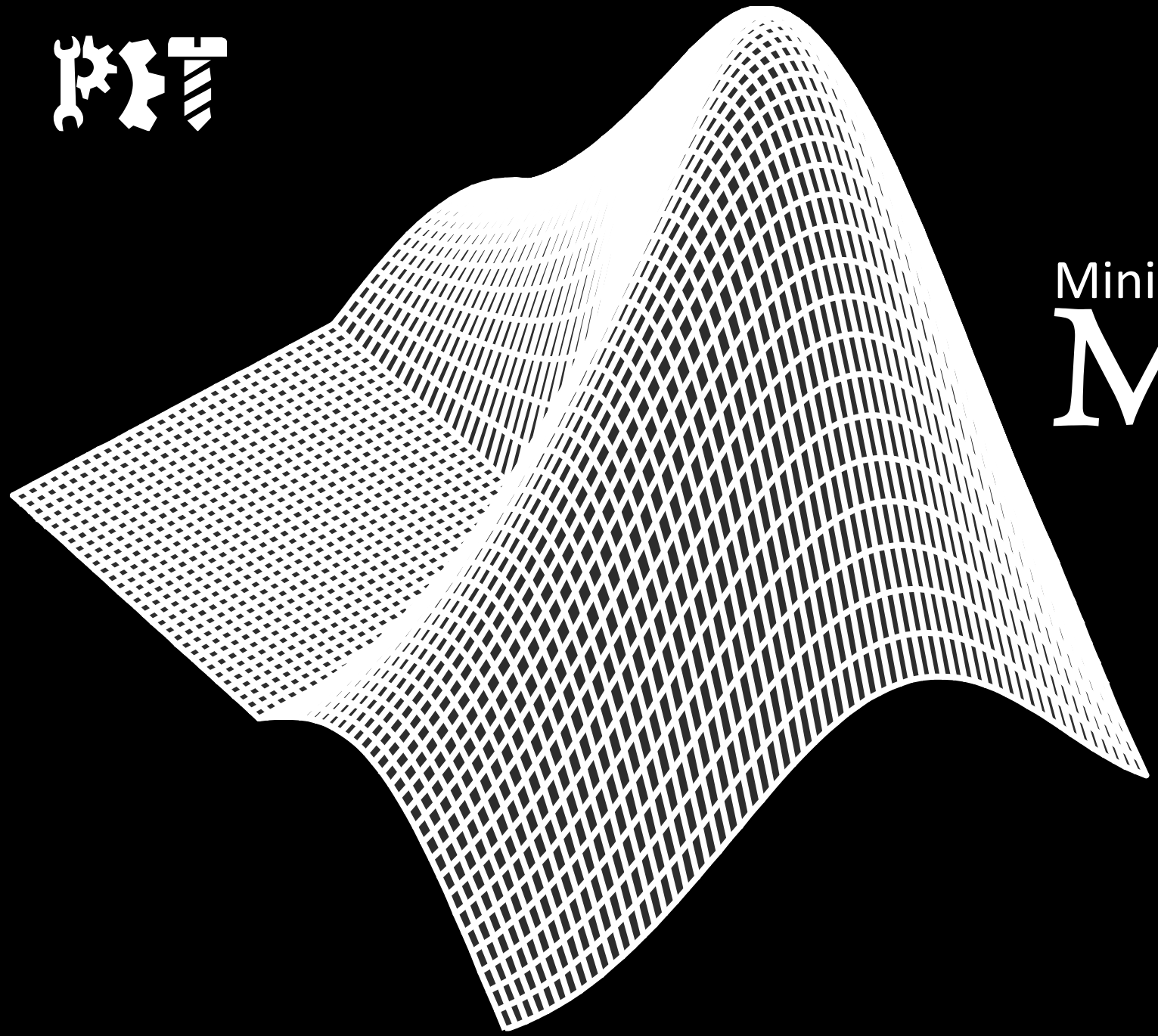


Minicurso

MATLAB

BÁSICO

MÓDULO 3

Roteiro do terceiro módulo

- Gráficos Bidimensionais e Tridimensionais
- Interpolação e Extrapolação
- Ajuste de Curvas

Gráficos Bidimensionais

O MatLab se apresenta como uma ferramenta potente e eficaz quanto a geração de informações gráficas de até quatro variáveis

Gráficos Bidimensionais – Criando um gráfico

→ figure

Abre uma nova janela para plotar gráficos

→plot(x,y)

Gera um gráfico que relaciona os vetores x e y.

Os vetores devem ter as mesmas dimensões

Gráficos Bidimensionais – Criando um gráfico

→ xlabel(‘texto_X’) e ylabel(‘texto_Y’)

Nomeia os eixos x e y

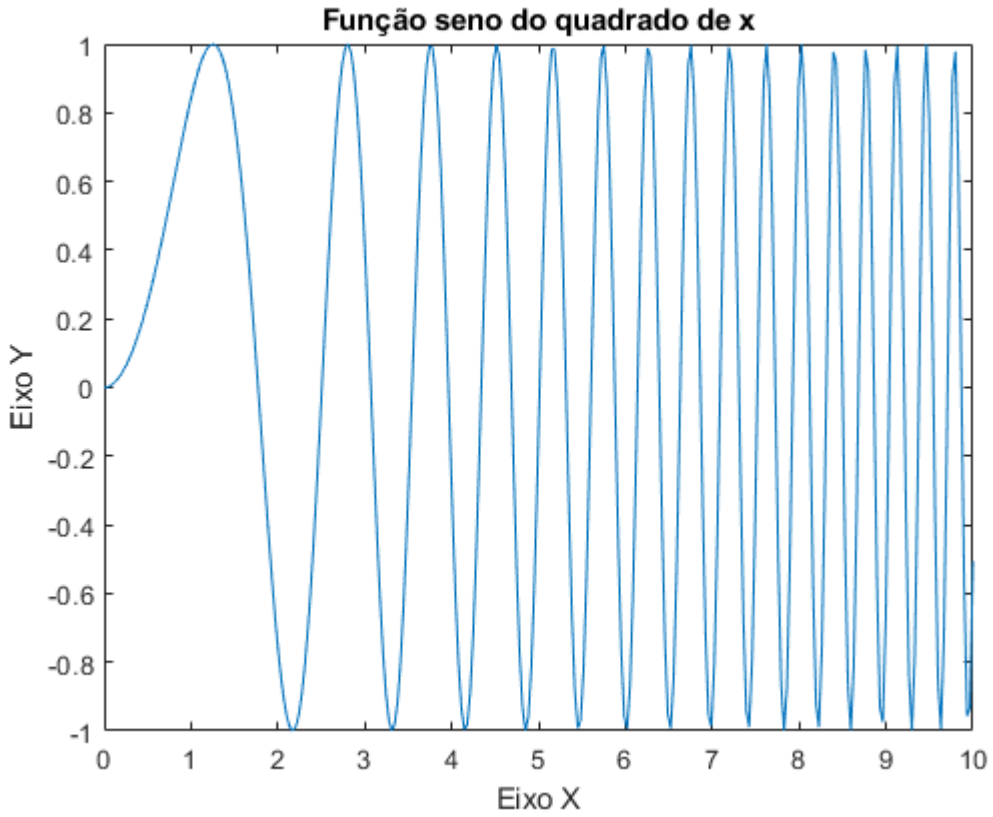
→ title(‘Texto do título’)

Define um título para o gráfico

Gráficos Bidimensionais – Criando um gráfico

→ Exemplo

```
>> x=linspace(0,10,300);
>> y=sin(x.^2);
>> figure
>> plot(x,y)
>> xlabel('Eixo X');
ylabel('Eixo Y');
>> title('Função seno do
quadrado de x');
```



Gráficos Bidimensionais – Criando um gráfico

→ xlim([lim_inferior,lim_superior]);
→ ylim([lim_inferior,lim_superior])

Permite definir os limites dos eixos exibidos na tela do gráfico

Gráficos Bidimensionais – Criando um gráfico

→ grid

Exibe a malha quadriculada no fundo do gráfico, para ajudar a visualizar as retas paralelas aos eixos. Para esconde-la, utilize grid off

Gráficos Bidimensionais – Criando um gráfico

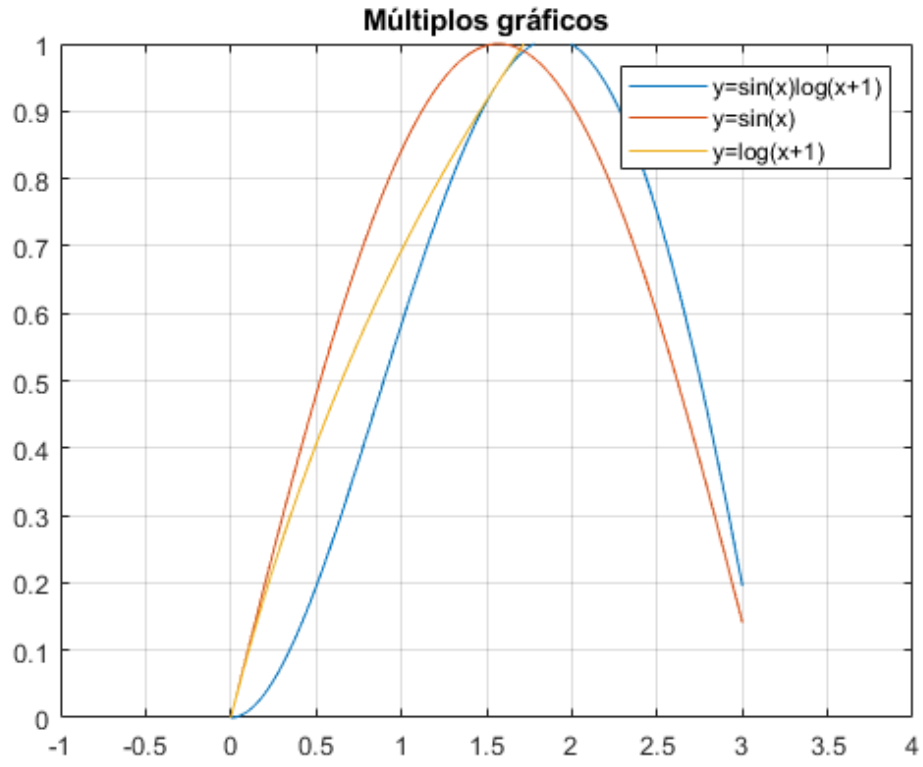
→ hold on

Armazena uma função definida em um plot no próximo comando. Assim é possível plotar mais de uma curva no mesmo gráfico

Gráficos Bidimensionais – Criando um gráfico

→ Exemplo

```
>> x=0:0.01:3;
>> y=sin(x)*log(x+1);
>> y=sin(x).*log(x+1);
>> plot(x,y)
>> grid;
>> xlim([-1 4]);
>> ylim([0 1]);
>> hold on
>> y2=sin(x);
>> y3=log(x+1);
>> plot(x,y2)
>> plot(x,y3)
>> title('Múltiplos gráficos')
>> legend('y=sin(x)log(x+1)','y=sin(x)','y=log(x+1)')
```



Gráficos Bidimensionais – Criando um gráfico

→ ‘Color’,parâmetro

Há três formas de se definir a cor de uma linha, ou de um elemento gráfico, pelo nome, pelo atalho, ou pelo array de fração de RGB.

[1 1 0]	y	yellow
[1 0 1]	m	magenta
[1 0 0]	r	red
[0 1 0]	g	green
[0 0 1]	b	blue

Outros atalhos para cores

cyan(c)
[.3 0 1] = 30% de vermelho, 0% de verde, 100% de azul

white(w)
[.2 .5 1] = 20% de vermelho, 50% de verde, 100% de azul

black(k)
[.9 .42 .2] = 90% de vermelho, 42% de verde, 20% de azul

Gráficos Bidimensionais – Criando um gráfico

→ Estilo da linha

Dentro do comando plot, pode-se atribuir à linha de gráfico um estilo de linha ou marcador para representar os pontos, para alterar o estilo

Gráficos Bidimensionais – Criando um gráfico

→ Estilo da linha

Comandos	Descrição
‘-’	Linha cheia
‘--’	Linha tracejada
‘:’	Linha pontilhada
‘-.’	Linha traço-ponto
‘+’	Marcador em cruz
‘o’	Marcador circular
‘*’	Marcador em asterisco
‘X’	Marcador em x

Gráficos Bidimensionais – Criando um gráfico

→ Estilo da linha

Comandos	Descrição
‘square’ ou ‘s’	Marcador quadrado
‘diamond’ ou ‘d’	Marcador em diamante
‘^’	Marcador triangular para cima
‘v’	Marcador triangular para baixo
‘>’	Marcador triangular para direita
‘<’	Marcador triangular para esquerda
‘pentagram’ ou ‘p’	Marcador estrela 5 pontas
‘hexagram’ ou ‘h’	Marcador estrela de 6 pontas

Gráficos Bidimensionais – Criando um gráfico

→ ‘linewidth’, valor_peso

Altera o peso da linha no gráfico, o número que sucede ‘linewidth’ é o peso associado.

Gráficos Bidimensionais – Criando um gráfico

subplot(m,n,p)

Plota vários gráficos na mesma janela de figura. Os valores de *m* e *n* devem ser fixados, eles designam a quantidade de gráficos que será mostrada na tela em que m é o número de gráficos por linha e n é o número de gráficos por coluna. O valor de p varia de acordo com a posição que é desejada mostrar.

Gráficos Bidimensionais – Criando um gráfico

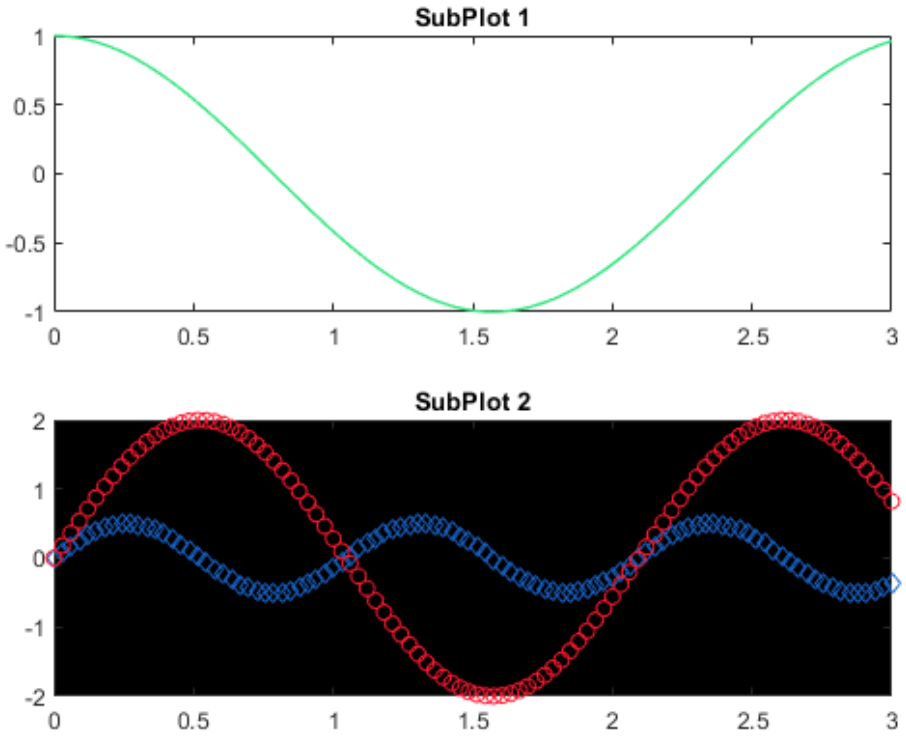
→ set(gca, ‘Color’, parâmetro)

Muda a cor do background

Gráficos Bidimensionais – Criando um gráfico

→ Exemplo

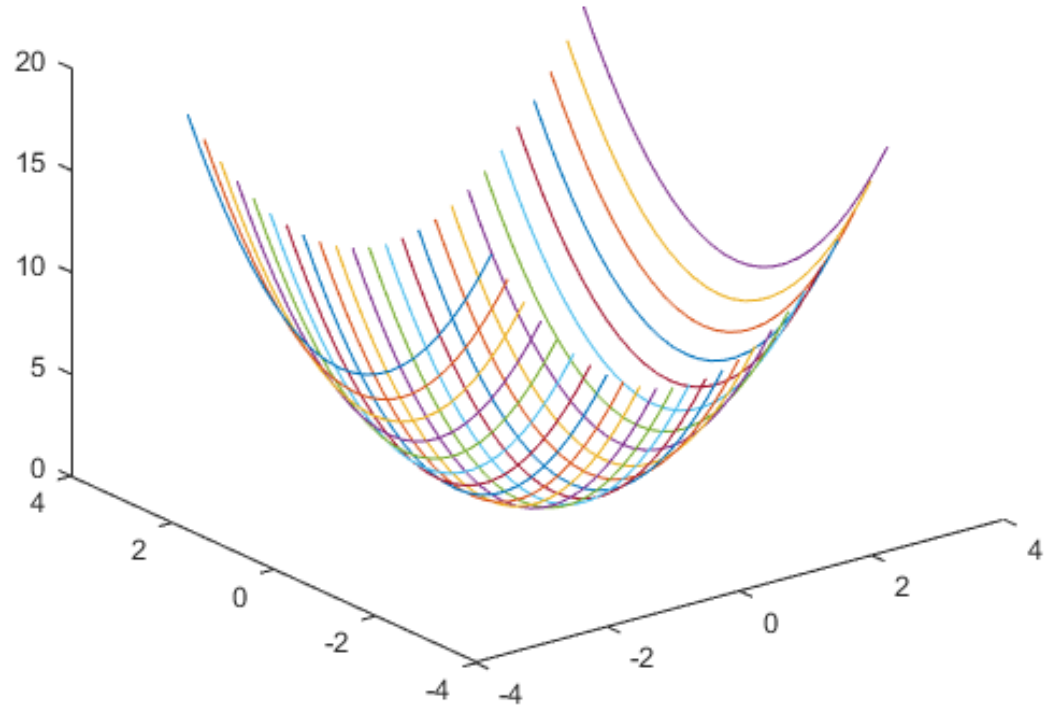
```
x=linspace(0,3,100)
y=sin(3*x)
z=cos(2*x)
w=1/2*sin(6*x)

sub1=subplot(2,1,1)
sub2=subplot(2,1,2)

plot(sub1,x,z,'Linewidth',1,'Color',[.2 .9 .5])
title(sub1,'SubPlot 1')
plot(sub2,x,w,'d','Linewidth',0.5,'Color',[.1 .4 .8])
hold on;
plot(sub2,x,2*y,'o','Linewidth',0.5,'Color',[1 .1 .2])
set(gca,'Color','k')
title(sub2,'SubPlot 2')
```



Gráficos Tridimensionais – Criando um gráfico

→ plot3(x,y,z)

Plota uma curva tridimensional em que os vetores x, y e z devem ser discretizados



Gráficos Tridimensionais – Criando um gráfico

→ [X,Y]= meshgrid(x,y)

Uma vez definida a discretização dos valores de x e y, deseja-se criar uma matriz de valores para gerar um conjunto de pontos em um domínio retangular para associar a uma imagem Z. Para isso é utilizado o comando meshgrid que vai gerar matrizes X e Y que correspondem ao conjunto de pontos neste domínio retangular

Gráficos Tridimensionais – Criando um gráfico

→ Exemplo

```
>> x=linspace(-3,3,10);
>> y=linspace(-3,3,10);
>> [X,Y]=meshgrid(x,y)

X =

   -3.0000   -2.3333   -1.6667   -1.0000   -0.3333    0.3333    1.0000    1.6667    2.3333    3.0000
   -3.0000   -2.3333   -1.6667   -1.0000   -0.3333    0.3333    1.0000    1.6667    2.3333    3.0000
   -3.0000   -2.3333   -1.6667   -1.0000   -0.3333    0.3333    1.0000    1.6667    2.3333    3.0000
   -3.0000   -2.3333   -1.6667   -1.0000   -0.3333    0.3333    1.0000    1.6667    2.3333    3.0000
   -3.0000   -2.3333   -1.6667   -1.0000   -0.3333    0.3333    1.0000    1.6667    2.3333    3.0000
   -3.0000   -2.3333   -1.6667   -1.0000   -0.3333    0.3333    1.0000    1.6667    2.3333    3.0000
   -3.0000   -2.3333   -1.6667   -1.0000   -0.3333    0.3333    1.0000    1.6667    2.3333    3.0000
   -3.0000   -2.3333   -1.6667   -1.0000   -0.3333    0.3333    1.0000    1.6667    2.3333    3.0000
   -3.0000   -2.3333   -1.6667   -1.0000   -0.3333    0.3333    1.0000    1.6667    2.3333    3.0000
   -3.0000   -2.3333   -1.6667   -1.0000   -0.3333    0.3333    1.0000    1.6667    2.3333    3.0000


Y =

   -3.0000   -3.0000   -3.0000   -3.0000   -3.0000   -3.0000   -3.0000   -3.0000   -3.0000   -3.0000
   -2.3333   -2.3333   -2.3333   -2.3333   -2.3333   -2.3333   -2.3333   -2.3333   -2.3333   -2.3333
   -1.6667   -1.6667   -1.6667   -1.6667   -1.6667   -1.6667   -1.6667   -1.6667   -1.6667   -1.6667
   -1.0000   -1.0000   -1.0000   -1.0000   -1.0000   -1.0000   -1.0000   -1.0000   -1.0000   -1.0000
   -0.3333   -0.3333   -0.3333   -0.3333   -0.3333   -0.3333   -0.3333   -0.3333   -0.3333   -0.3333
    0.3333    0.3333    0.3333    0.3333    0.3333    0.3333    0.3333    0.3333    0.3333    0.3333
    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
    1.6667    1.6667    1.6667    1.6667    1.6667    1.6667    1.6667    1.6667    1.6667    1.6667
    2.3333    2.3333    2.3333    2.3333    2.3333    2.3333    2.3333    2.3333    2.3333    2.3333
    3.0000    3.0000    3.0000    3.0000    3.0000    3.0000    3.0000    3.0000    3.0000    3.0000
```

Gráficos Tridimensionais – Criando um gráfico

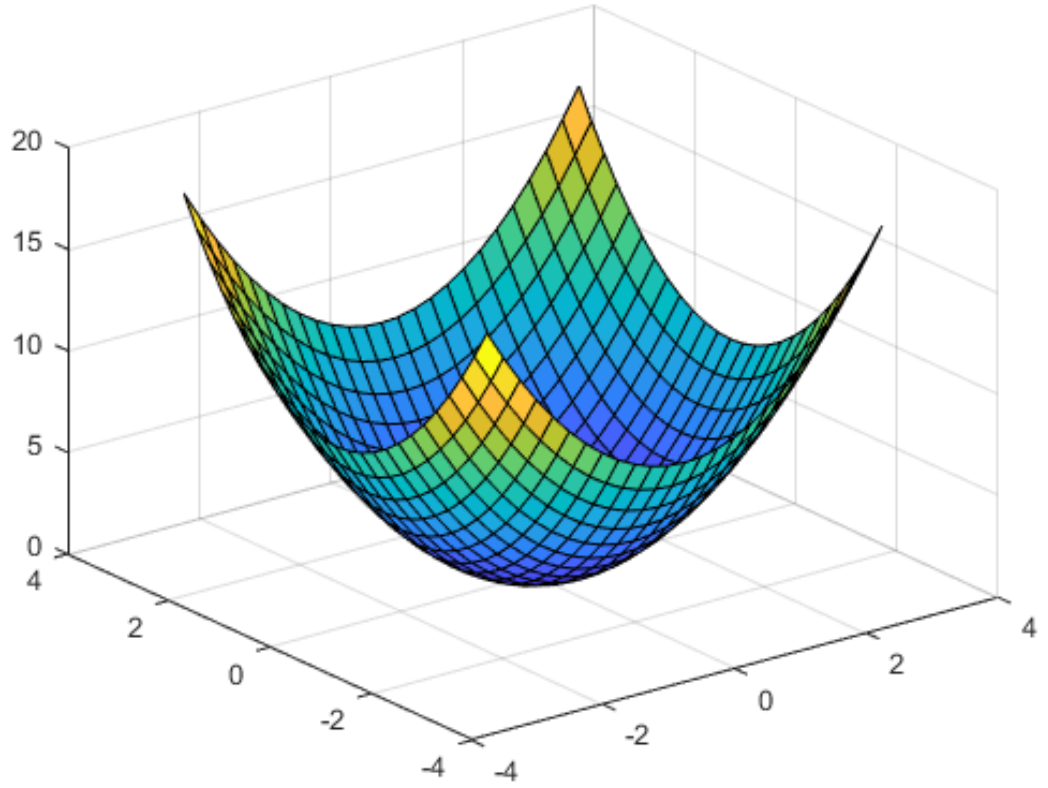
→ surf(X,Y,Z)

Plota uma superfície. As matrizes X, Y e Z devem ter o mesmo tamanho

Gráficos Tridimensionais – Criando um gráfico

→ Exemplo

```
>> Z=X.^2+Y.^2
>> surf(X,Y,Z)
```



Gráficos Tridimensionais – Criando um gráfico

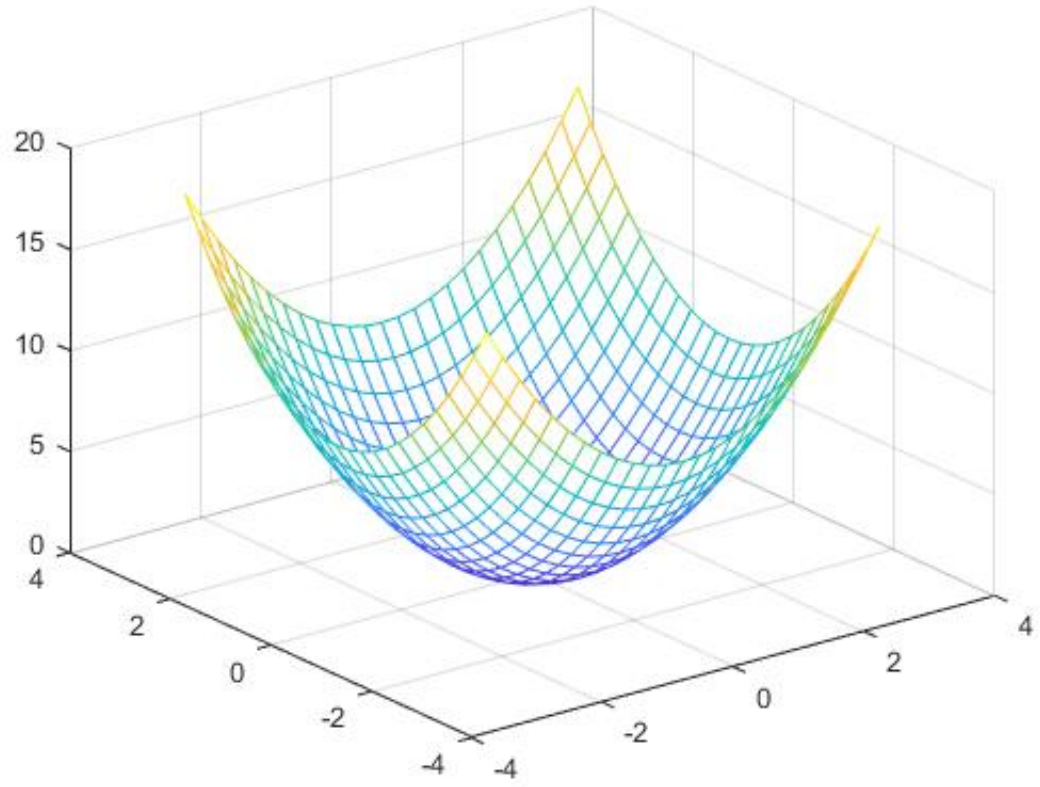
→ mesh(X,Y,Z)

Enquanto o surf gera uma superfície entre os pontos, o comando mesh mostrará os pontos e elementos da função Z(X,Y)

Gráficos Tridimensionais – Criando um gráfico

→ Exemplo

```
>> Z=X.^2+Y.^2
>> plot(X,Y,Z)
```



Gráficos Tridimensionais – Criando um gráfico

→ view(az,el)

Especifica os ângulos associados à vista do gráfico com precisão utilizando os parâmetros *az*(ângulo azimutal) e *el*(ângulo de elevação). Os ângulos são dados em graus.

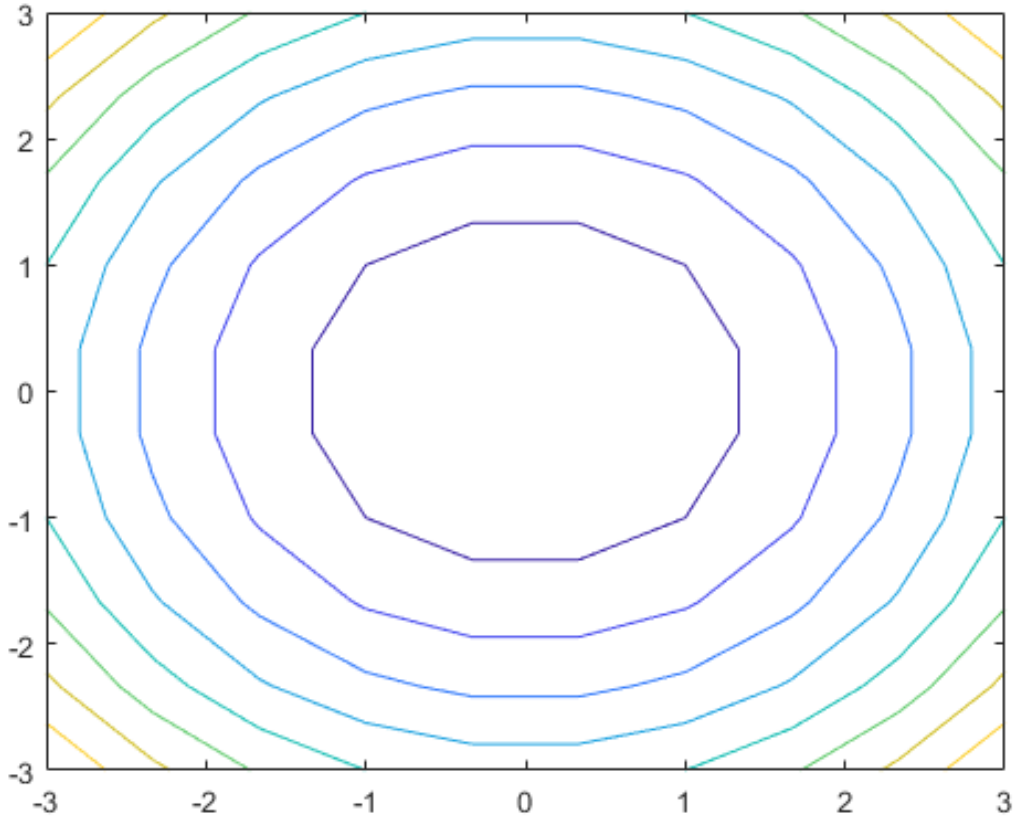
```
>> surf(X,Y,Z)
>> view(60,30)
```

Gráficos Tridimensionais – Criando um gráfico

→ contour(X,Y,Z)

Cria um mapa de curvas de níveis bidimensionais.

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



Interpolação e Extrapolação

Quando se trabalha com um conjunto de dados coletados de maneira discreta, por vezes, é vantajoso pensar em uma função analítica contínua que representa aquele conjunto de dados da melhor maneira possível. Aproximações podem ser desenvolvidas para se obter estimativas de $f(x)$, em valores de que não constam do conjunto conhecido.

Interpolação e Extrapolação

→ Interpolação de curva

Determinar uma estimativa de um valor x_i pertencente ao intervalo de dados que se localiza entre o valor mínimo e máximo conhecido.

$$X_1 \leq x_i \leq X_N$$

→ Extrapolação

Predizer sobre o valor de um dado x_i fora do conjunto de dados conhecido

$$x_i \leq X_1 \quad \text{ou} \quad x_i \geq X_N$$

→ Interpolação e Extrapolação polinomial

Suponha que tenha sido importada para o MatLab uma tabela contendo os seguintes valores para um dado experimento que relaciona a profundidade de um tubo (P), em metro, com a temperatura(T), em Kelvin. Para realizar uma interpolação unidimensional $y = f(x)$ no nosso conjunto de dados basta utilizar interp1(P,T,p) em que p é um valor de P não contido no conjunto fornecido.

Interpolação e Extrapolação

→ Função Interp1

Sintaxes:

Para interpolação:

```
interp1(x,Y,xi) ou
interp1(x,Y,xi,method);
```

Para extrapolação:

```
interp1(x,Y,xi,method,'extrap')
```

A string ‘extrap’ é adicionada para extrapolação.

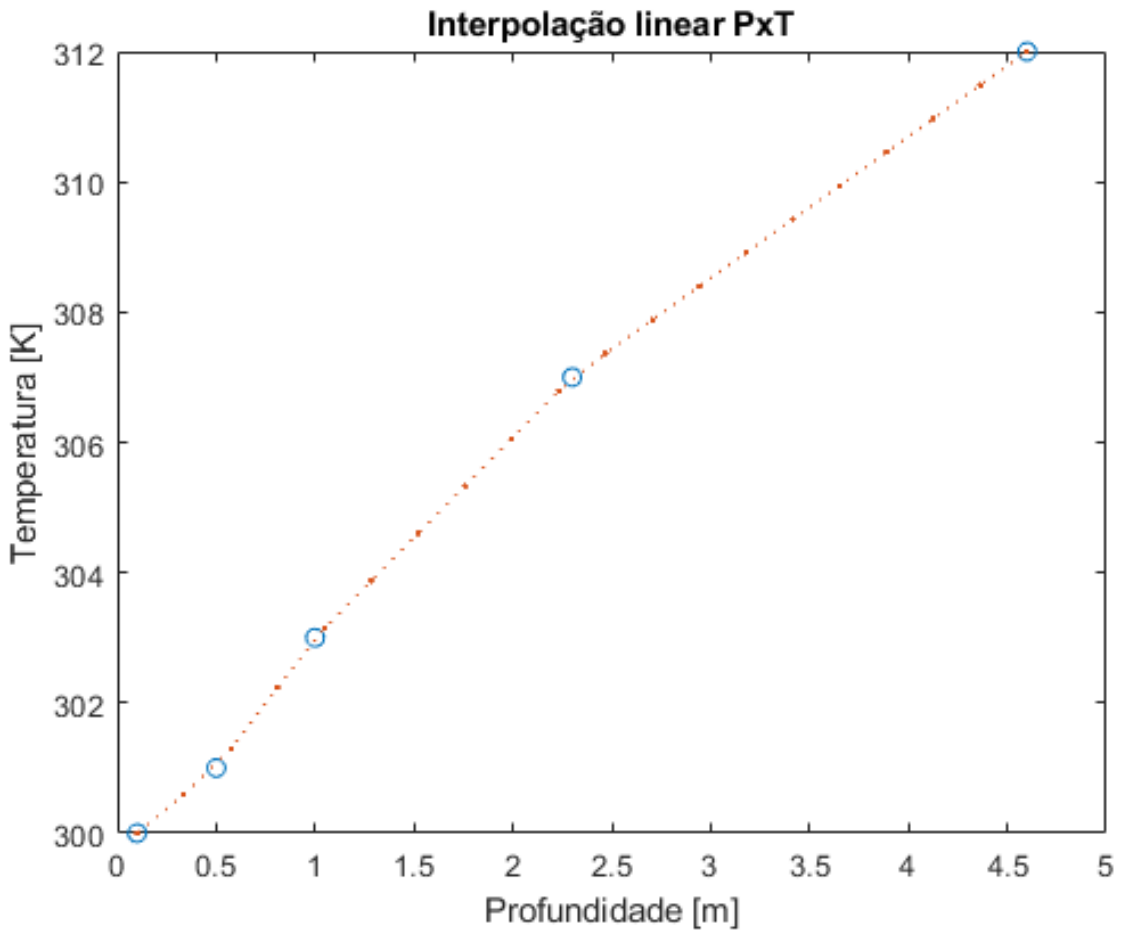
Entrada	Descrição	
x, Y	Vetor contendo os pontos conhecido	
x_i	Pode ser um escalar ou um vetor unidimensional ou bidimensional.	
method	’nearest’	Interpolação vizinha mais próxima
	‘linear’	Interpolação linear
	‘spline’	Spline cúbico
	‘pchip’	Interpolação de Hermite
	‘cubic’	Interpolação cúbica
	‘v5cubic’	Interpolação cúbica matlab

Interpolação e Extrapolação

→ interp1(P,T,p)

```
>> P=[0.1;0.5;1;2.3;4.6]

P =

    0.1000
    0.5000
    1.0000
    2.3000
    4.6000

>> T=[300;301;303;307;312]

T =

   300
   301
   303
   307
   312

>> interp1(P,T,3.5)

ans =

  309.6087

>> interp1(P,T,'pchip',4.7)

ans =

  312.1565
```

Interpolação e ajuste de curvas – Interpolação polinomial



Interpolação e ajuste de curvas – Interpolação polinomial

Na interpolação, normalmente fazemos uma conexão entre alguns pontos, por um tipo específico de função.

Para interpolar um polinômio de grau N a um conjunto de N+1 pontos, resolve-se a equação linear:

$$a_0 + a_1 x_i + a_2 x_i^2 + \cdots + a_{n+1} x_i^n = y_i$$

Que é resolvida em um sistema de N+1 equações, uma equação para cada ponto i={1,...N+1}.

Interpolação e ajuste de curvas – Interpolação polinomial

Para interpolar um segmento de reta, são necessários dois pontos. Para interpolar uma parábola são necessários 3 pontos

$$y_1 = ax_1^2 + bx_1 + c$$
$$y_2 = ax_2^2 + bx_2 + c$$
$$y_3 = ax_3^2 + bx_3 + c$$

E para polinômios de grau n, são necessários n+1 pontos.

Interpolação e ajuste de curvas – Ajuste de curvas

→ Ajuste de curvas pelo comando *polyfit*
→ p=polyfit(x,y,n）

p = o vetor que receberá os **coeficientes** do polinômio de ajuste
x = o vetor que contém os valores das abcissas dos pontos que serão ajustados
y = o vetor que contém os valores das ordenadas dos pontos que serão ajustados

Interpolação e ajuste de curvas – Ajuste de curvas

→ Ajuste de curvas pelo comando *polyfit*
→ p=polyfit(x,y,n)

n = indica o grau do polinômio que se deseja o ajuste (n=1, 2, 3...)

→ Exemplo

```
>> x = [1 2 3 5];
>> y = [2 4 6 11];
>> p = polyfit(x,y,1)

p =

    2.2571   -0.4571
```

Interpolação e ajuste de curvas – Ajuste de curvas

→ Ajuste de curva para funções que não são polinomiais

→ Potência: $y = bx^m$

```
>> p=polyfit(log(x),log(y),1)
```

→ Exponencial: $y = be^{mx}$

```
>> p=polyfit(x,log(y),1)
```

$y = b10^{mx}$

```
>> p=polyfit(x,log10(y),1)
```

Interpolação e ajuste de curvas – Ajuste de curvas

→ Ajuste de curva para funções que não são polinomiais

→ Logarítimica: $y = m \ln x + b$

```
>> p=polyfit(log(x),y,1)
```

$$y = m \log x + b$$

```
>> p=polyfit(log10(x),y,1)
```

Interpolação e ajuste de curvas – Ajuste de curvas

→ Ajuste de curva para funções que não são polinomiais

→ Hiperbólica: $y = \frac{1}{mx+b}$

```
>> p=polyfit(x,1./y,1)
```