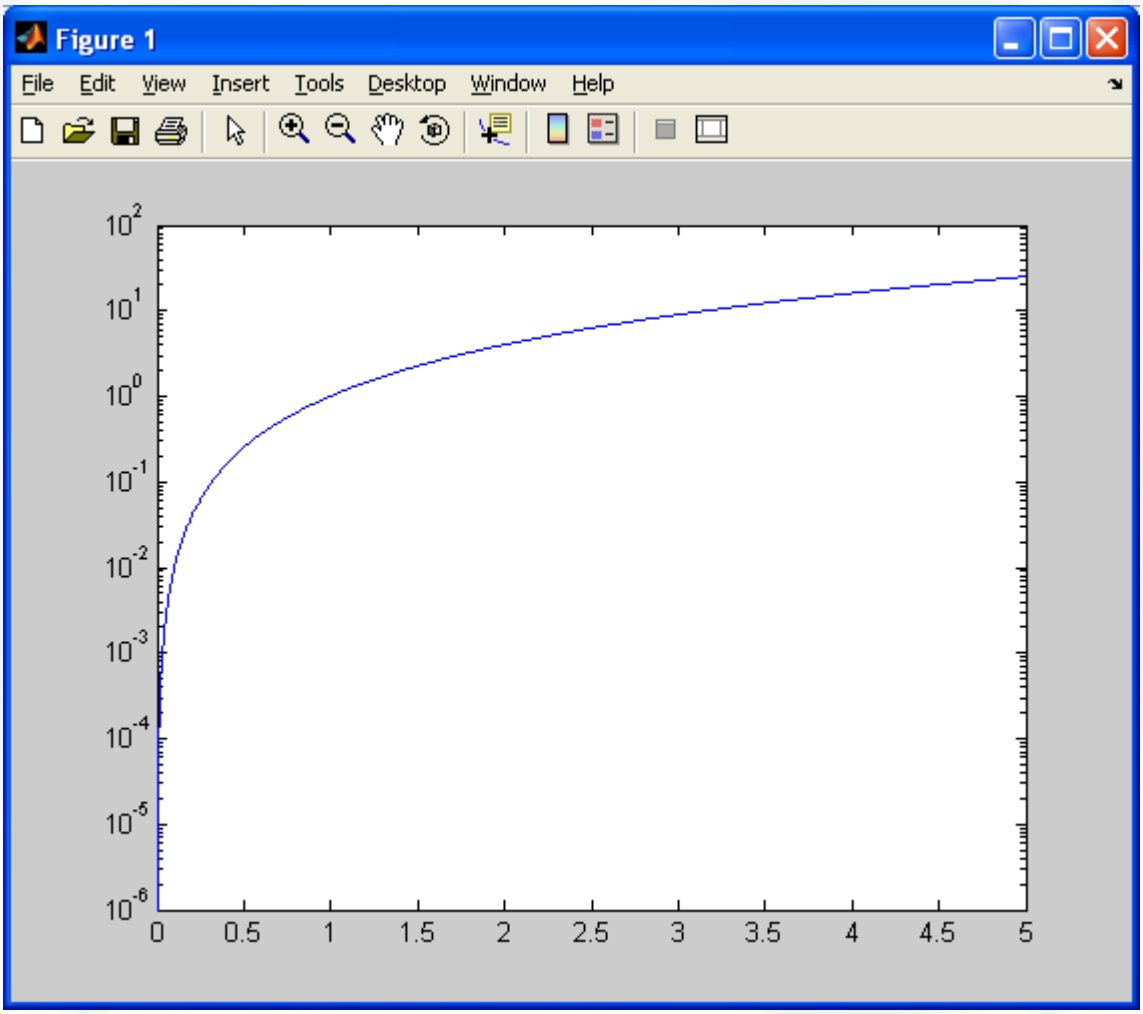


Figura 75: Tipo de plot - semilogy

– loglog

Neste último caso, ambos os eixos do gráfico serão representados em escala logaritmica.

Exemplo:

```
clear all
close all
clc

x = 0:0.001:5; % Inicializo o vetor x
y = x.^2; % Inicializo o vetor y

loglog(x,y)
```

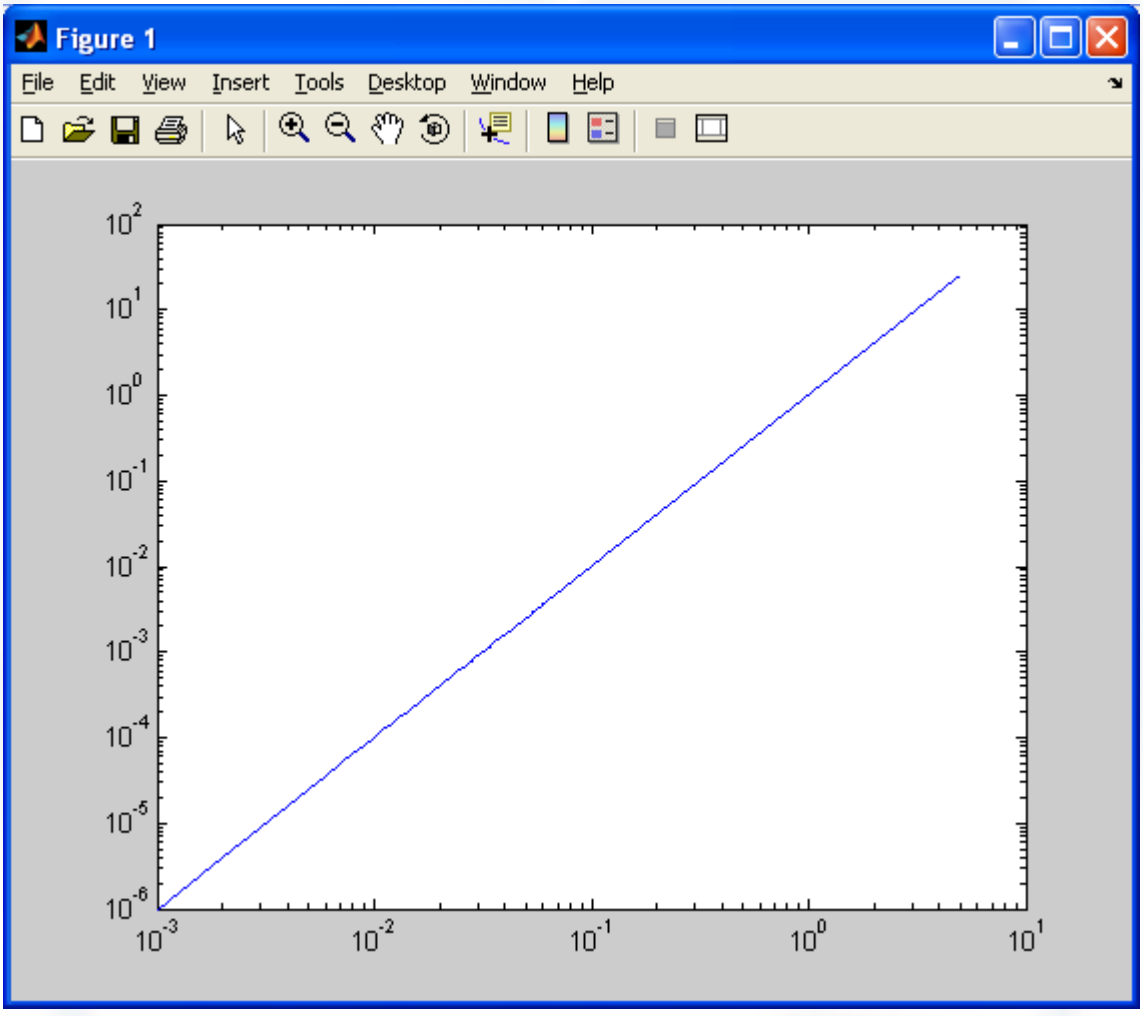


Figura 76: Tipo de plot - loglog

Lembre-se: Apesar de os gráficos serem representados em escalas diferentes da linear, nos vetores criados no Matlab eles permanecem inalterados. Isto quer dizer que, apesar de neste último exemplo o gráfico ter sido representado como uma reta (pois é uma quadrática na escala loglog), na memória permanecem os dados da curva original em escala linear.

Seção 2.2 – Funções Matemáticas

Constantes Matemáticas

– Pi – π

No Matlab a constante numérica $\pi = 3,14159265...$ é representada por *pi*. Portanto, não é aconselhável a utilização de *pi* como variável num programa.

– i,j

O "i" e o "j" neste programa representam, ambos, a unidade complexa. Por isso, não é aconselhado utilizar "i" e "j" como nomes de variáveis, pois isto pode comprometer a compreensão do código e gerar erros de cálculo difíceis de se descobrir.

– Eps

O "eps" é a precisão que se tem entre os números neste ambiente. Este número equivale a 2^-52.

– Realmin, realmax

As contantes *realmin* e *realmax* representam o mínimo e o máximo valores que um número representado em ponto flutuante pode representar no Matlab. A constante *realmin* vale 2^{-1022} e a *realmax* vale $(2-eps).2^{1023}$.

– Inf, NaN

O "infinity" (Inf) e o "Not a Number" são entidades que representam o resultado de operações que poderiam gerar erros numéricos. Quando tentamos dividir algum valor não-nulo por zero, o resultado desta operação é "Inf". Quando temtamos dividir zero por zero "0/0" ou infinito por infinito "Inf/Inf", tentamos fazer uma operação que não é bem definida matematicamente. Portanto, o resultado desta operação é "NaN".

Exemplo:

Command Window:

```
>> pi

ans =

    3.1416
```

Nota: a precisão de *pi* é a máxima permitida pelo software, porém são apresentados resultados só até 4 casas decimais no *Command Window*.

Funções matemáticas

– Funções Trigonométricas

As funções trigonométricas do Matlab utilizam **ângulos em radianos**. A seguir, temos uma tabela com as principais funções do sistema:

Função Trigonométrica	Função Matlab
Seno	sin(x)
Cosseno	cos(x)
Tangente	tan(x)
Cotangente	cot(x)
Secante	sec(x)
Cossecante	csc(x)

– Funções Trigonométricas inversas

As funções trigonométricas inversas do Matlab fornecem resultados com **ângulos em radianos**.

Função Trigonométrica	Função Matlab	Limite para x para resultado real
Arco Seno	asin(x)	[-1,1]
Arco Cosseno	acos(x)	[-1,1]
Arco Tangente	atan(x)	
Arco Cotangente	acot(x)	
Arco Secante	asec(x)	
Arco Cossecante	acsc(x)	

Nota: fora dos limites de x, o resultado obtido é complexo

– Funções Trigonométricas Hiperbólicas

As funções trigonométricas hiperbólicas do Matlab utilizam **ângulos em radianos**. A seguir, temos uma tabela com as principais funções encontradas:

Função Trigonométrica Hiperbólica	Função Matlab
Seno Hiperbólico	sinh(x)
Cosseno Hiperbólico	cosh(x)
Tangente Hiperbólica	tanh(x)
Cotangente Hiperbólica	coth(x)
Secante Hiperbólica	sech(x)
Cossecante Hiperbólica	csch(x)

– Funções Trigonométricas Hiperbólicas Inversas

As funções trigonométricas inversas do Matlab fornecem resultados com **ângulos em radianos**.

Função Trigonométrica Hiperbólica Inversas	Função Matlab
Arco Seno Hiperbólico	asinh(x)
Arco Cosseno Hiperbólico	acosh(x)
Arco Tangente Hiperbólico	atanh(x)
Arco Cotangente Hiperbólico	acoth(x)
Arco Secante Hiperbólico	asech(x)
Arco Cossecante Hiperbólico	acsch(x)

Nota: todas as funções trigonométricas podem trabalhar com valores x complexos

As funções trigonométricas podem receber x (seu argumento) na forma de variável simples, vetor ou matriz. Nessa sintaxe, a função calcula a função trigonométrica relativa a cada um dos seus elementos:

Exemplo:

Função trigonométrica de variável.

```
clear all;
close all;
clc;

x = pi/3; % 60 graus
y = sin(x)

c = pi/4+i*pi/4; %variavel complexa
z1 = sin(c)

v = 2; %valor fora da regiao com resultado real
z2 = asin(v)
```

Command Window:

```
y =

    0.8660


z1 =

   0.9366 + 0.6142i


z2 =

   1.5708 - 1.3170i
```

Exemplo:

Função trigonométrica de valores de vetor.

```
clear all
close all
clc

%vetor de 0° ate 360° espacado de 0.1 graus
x = 0:0.1:2*pi
y = cos(x)

%vetor de 30 pontos entre de 00° ate 360°
vet = linspace(0,2*pi,30)
vet_angle = sin(vet)

plot(x,y,'r+',vet,vet_angle,'b.')
legend('cosseno','seno')
```

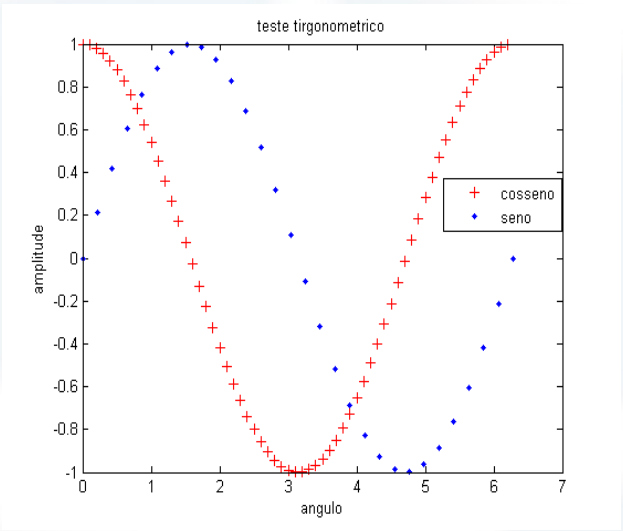


Figura 77: Seno e Cosseno

Exemplo:

Função trigonométrica de valores de matriz

```
clear all
close all
clc

%matriz 2x2 com angulos em radianos
A = [0 pi/6;pi/3 pi/2]
B = sin(A) %calculo da funcao para todos os elementos da matriz
```

Command Window:

```
A =

         0    0.5236
    1.0472    1.5708


B =

         0    0.5000
    0.8660    1.0000
```

– Funções trigonométricas em graus

No Matlab, existem versões **em graus** para as funções trigonométricas.

Função Trigonométrica	Função Matlab
Seno	sind(x)
Cosseno	cosd(x)
Tangente	tand(x)
Cotangente	cotd(x)
Secante	secd(x)
Cossecante	cscd(x)

Também existem as versões **em graus** para as funções trigonométricas inversas. Neste caso, o resultado retornado será um valor em graus, e não mais em radianos.

Função Trigonométrica	Função Matlab
Arco Seno	asind(x)
Arco Cosseno	acosd(x)
Arco Tangente	atand(x)
Arco Cotangente	acotd(x)
Arco Secante	asecd(x)
Arco Cossecante	acscd(x)

– Função exponencial

A função *exp(variavel)* retorna a exponencial de cada elemento da variavel. No caso da variavel ser complexa (*variavel = x+i*y*), a função calcula:

$$\exp(x+iy)=e^{x}(\cos(y)+isin(y))$$

Nota: A base da exponencial calculada é o número de Euler, que equivale a 2,718281828459045...

Exemplo:

Exponencial de variável.

```
clear all
close all
clc

a = 0;
b = exp(a)

c = 100;
d = exp(c)

f = 2+2*i;
g = exp(f)
```

Command Window:

```
b =

     1


d =

  2.6881e+043


g =

  -3.0749 + 6.7188i
```

Exemplo:

Exponencial de valores de vetor.

```
clear all
close all
clc

a = 0:0.1:1;
b = exp(a)
```

Command Window:

```
b =

    1.0000    1.1052    1.2214    1.3499    1.4918    1.6487    1.8221    2.0138
2.2255    2.4596    2.7183
```

Exemplo:

Exponencial de valores de matriz.

```
clear all
close all
clc

A = [1 2;3 4];
B = exp(A);

disp('Exponencial de cada elemento da matriz')
disp(B)

disp('Verificacao')
disp(exp(A(1,1)))
disp(exp(A(1,2)))
disp(exp(A(2,1)))
disp(exp(A(2,2)))
```

Command Window:

```
Exponencial de cada elemento da matriz
    2.7183    7.3891
   20.0855   54.5982

Verificacao
    2.7183

    7.3891

   20.0855

   54.5982
```

IMPORTANTE: a função *exp(A)* realiza a operação de exponencial em cada elemento da matriz:

$$\exp(A)=\begin{Bmatrix} e^{a_{11}} & e^{a_{12}} & \dots & e^{a_{1n}} \\ e^{a_{21}} & e^{a_{22}} & \dots & e^{a_{2n}} \\ \dots & \dots & \dots & \dots \\ e^{a_{n1}} & e^{a_{n2}} & \dots & e^{a_{nn}} \end{Bmatrix}$$

Ela possui resultado diferente da função *expm(A),* que realiza a operação de exponencial matricial:

$$expm(A)=e^{A}$$

– Logaritmo na base *e* (logaritmo neperiano, ou natural)

Este logaritmo é representado pela função *log(x)*.

IMPORTANTE: No Matlab a função *log(x)* realiza o cálculo do logaritmo na base *e*, ou seja, *log(x) = ln(x)*. O logaritmo na base 10 é calculado pela função *log10(x)*.

Exemplo:

```
clear all
close all
clc

v = [0 10 100 1000 10000]
d = log(v)
```

Command Window:

```
v =

           0          10         100        1000       10000

Warning: Log of zero.

d =

      -Inf    2.3026    4.6052    6.9078    9.2103
```

Nota: o *Warning* apresentado pelo programa ocorre pois $\lim_{x\to 0} \ln(x) = -\infty$

– Logaritmo na base 10

A função *log10(x)* realiza o cálculo do logaritmo na base 10 neste ambiente.

IMPORTANTE: No Matlab a função *log(x)* realiza o cálculo do logaritmo na base *e*, ou seja, *log(x) = ln(x)*. O logaritmo na base 10 é calculado pela função *log10(x)*.

Exemplo:

```
clear all
close all
clc

v = [0 10 100 1000 10000]
d = log10(v)
```

Command Window:

```
v =

           0          10         100        1000       10000
```

```
Warning: Log of zero.

d =

  -Inf     1     2     3     4
```

Nota: o *Warning* apresentado pelo programa ocorre pois $\lim_{x\to 0}\log_{10}(x)=-\infty$

Polinômios

Em linguagem Matlab, um polinômio é representado na forma de um vetor. Seja um polinômio do tipo:

$$p(x)=a_1 x^n+a_2 x^{n-1}+...+a_n x+a_{n+1}$$

Ele é representado na forma de um vetor com os seus coeficientes:

p = [a1 a2 … an an1];

Exemplo:

Polinômio	Em Matlab
$p(x)=2x^3+3x^2+4x+5$	p = [2 3 4 5]
$p(x)=x^4+x^2+5$	p = [1 0 1 0 5]
$p(x)=3x^4+3x^3+x^2+5$	p = [3 3 1 0 5]
$p(x)=3x^3+x^2+5$	p = [3 1 0 5]

– Cálculo de raízes de polinômios

O comando *roots(polinomio)*calcula as raízes de um polinômio dado na forma de um vetor coluna.

Exemplo:

```
clear all
close all
clc

%polinomios
p1 = [1 2 1];      %p1 = x^2 + 2*x + 1
p2 = [1 5 4];      %p2 = x^2 + 5*x + 4
p3 = [2 1 3 1];    %p3 = 2*x^3 + x^2 + 3*x + 1
p4 = [1 0 0 2 1]; %p4 = x^4 + 2*x + 1
%raizes
r1 = roots(p1)
r2 = roots(p2)
r3 = roots(p3)
r4 = roots(p4)

disp(['raizes do polinomio p1: ' num2str(r1')])
disp(['raizes do polinomio p2: ' num2str(r2')])
disp(['raizes do polinomio p3: ' num2str(r3')])
```

```
disp(['raizes do polinomio p4: ' num2str(r4')])
```

Command Window:

```
r1 =

    -1
    -1


r2 =

    -4
    -1


r3 =

  -0.0772 + 1.2003i
  -0.0772 - 1.2003i
  -0.3456


r4 =

   0.7718 + 1.1151i
   0.7718 - 1.1151i
  -1.0000
  -0.5437

raizes do polinomio p1: -1 -1
raizes do polinomio p2: -4 -1
raizes do polinomio p3: -0.077186-1.2003i        -0.077186+1.2003i         -0.34563+0i
raizes do polinomio p4: 0.77184-1.1151i          0.77184+1.1151i               -1+0i
-0.54369+0i
```

– Cálculo do polinômio para valores específicos de *x*

Uma vez criado um vetor p, é possível calculá-lo para determinados valores de *x* utilizando o comando *polyval(p,x)*, sendo que o valor pode ser avaliado para uma variável ou para um vetor de valores.

Exemplo:

Calculando polinômio para uma variável.

```
clear all
close all
clc

p = [1 2 3 4]; %p = x^3 + 2*x^2 + 3*x + 4
x = 2;
valor = polyval(p,x);%valor = 1*2^3 + 2*2^2 + 3*2^1 + 4*2^0
disp(['polinomio calculado para x = ' int2str(x)])
disp(valor)
```

Command Window:

```
polinomio calculado para x = 2
    26
```

Exemplo:

Calculando polinômio para vetor.

```
clear all
close all
clc

p = [1 2 3 4]; %p = x^3+2*x^2+3*x+4
x = [2 3 4 5]; %valores
valor = polyval(p,x);
disp(['polinomio calculado para x = ' int2str(x(1)) ' vale ' int2str(valor(1))])
disp(['polinomio calculado para x = ' int2str(x(2)) ' vale ' int2str(valor(2))])
disp(['polinomio calculado para x = ' int2str(x(3)) ' vale ' int2str(valor(3))])
disp(['polinomio calculado para x = ' int2str(x(4)) ' vale ' int2str(valor(4))])
```

Command Window:

```
polinomio calculado para x = 2 vale 26
polinomio calculado para x = 3 vale 58
polinomio calculado para x = 4 vale 112
polinomio calculado para x = 5 vale 194
```

– Interpolação polinomial de pontos

Supondo a coleta de dados de um sinal, sendo *dataX* vetor de coordenadas dos pontos no eixo x e *dataY* as coordenadas dos pontos no eixo y,com a função *p = polyfit(dataX,dataY,n)* é possível gerar um polinômio *p* de ordem *n* que interpola os pontos. O polinômio é representado no formato Matlab (vetor).

Exemplo:

Interpolar o conjunto de dados com a função *polyfit* para polinômio de ordem 4 e depois calcular o valor do polinômio para *x = 0.345*, não presente no vetor *dataX*.

```
clear all
close all
clc

%dados
dataX = 0:0.1:1           % em x
dataY = sin(2*pi*1*dataX) % em y

p = polyfit(dataX,dataY,9) %polinomio de ordem 9

%com o polinomio pode-se calcular o valor da funcao para intervalo
%fora de dataX
x = 0.345;
y = polyval(p,x);

disp(['Valor interpolado para x = ' num2str(x) ' vale ' num2str(y)])
```

Command Window:

```
dataX =

         0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000    0.7000    0.8000
0.9000    1.0000
```

```
dataY =

         0    0.5878    0.9511    0.9511    0.5878    0.0000   -0.5878   -0.9511   -0.9511
-0.5878   -0.0000


p =

  -34.4790  155.1555 -234.9748   98.3532   42.4881    9.9261  -42.8775    0.1297    6.2787
0.0000

Valor interpolado para x = 0.345 vale 0.82708
```

Seção 2.3 – Gráficos em 3D

Plot3

Com o comando plot3(X,Y,Z) é possível gerar gráficos tridimensionais a partir das coordenadas X,Y e Z de um conjunto de pontos.

Exemplo:

```
clear all
close all
clc

%vetor de tempo - 0 a 10 s
t = 0:0.001:10;
%dados em X
x = exp(-0.1*t).*cos(2*pi*4*t);%multiplicacao ponto-a-ponto
%dados em Y
y = exp(-0.1*t).*sin(2*pi*4*t);%multiplicacao ponto-a-ponto
%dados em Z
z = t;

figure                  %nova janela para grafico
plot3(x,y,z)            %ploto graficos na janela em 3D
xlabel('X')             %nome eixo X
ylabel('Y')             %nome eixo Y
zlabel('Z')             %nome eixo Z
title('Teste plot3D') %titulo do grafico
grid on                 %habilito linhas de grid
```

Nota: o comando *zlabel('texto desejado')* funciona analogamente as funções *xlabel* e *ylabel*, acrescentando título ao eixo z.

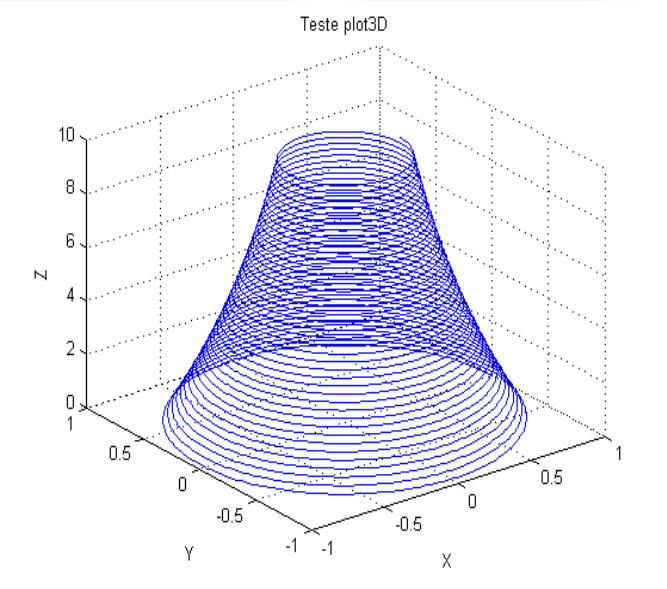


Figura 78: Gráfico obtido do exemplo

Outro exemplo:

Soma de algumas componentes da Série de Fourier de uma onda quadrada.

$$F(x)=\frac{4}{n}\sum_{n=1,3,5,\ldots}^{\infty}\frac{1}{n}\sin\left(\frac{n\pi x}{L}\right)$$

```
clear all
close all
clc

L = 1;            %periodo serie de Fourier
n = [1 3 5 7 9]; %componentes serie de Fourier
x = 0:0.001:2*pi;%angulo

%1ª componente
f1 = (1/n(1))*sin((n(1)*pi*x)/L);
n1 = n(1)*ones(size(f1));
%2ª componente
f2 = (1/n(2))*sin((n(2)*pi*x)/L);
n2 = n(2)*ones(size(f2));
%3ª componente
f3 = (1/n(3))*sin((n(3)*pi*x)/L);
n3 = n(3)*ones(size(f3));
%4ª componente
f4 = (1/n(4))*sin((n(4)*pi*x)/L);
n4 = n(4)*ones(size(f4));
%5ª componente
f5 = (1/n(5))*sin((n(5)*pi*x)/L);
n5 = n(5)*ones(size(f5));
%soma das componentes
f = (4/pi)*(f1+f2+f3+f4+f5);

figure
plot(x,f1+f2+f3+f4+f5)
title('Componentes Serie Fourier : f1+ f2 + f3 + f4 + f5')
xlabel('x (rad)')
ylabel('amplitude')

figure
plot3(x,[n1;n2;n3;n4;n5],[f1;f2;f3;f4;f5])
title('Componentes Serie Fourier')
xlabel('x (rad)')
ylabel('n')
zlabel('amplitude')
grid on                      %habilito linhas de grid
set(gca,'drawmode','fast')   %habilito modo de desenhos rapido
```

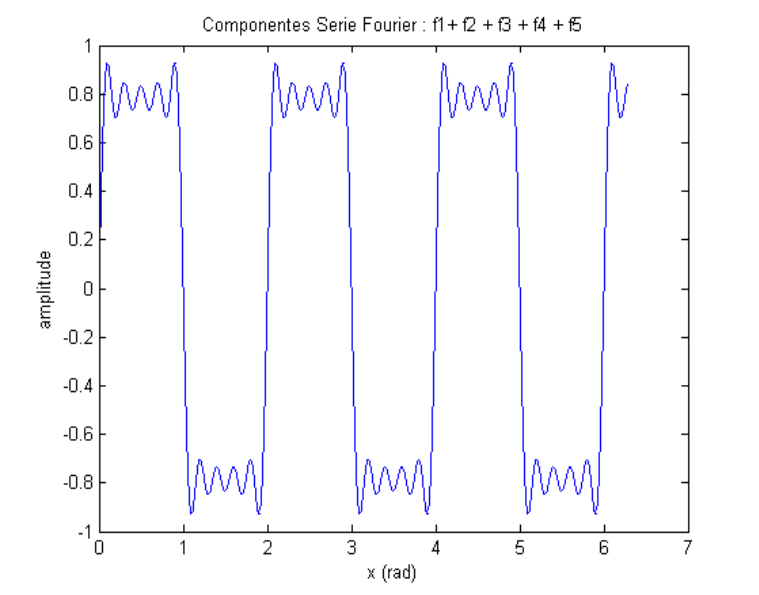


Figura 79: Soma de algumas componentes da Série de Fourier

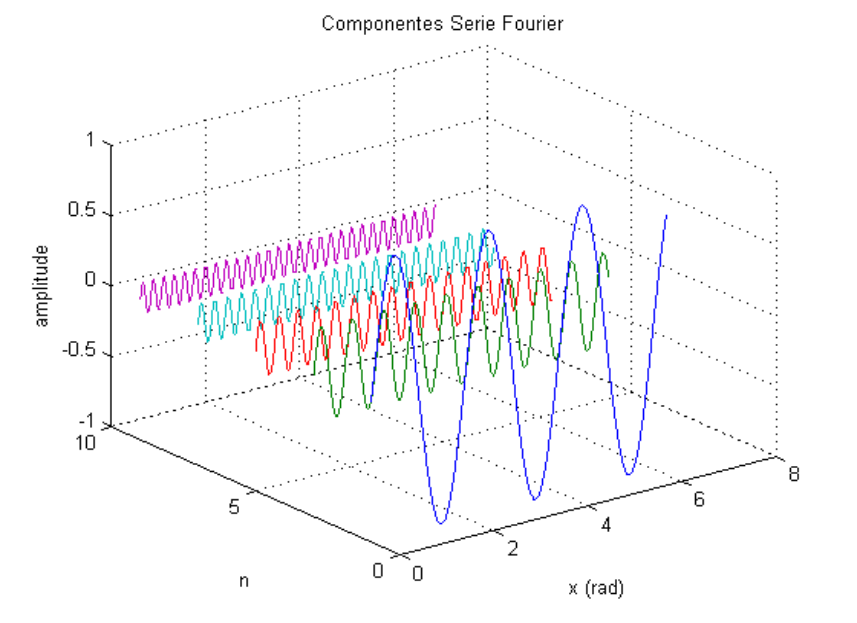


Figura 80: Algumas componentes da Série de Fourier

Acelerando o processo de gerar gráficos 3D

É possível se aumentar a velocidade de plotagem de gráficos utilizando o seginte comando ao final do comandos de plot em 3D:

```
set(gca,'drawmode','fast') %habilito modo de desenhos rápido
```

Exemplo:

A partir do exemplo anterior.

```
clear all
close all
clc

%vetor de tempo - 0 a 10 s
t = linspace(0,10,1000000);%AUMENTO QUANTIDADE DE PONTOS - 10000000
%dados em X
x = exp(-0.1*t).*cos(2*pi*4*t);%multiplicacao ponto-a-ponto
%dados em Y
y = exp(-0.1*t).*sin(2*pi*4*t);%multiplicacao ponto-a-ponto
%dados em Z
z = t;

figure                      %nova janela para grafico
plot3(x,y,z)                %ploto graficos na janela em 3D
xlabel('X')                 %nome eixo X
ylabel('Y')                 %nome eixo Y
zlabel('Z')                 %nome eixo Z
title('Teste plot3D')       %titulo do grafico
grid on                     %habilito linhas de grid
set(gca,'drawmode','fast') %habilito modo de desenhos rápido
```

Nota: *gca* refere-se a *get current axis handle*. Recomenda-se a procura no Help (F1) do software para maiores detalhes.

Subplot

O comando *subplot* funciona também no caso 3D. Com a mesma sintaxe do caso 2D é possível mesclar gráficos de diferentes tipos em cada campo.

Exemplo:

```
clear all
close all
clc

%vetor de tempo - 0 a 10 s
t = linspace(0,10,10000);%QUANTIDADE DE PONTOS - 10000
%dados em X
x = exp(-0.1*t).*cos(2*pi*4*t);%multiplicacao ponto-a-ponto
%dados em Y
y = exp(-0.1*t).*sin(2*pi*4*t);%multiplicacao ponto-a-ponto
%dados em Z
z = t;


figure                  %nova janela para grafico
%4 GRÁFICOS: 2 LINHAS 2 COLUNAS
subplot(2,2,1)          %-------grafico linha 1 coluna 1
plot3(x,y,z)            %grafico 3D
xlabel('X')             %nome eixo X
ylabel('Y')             %nome eixo Y
zlabel('Z')             %nome eixo Z
grid on                 %habilito linhas de grid
subplot(2,2,2)          %-------grafico linha 1 coluna 2
```

```
plot(x,y,'r')           %grafico 2D
xlabel('X')             %nome eixo X
ylabel('Y')             %nome eixo Y
grid on                 %habilito linhas de grid
subplot(2,2,3)          %-------grafico linha 2 coluna 1
plot(t,y,'g')           %grafico 2D
xlabel('t')             %nome eixo X
ylabel('Y')             %nome eixo Y
grid on                 %habilito linhas de grid
subplot(2,2,4)          %-------grafico linha 2 coluna 2
plot(t,x,'c')           %grafico 2D
xlabel('t')             %nome eixo X
ylabel('X')             %nome eixo Y
grid on                 %habilito linhas de grid
set(gca,'drawmode','fast')   %habilito modo de desenhos rapido
```

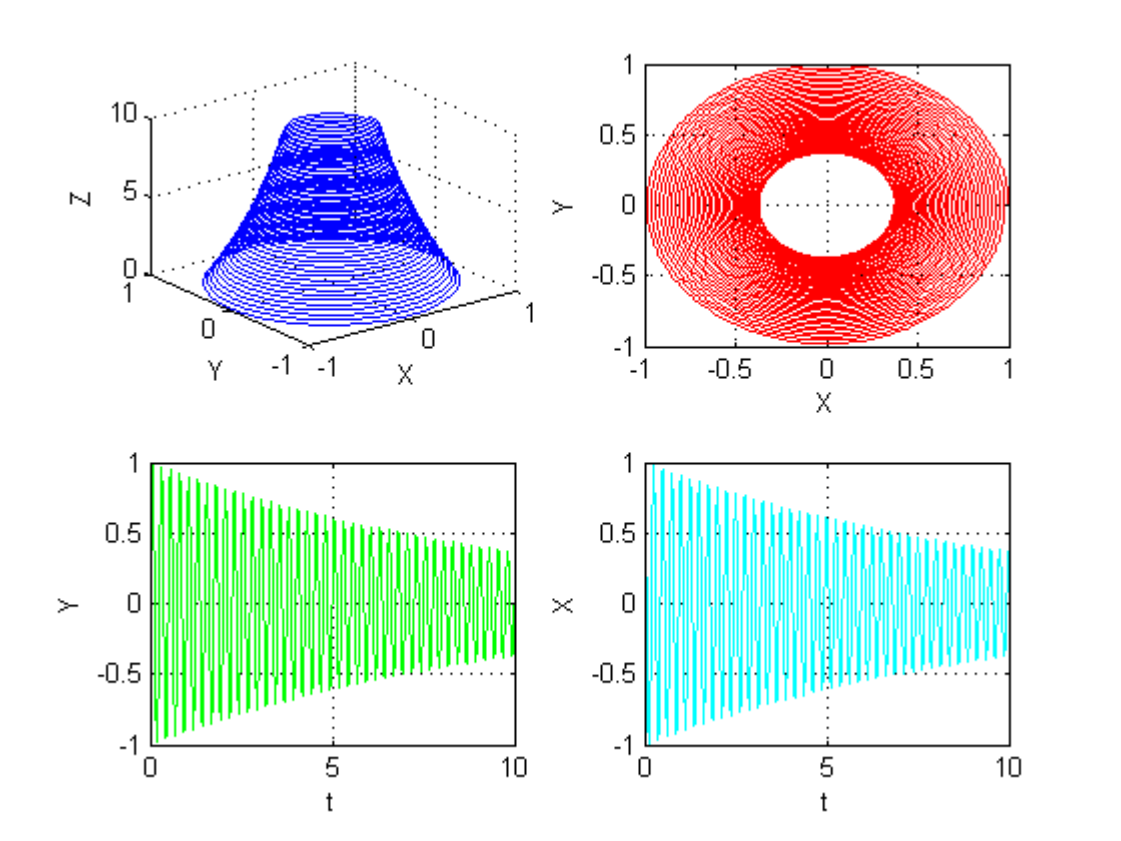


Figura 81: Gráfico obtido do exemplo com subplot

View

Função que permite ao usuário escolher a vista adequada para o plot 3D. Para configurá-lo é importante saber os ângulos azimutal (az) e de elevação (el). Segue sua representação com relação aos eixos x, y e z. *Viewpoint* representa o ponto de vista do usuário.

Importante: os ângulos devem ser fornecidos **em graus**.

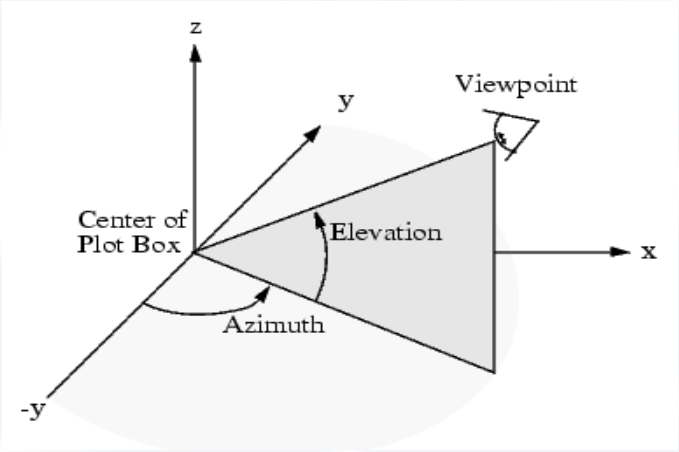


Figura 82: Ângulos azimutal e de elevação (Fonte: Help do Software)

A sintaxe utilizada deve se situar após o comando de plot:

Sintaxe	Descrição
view(az,el)	seta o Viewpoint para as coordenadas az e el
view([az,el])	seta o Viewpoint para as coordenadas az e el
view([x,y,z]	seta o Viewpoint para as coordenadas x, y e z. A magnitude dos pontos é ignorada.
view(2)	vista default 2D: az = 0, el = 90
view(3)	vista default 3D: az = -37.5, el = 30
view(T)	seta o Viewpoint com respeito à matriz de transformação T

Exemplos:

A partir do exemplo com série de Fourier.

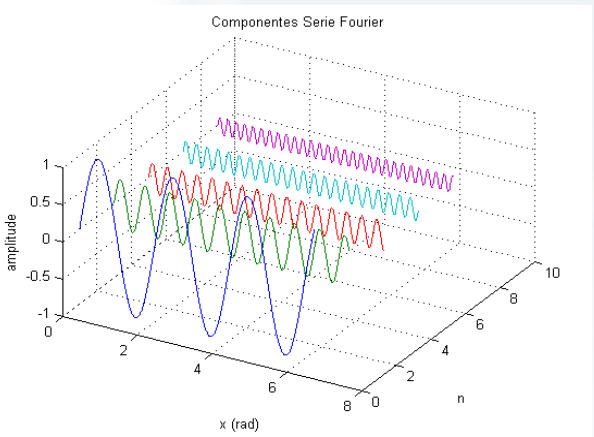
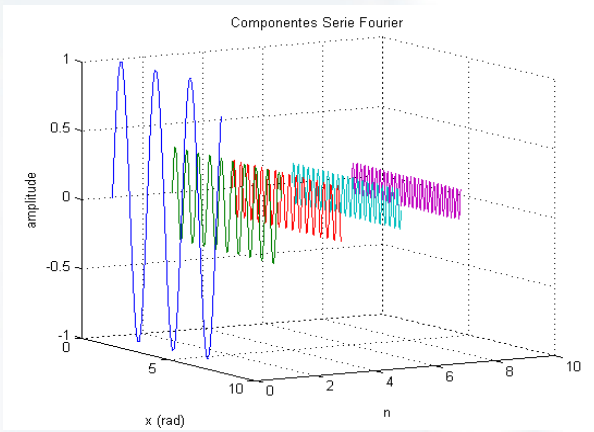
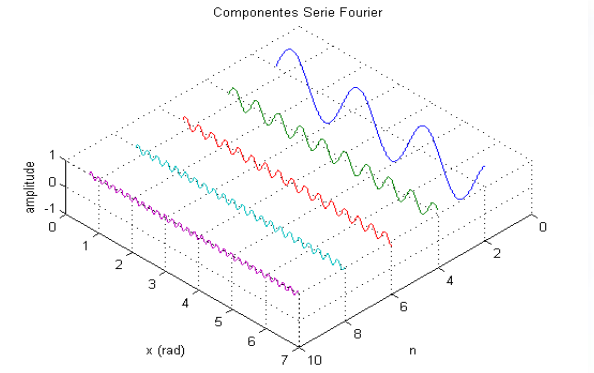
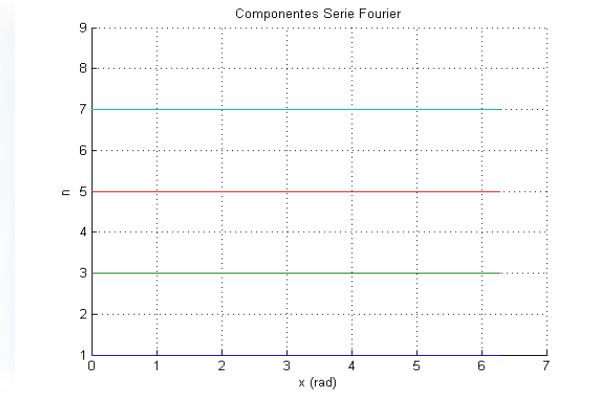
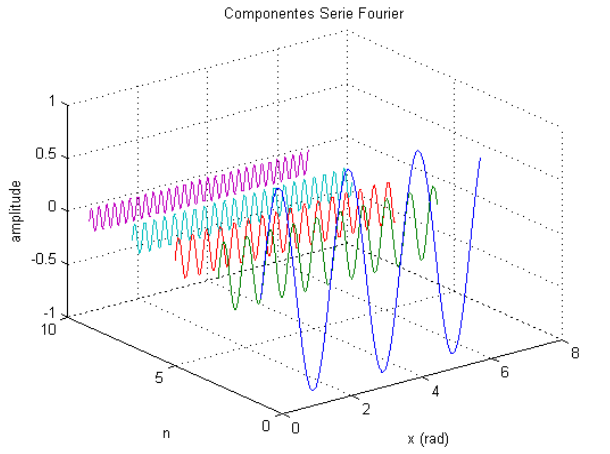
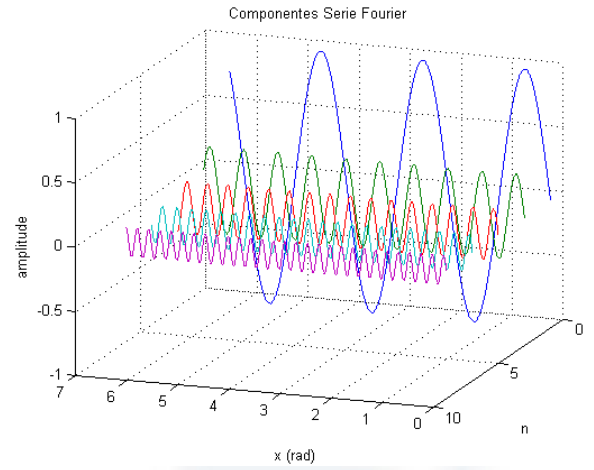
`view(30,45)`	`view([60 10])`
	
`view([-1,1,-5])`	`view(2)`
	
`view(3)`	`T = viewmtx(20,20) ; view(T);`
	

Figura 82: Modificações com o comando view

Observação:

Também é possível alterar a visualização do gráfico 3D a partir do botão *Rotate 3D*, situado na barra de ferramentas do gráfico.

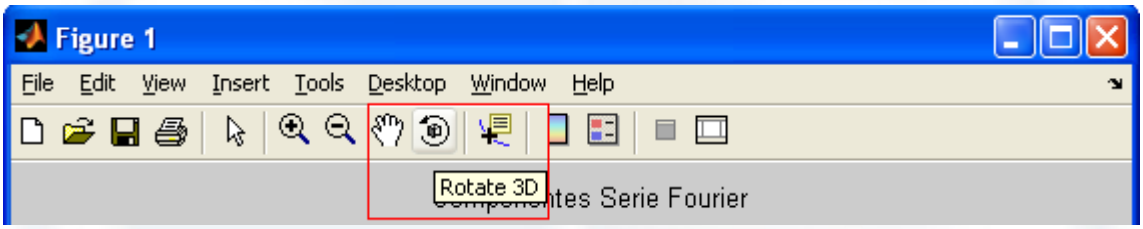


Figura 83: Botão de rotacionar o gráfico na barra de ferramentas da janela

Uma vez apertado o botão, com o movimento do mouse é possível alterar os ângulos azimutal e de elevação do gráfico:

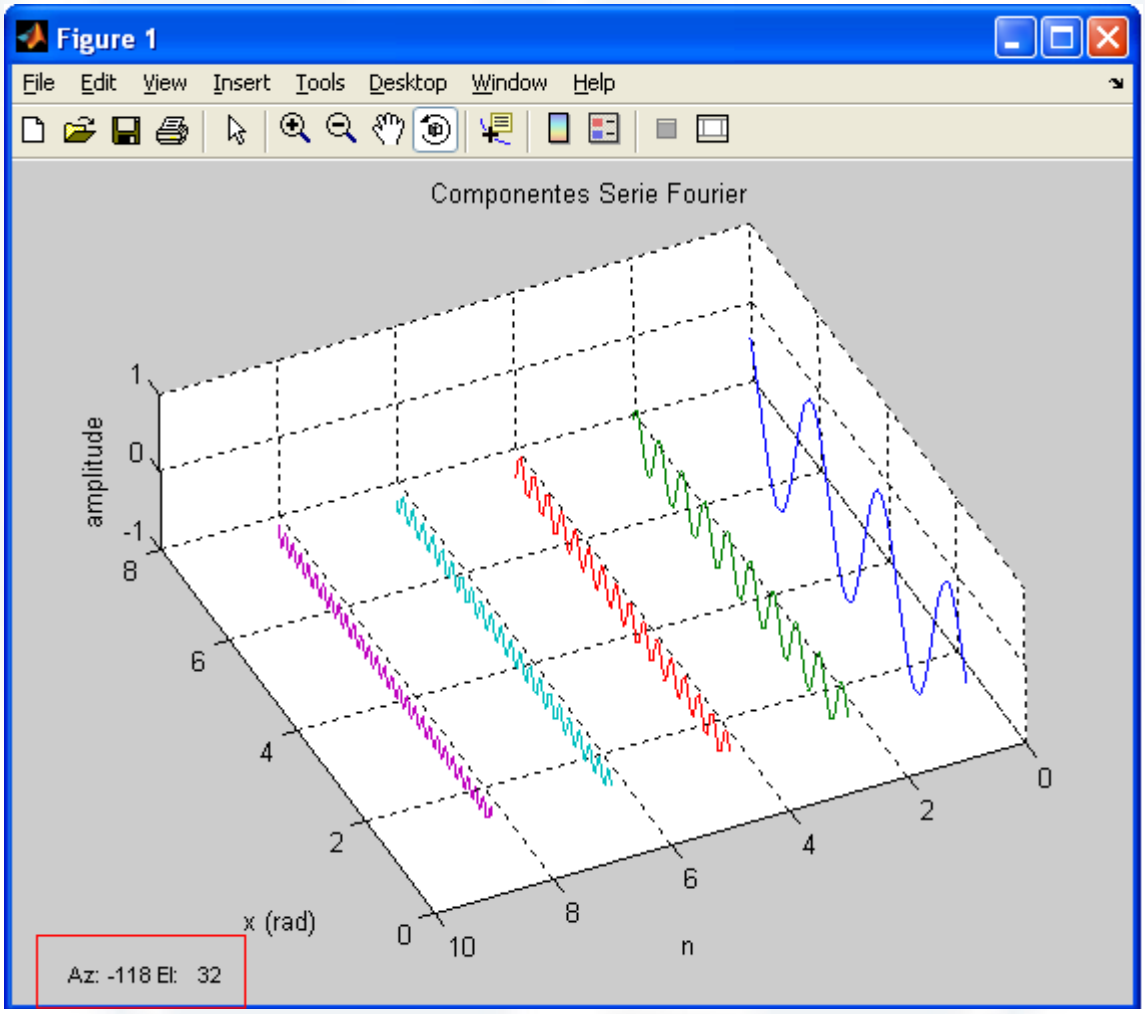


Figura 84: Gráfico 3D sendo girado com o mouse

Surf

Com o comando surf(X,Y,Z) é possível gerar um gráfico de superfície em três dimensões, sendo X e Y vetores delimitado os eixos e Z uma matriz de amplitudes para os pontos.

Exemplo:

Quádrica – Parabolóide Elíptico: $z = x^2 + y^2$

```
clear all
close all
clc

%QUADRICA - PARABOLOIDE ELIPTICO

teta = 0:0.2:2*pi;                          %angulo
x = cos(teta);                              %coordenadas em x
y = sin(teta);                              %coordenadas em y
z = zeros(size(x,2),size(y,2));             %matriz de amplitudes

%corro matriz
for lin=1:size(x,2)
   for col=1:size(y,2)
      z(lin,col) = x(lin)^2+y(col)^2;       %preencho matriz de amplitudes
   end
end

surf(x,y,z)                                 %ploto superficie
set(gca,'drawmode','fast')                  %habilito modo de desenhos rapido
```

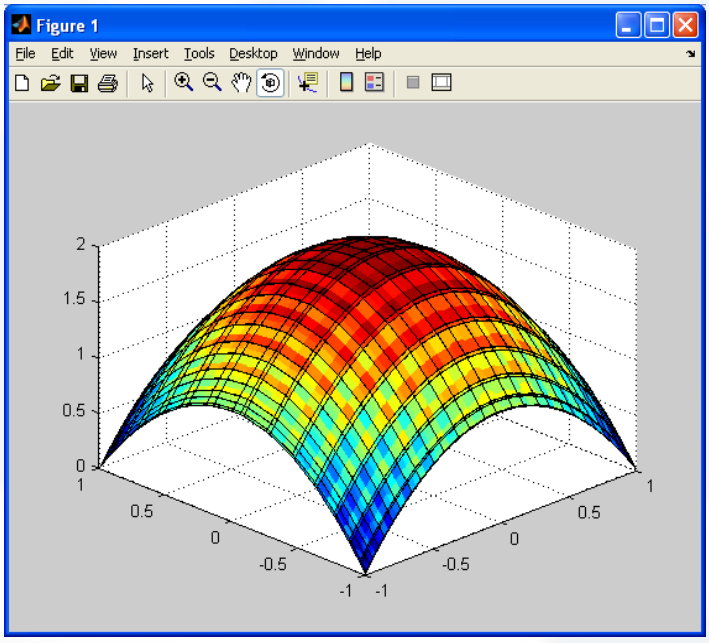


Figura 85: Parabolóide Elíptico

Principais propriedades do gráfico de superfície

Para configurar as propriedades, temos a sintaxe surf(x,y,z,'prorpriedade',valor_propriedade). Temos como principais itens:

– EdgeColor

Esta propriedade controla a cor das arestas da superfície.

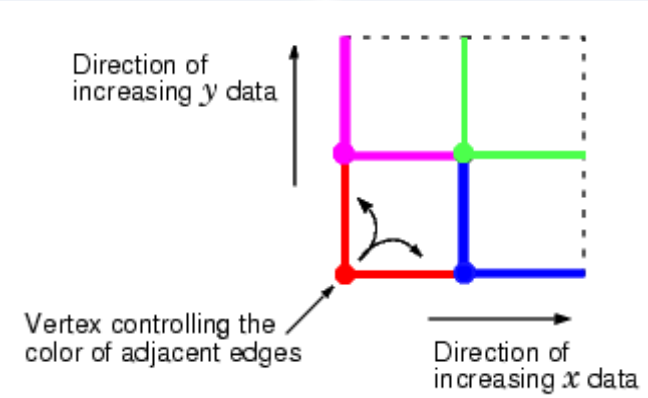


Figura 86: Desenho representativo dos vértices (Vertex) e as arestas adjacentes (Edges) (Fonte: Help do Software)

Seus valores podem ser:

Valor para propriedade	Descrição
ColorSpec	Vetor de cor RGB – por *default*, preto
none	Arestas não são desenhadas
flat	A cor de cada vértice determina a cor das arestas adjacentes (vide figura acima)
interp	Interpolação linear das cores dos ve´rtices determina a cor das arestas

Exemplo:

Quádrica – Parabolóide Hiperbólico: $z = x^2 - y^2$

```
clear all
close all
clc

%QUADRICA - PARABOLOIDE HIPERBOLICO

teta = 0:0.2:2*pi;                          %angulo
x = cos(teta);                              %coordenadas em x
y = sin(teta);                              %coordenadas em y
z = zeros(size(x,2),size(y,2));             %matriz de amplitudes

%corro matriz
```

```
for lin=1:size(x,2)
   for col=1:size(y,2)
      z(lin,col) = x(lin)^2-y(col)^2;      %preencho matriz de amplitudes
   end
end

figure
surf(x,y,z,'EdgeColor',[0 0 1])            %ploto superficie
                                           %aresta azul - RGB - [0 0 1]
set(gca,'drawmode','fast')                 %habilito modo de desenhos rapido

figure
surf(x,y,z,'EdgeColor','interp')           %ploto superficie
                                           %aresta interpolada
set(gca,'drawmode','fast')                 %habilito modo de desenhos rapido
```

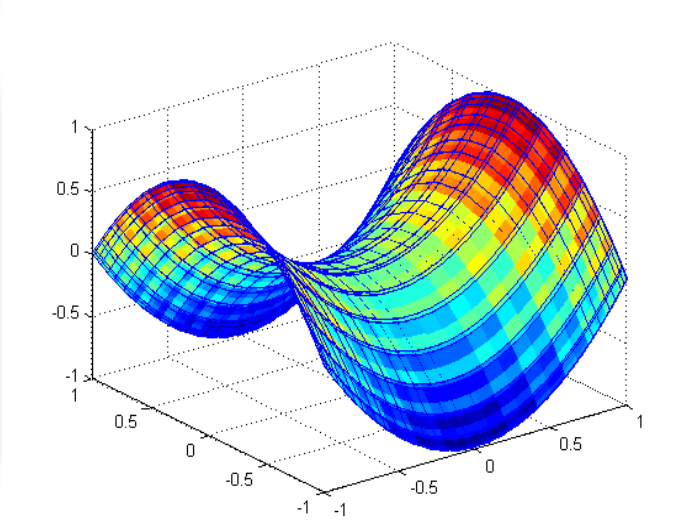


Figura 87: Parabolóide Hiperbólico com aresta em azul

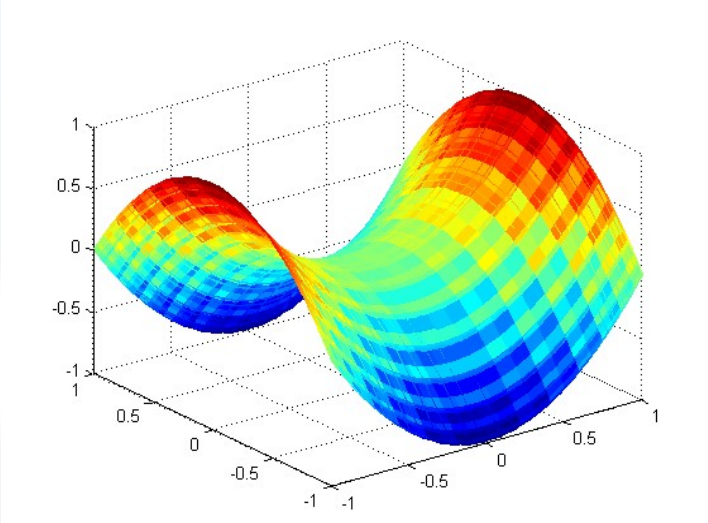


Figura 88: Parabolóide Hiperbólico com aresta interpolada

– FaceColor

Esta propriedade controla a cor das faces da superfície.

Seus valores podem ser:

Valor para propriedade	Descrição
ColorSpec	Vetor de cor RGB – por *default*, preto
none	Não pinta face da superfície
flat	A cor do primeiro vértice determina a cor da face
interp	Interpolação bilinear das cores dos vértices determina a cor da face

Exemplo:

Cone - $(\frac{x}{\sqrt{3}})^2-(\frac{y}{2})^2=z^2$

```
clear all
close all
clc

%QUADRICA - Cone

x = -2:0.1:2;
y = -2:0.1:2;
z = zeros(size(x,2),size(y,2));           %matriz de amplitudes

%corro matriz
for lin=1:size(x,2)
   for col=1:size(y,2)
      z(lin,col) = sqrt(x(lin)^2/3+y(col)^2/4);     %preencho matriz de amplitudes
   end
end

%ploto superficie
%aresta vermelha - RGB - [1 0 0]
%face            - sem preenchimento
figure
surf(x,y,z,'FaceColor','none','EdgeColor',[1 0 0])
view(-60,15)                              %seto vista
set(gca,'drawmode','fast')                %habilito modo de desenhos rapido

figure
%ploto superficie
%aresta interpolada
%face   cuja cor do primeiro ponto determina cor da face
surf(x,y,z,'FaceColor','interp','EdgeColor','flat')
view(-45,15)                              %seto vista
set(gca,'drawmode','fast')                %habilito modo de desenhos rapido
```

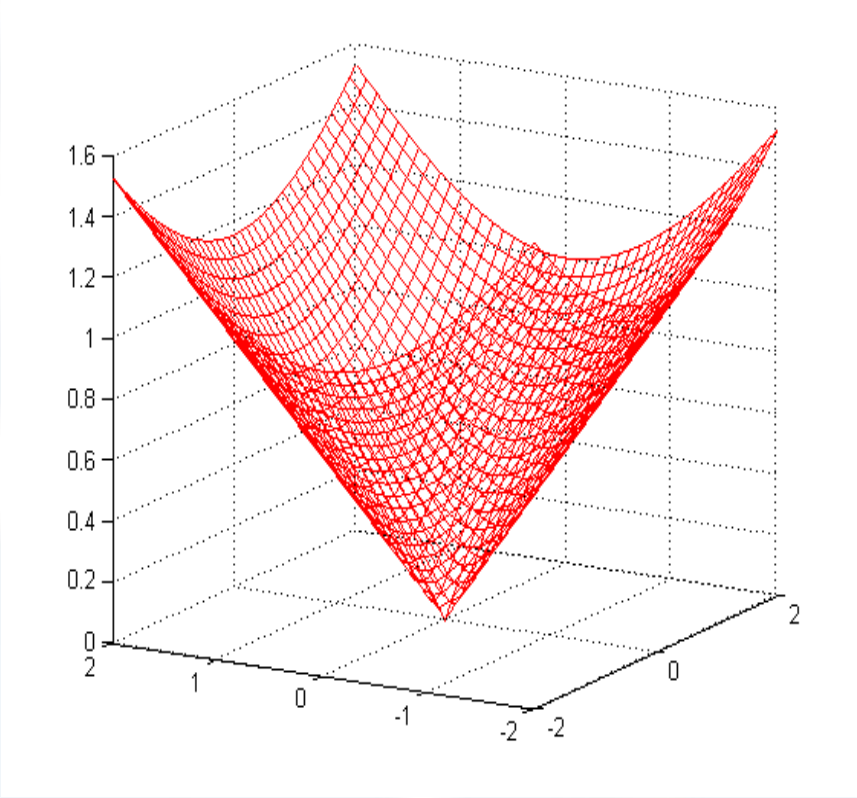


Figura 89: Cone sem preenchimento na face e com aresta vermelha

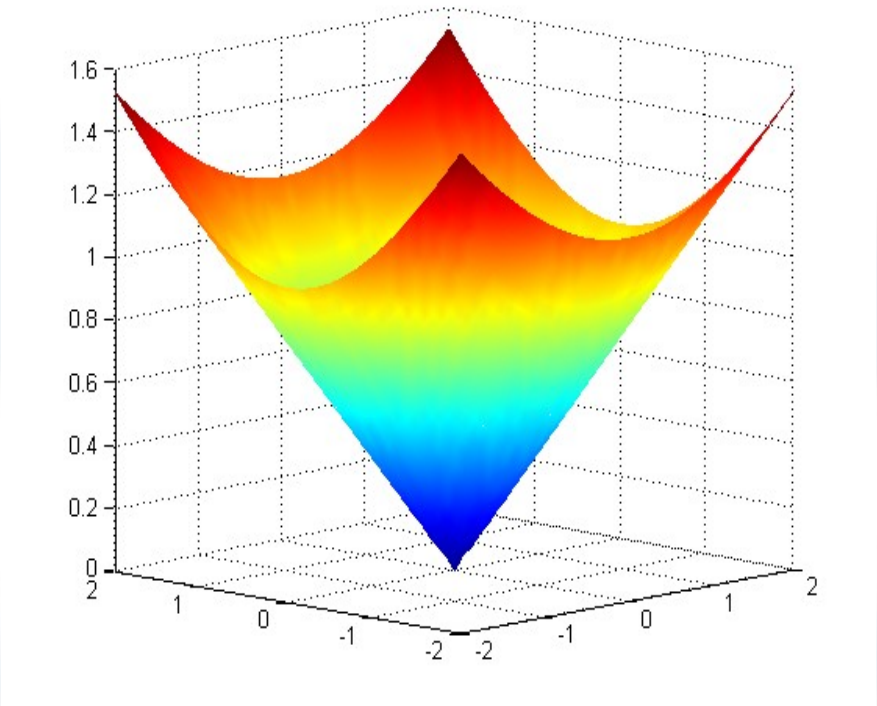


Figura 90: Cone com preenchimento interpolado na face e com aresta determinada pelo primeiro ponto

– LineStyle

Determina o tipo de linha da aresta. Pode possuir valores semelhantes aos do comando plot:

Caractere	Tipo de Linha
-	sólida
:	pontilhada
-.	traço-ponto
--	tracejada
(none)	nenhuma linha

Exemplo:

Quádrica – Parabolóide Hiperbólico: $z = (\frac{x}{5})^2 - (\frac{y}{3})^2$

```
clear all
close all
clc

%QUADRICA - Parabolóide Hiperbólico

x = -4:0.1:4;
y = -4:0.1:4;
z = zeros(size(x,2),size(y,2));           %matriz de amplitudes

%corro matriz
for lin=1:size(x,2)
   for col=1:size(y,2)
      z(lin,col) = (x(lin)/5)^2-(y(col)/3)^2 ;     %preencho matriz de amplitudes
   end
end

%ploto superficie
%aresta vermelha - RGB - [1 0 0]
%face            - sem preenchimento
%linha           - pontilhada
figure
surf(x,y,z,'FaceColor','none','EdgeColor',[1 0 0],'LineStyle',':')
set(gca,'drawmode','fast')                %habilito modo de desenhos rapido

figure
%ploto superficie
%aresta interpolada
%face            - sem preenchimento
%linha           - traço-ponto
surf(x,y,z,'FaceColor','none','EdgeColor','interp','LineStyle','-.')
set(gca,'drawmode','fast')                %habilito modo de desenhos rapido
```

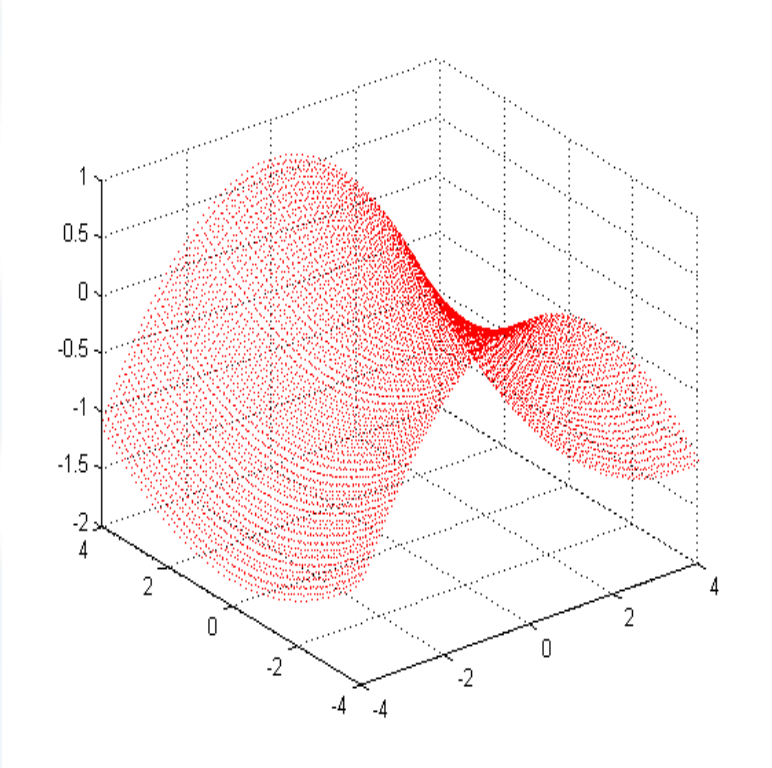


Figura 91: Parabolóide Hiperbólico – linha pontilhada

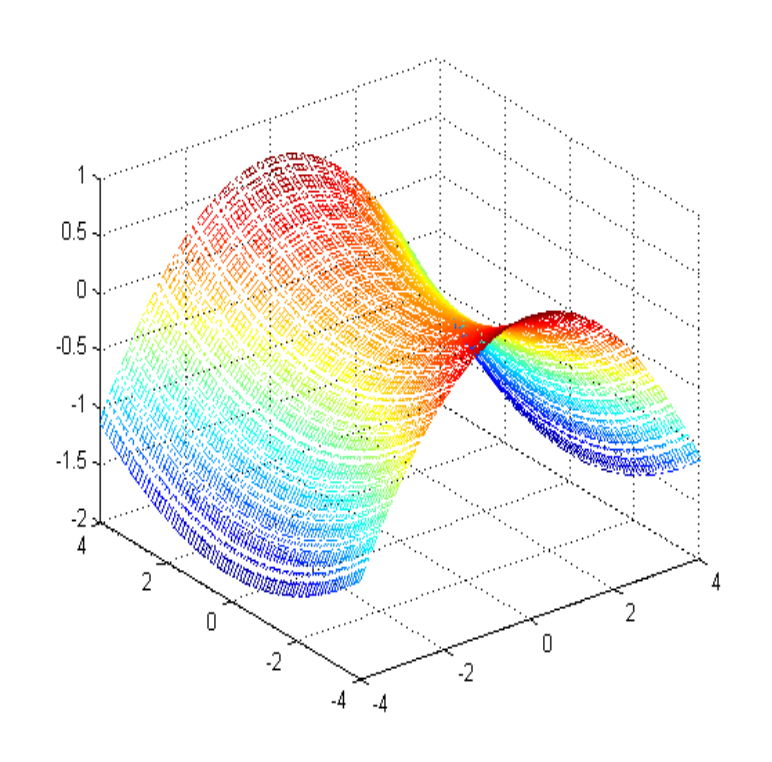


Figura 92: Parabolóide Hiperbólico – linha traço-ponto

– Marker

Altera o marcador presente nos vértices da superfície, utilizando valores semelhantes a função plot em 2D.

A seguir são mostrados os tipos de marcadores:

Caractere	Marcador
.	ponto
o	circulo
x	marca x
+	mais (cruz)
*	estrela
s	quadrado
d	diamante
v	triangulo (para baixo)
^	triangulo (para cima)
<	triangulo (para esquerda)
>	triangulo (para direita)
p	pentágono (estrela de 5 pontas)
h	hexágono

Exemplo:

Plano $z = x - y + 5$

```
clear all
close all
clc

%Plano

x = -10:1:10;
y = -10:1:10;
z = zeros(size(x,2),size(y,2));            %matriz de amplitudes

%corro matriz
for lin=1:size(x,2)
   for col=1:size(y,2)
      z(lin,col) = x(lin)-y(col) + 5 ;      %preencho matriz de amplitudes
   end
end

%ploto superficie
%aresta vermelha - RGB - [1 0 0]
%face            - sem preenchimento
%linha           - sem linha
figure
surf(x,y,z,'FaceColor','none','EdgeColor',[1 0 0],'LineStyle','none','Marker','o')
```

```
set(gca,'drawmode','fast')                  %habilito modo de desenhos rapido

figure
%ploto superficie
%aresta azul - RGB - [0 0 1]
%face            - sem preenchimento
%linha           - sem linha
surf(x,y,z,'FaceColor','none','EdgeColor',[0 0 1],'LineStyle','none','Marker','+')
set(gca,'drawmode','fast')                  %habilito modo de desenhos rapido
```

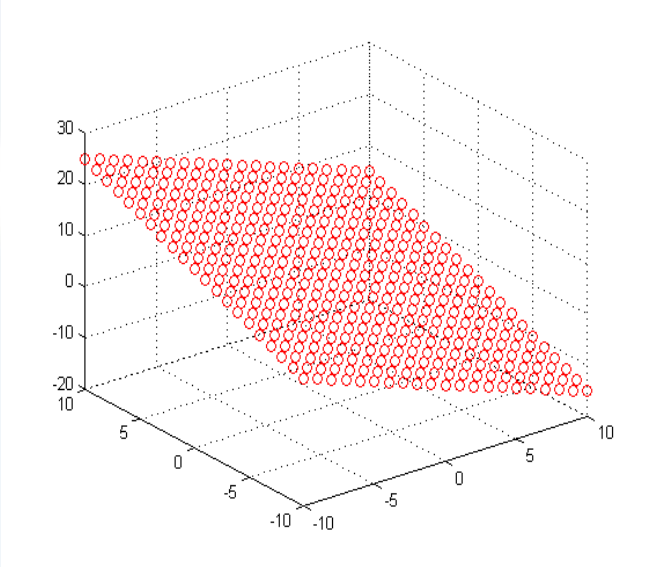


Figura 93: Plano com marcadores nos vértices tipo círculo

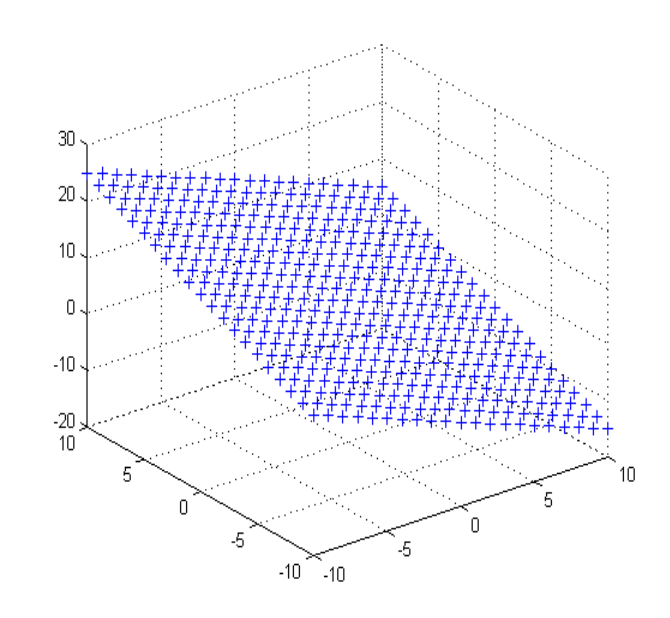


Figura 94: Plano com marcadores nos vértices tipo cruz

– Surfc

Com o comando *surfc(x,y,z)* é possível plotar as superfícies e, além disso, as curvas de nível correpondentes no plano xy.

Exemplo:

A partir dos exemplos anteriores, plotamos as curvas de nível de cone, parabolóides hiperbólicos e parabolóide elíptico.

```
clear all
close all
clc

%QUADRICA - Cone

x = -2:0.1:2;
y = -2:0.1:2;
z = zeros(size(x,2),size(y,2));           %matriz de amplitudes

%corro matriz
for lin=1:size(x,2)
   for col=1:size(y,2)
      z(lin,col) = sqrt(x(lin)^2/3+y(col)^2/4);     %preencho matriz de amplitudes
   end
end

figure
subplot(2,2,1)
surfc(x,y,z,'FaceColor','interp','EdgeColor','interp')
view(-46,10)                              %seto vista
set(gca,'drawmode','fast')                %habilito modo de desenhos rapido

%QUADRICA - PARABOLOIDE HIPERBOLICO

teta = 0:0.2:2*pi;                        %angulo
x = cos(teta);                            %coordenadas em x
y = sin(teta);                            %coordenadas em y
z = zeros(size(x,2),size(y,2));           %matriz de amplitudes

%corro matriz
for lin=1:size(x,2)
   for col=1:size(y,2)
      z(lin,col) = x(lin)^2-y(col)^2;     %preencho matriz de amplitudes
   end
end

subplot(2,2,2)
surfc(x,y,z,'FaceColor','interp','EdgeColor','interp') %ploto superficie
view(-86,39)
set(gca,'drawmode','fast')                %habilito modo de desenhos rapido

%QUADRICA - Parabolóide Eliptico

x = -4:0.1:4;
y = -4:0.1:4;
z = zeros(size(x,2),size(y,2));           %matriz de amplitudes

%corro matriz
for lin=1:size(x,2)
   for col=1:size(y,2)
      z(lin,col) = (x(lin))^2+(y(col))^2 ;     %preencho matriz de amplitudes
   end
end

subplot(2,2,3)
surfc(x,y,z,'FaceColor','interp','EdgeColor','interp')
view(-139,58)
set(gca,'drawmode','fast')                %habilito modo de desenhos rapido
```

```
%QUADRICA - Parabolóide Hiperbólico

x = -4:0.1:4;
y = -4:0.1:4;
z = zeros(size(x,2),size(y,2));           %matriz de amplitudes

%corro matriz
for lin=1:size(x,2)
   for col=1:size(y,2)
      z(lin,col) = (x(lin)/5)^2-(y(col)/3)^2 ;     %preencho matriz de amplitudes
   end
end

subplot(2,2,4)
surfc(x,y,z,'FaceColor','interp','EdgeColor','interp')
view(-7,17)
set(gca,'drawmode','fast')                %habilito modo de desenhos rapido
```

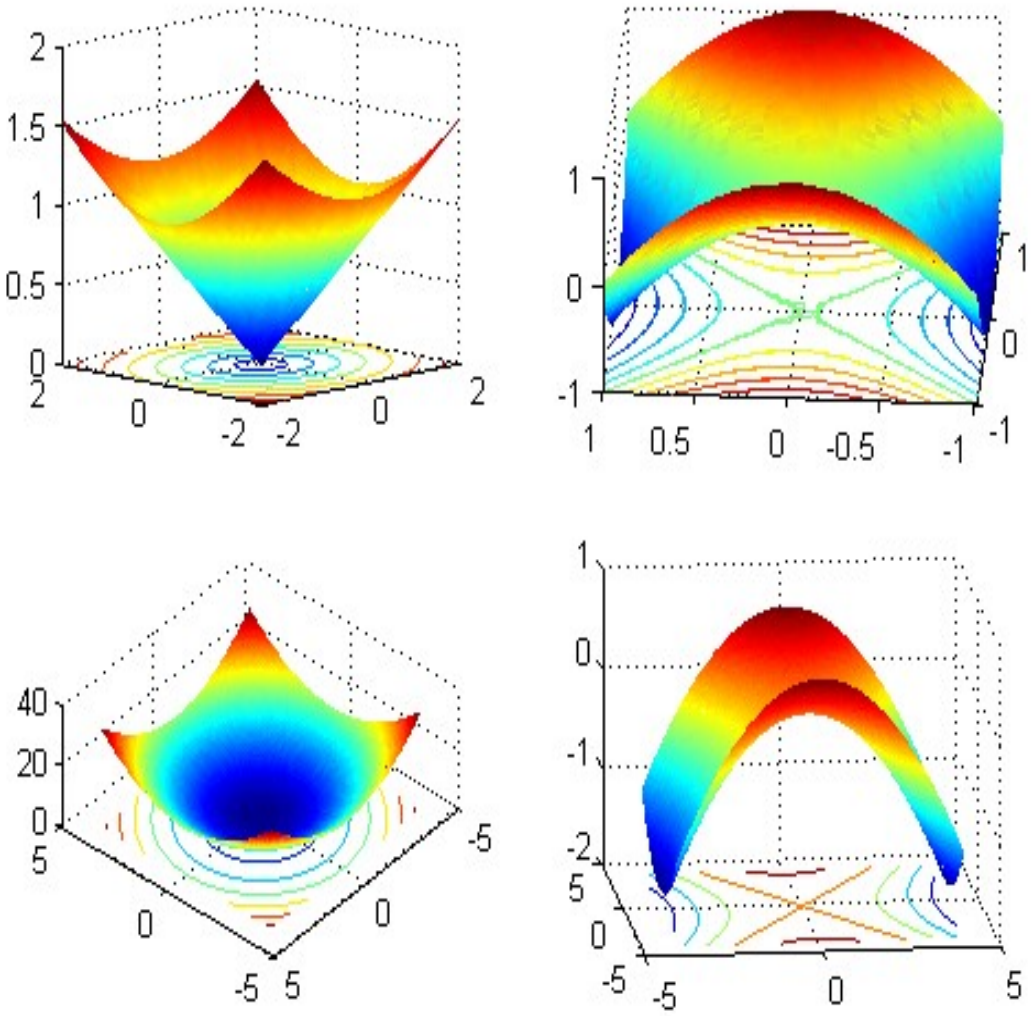


Figura 95: Curvas de Nível das Quádricas

– ColorBar

Ao se plotar superfícies, para clarificar a amplitude dos pontos, pode ser utilizada uma *ColorBar*, que relaciona as cores com as amplitudes visualizadas. O comando *colorbar* deve ser incluído logo em seguida do comando de geração da superfície.

```
clear all
close all
clc

%QUADRICA - PARABOLOIDE HIPERBOLICO

teta = 0:0.1:2*pi;                          %angulo
x = cos(teta);                              %coordenadas em x
y = sin(teta);                              %coordenadas em y
z = zeros(size(x,2),size(y,2));             %matriz de amplitudes

%corro matriz
for lin=1:size(x,2)
   for col=1:size(y,2)
      z(lin,col) = x(lin)^2-y(col)^2;       %preencho matriz de amplitudes
   end
end

figure
subplot(2,1,1)
surf(x,y,z,'FaceColor','interp','EdgeColor','interp')          %ploto superficie

subplot(2,1,2)
surf(x,y,z,'FaceColor','interp','EdgeColor','interp')          %ploto superficie
view(2)                                     %visualização em 2D
colorbar                                    %INSIRO COLORBAR NO GRÁFICO
set(gca,'drawmode','fast')                  %habilito modo de desenhos rapido
```

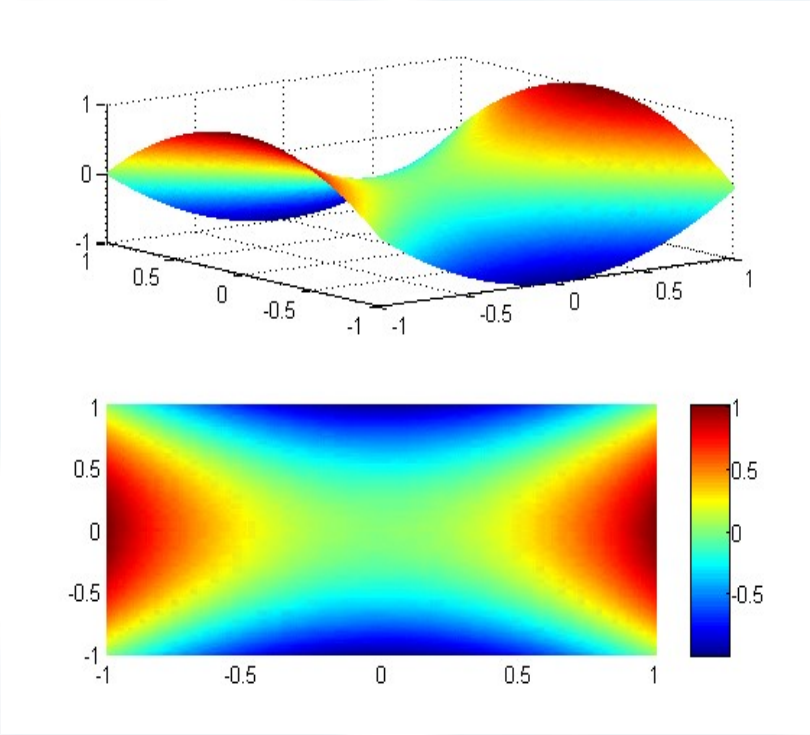


Figura 96: Colobar inserida no gráfico de visualização 2D