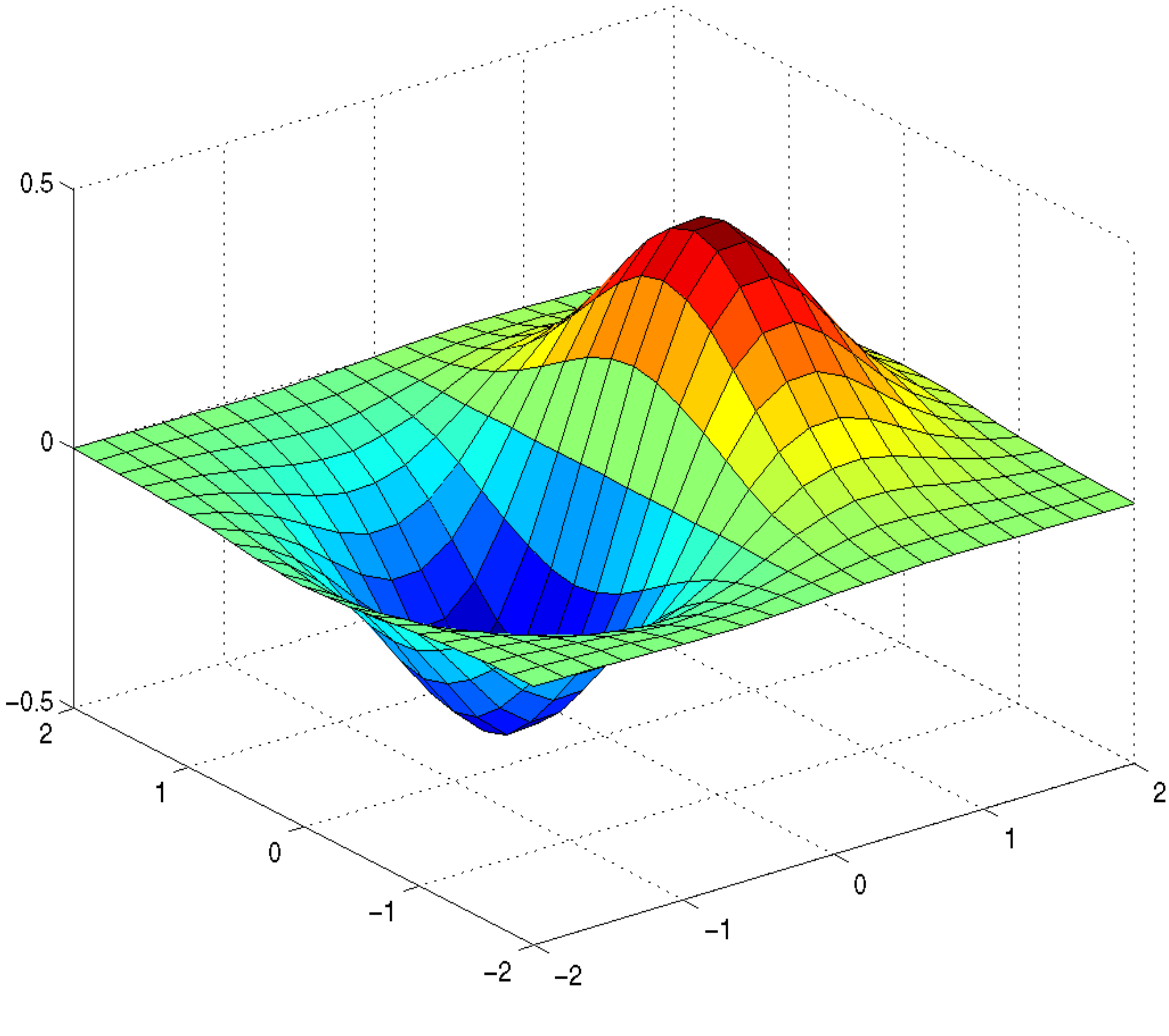
APOSTILA

MATLAB BÁSICO



Universidade Federal do Espírito Santo
CENTRO TECNOLÓGICO
DEPARTAMENTO DE ENGENHARIA MECÂNICA
PET MECÂNICA

APOSTILA
MATLAB BÁSICO

VITÓRIA

2011

Sumário

1. Introdução

MATLAB (que abrevia **MAT**riz **LAB**oratory – Laboratório de Matrizes) é um programa de computador especializado e otimizado para cálculos científicos e de engenharia. Inicialmente era projetado para cálculos com matrizes; ao longo dos anos, transformou-se em um sistema computacional flexível, capaz de resolver essencialmente qualquer problema técnico.

O programa MATLAB implementa a linguagem de programação MATLAB, juntamente com uma grande biblioteca de funções predefinidas que tornam as tarefas de programações técnicas mais fáceis e eficientes. Esta apostila apresenta a linguagem MATLAB e mostra como utilizá-la para resolver problemas técnicos básicos.

MATLAB é um programa muito grande, com uma rica variedade de funções. Até mesmo a versão básica do MATLAB, sem ferramentas adicionais, é muito mais rica que outras linguagens de programação técnica. Existem mais de 1.000 funções no MATLAB, e as ferramentas adicionais ampliam esses recursos com muito mais funções em diferentes especialidades. O objetivo desta apostila não é apresentar todas as funções do MATLAB. Em vez disso, o leitor deve aprender os fundamentos de como escrever, depurar e otimizar bons programas MATLAB, juntamente com um subconjunto das funções mais importantes. Outro aspecto igualmente importante é que o programador aprende a utilizar as ferramentas do próprio MATLAB para localizar a função adequada a um propósito específico a partir da enorme gama de opções disponíveis.

2. Ambiente MATLAB

A unidade fundamental de dados em qualquer programa MATLAB é a **matriz.** Uma matriz é uma coleção de valores de dados organizados em linhas e colunas, determinada por um nome único. Valores individuais de dados em uma matriz podem ser acessados por meio do nome da matriz seguido de índices entre parênteses que identificam a linha e a coluna de um valor particular. Até mesmo escalares são tratados como matrizes em MATLAB – eles são simplesmente matrizes com apenas uma linha e uma coluna. Aprenderemos a criar e a manipular matrizes na seção 4.0.

Existem 3 principais ferramentas presentes na área de trabalho e podem ser acessadas a partir delas que são:

- Command Window
- Command History
- Launch Pad
- Navegador de Ajuda

2.1. Command Window

O lado direito da área de trabalho MATLAB contém o **Command Window.** O usuário pode inserir comandos interativos pelo marcador (>>), no Command Window, e eles serão executados de imediato.

Vamos a um exemplo de cálculo interativo bem simples. Suponha que você queira calcular a área de um círculo com raio de 2,5 m. Isso pode ser feito no Command Window, digitando:

```
» area = pi*2.5^2

area =

19.6350
```

O MATLAB calcula a resposta assim que a tecla enter é pressionada, e armazena o resultado em uma variável (na realidade, em uma matriz 1x1) denomina área. O conteúdo dessa variável será exibido na Command Window e a variável pode ser usada em outros cálculos.

Se uma declaração é muito extensa para ser digitada em uma única linha, ela pode ser complementada em linhas sucessivas digitando **reticências** (...) no final de cada linha, e então continuando na linha seguinte. Por exemplo, as duas expressões a seguir são idênticas:

```
x1 = 1 + 1/2 + 1/3 + 1/4 + 1/5 + 1/6;
```

e

```
x1 = 1 + 1/2 + 1/3 + 1/4 …
          + 1/5 + 1/6;
```

Existem alguns comandos especiais que são muito executados na Command Window como veremos no quadro a seguir:

Command Window	
clc	Limpa o conteúdo do Command window.
diary	Grava o conteúdo do Command window para um ficheiro de texto chamado diary.
diary nome_ficheiro	Grava tudo o que se passa durante uma sessão (exceto gráficos) para o ficheiro escolhido. Se depois escrevermos diary off o MATLAB interrompe a gravação do que se passa. Se introduzirmos o comando diary on o MATLAB retoma a gravação.
Home	Move o cursor para o canto superior esquerdo.
more	Obriga os dados a saírem para o ecrã página a página. (more on e more off) O comando more(n) obriga à saída de n linhas por ecrã.

2.2. Command History

A Janela de Histórico de Comandos (Command History) exibe uma lista dos comandos que o usuário inseriu na Janela de Comandos (Command Window). A lista de comandos anteriores pode se estender a execuções anteriores do programa. Os comandos permanecem na lista até serem apagados. Para reexecutar qualquer comando, simplesmente clique duas vezes sobre ele com o botão esquerdo do mouse. Para apagar um ou mais comandos da Janela de Histórico de Comandos, selecione os comandos e clique sobre eles com o botão direito do mouse. Surgirá um menu, que permitirá ao usuário apagar os intens.

2.3. Launch Pad

O Espaço de Lançamento (Launch Pad) é uma ferramenta especial que agrupa referências a documentação, demonstrações e ferramentas relacionadas para o próprio MATLAB e para cada conjunto de ferramentas que você adquire. A informação é organizada por produto, com todas as referências listadas abaixo de cada produto ou conjunto de ferramentas. Pessoas diferentes irão adquirir produtos diferentes, portanto, o conteúdo exato dessa janela varia de instalação para instalação.

2.4. Obtendo Ajuda

Existem algumas formas de se obter Ajuda no MATLAB, mas aqui iremos abordar duas formas de ajuda baseadas em linhas de comando.

A primeira forma é digitar help ou help seguido de um nome de função na Janela de Comandos. Se você digitar somente help, o MATLAB exibirá uma lista de possíveis tópicos de ajuda na Janela de Comandos. Se uma função específica ou nome de conjunto de ferramentas for acrescentado, será apresentada ajuda para aquela função ou conjunto de ferramentas específico.

A segunda maneira de obter ajuda é o comando lookfor. Ele difere do comando help, pois este procura por um nome de função que case perfeitamente com o fornecido, enquanto o comando lookfor pesquisa informações resumida e rápida para cada função. Assim o comando lookfor é mais lento que o help, mas aumenta a chance de fornecer informação útil. Por exemplo, suponha que você esteja procurando uma função para inverter uma matriz. O MATLAB não tem uma função chamada inverse, assim, o comando help inverse não encontrará nada. Por outro lado, o comando lookfor inverse apresentará os seguintes resultados:

```
>> lookfor inverse

invhilb           - Inverse Hilbert matrix.

ipermute          - Inverse permute array dimensions.

acos              - Inverse cosine, result in radians.
```

acosd	- Inverse cosine, result in degrees.
acosh	- Inverse hyperbolic cosine.
acot	- Inverse cotangent, result in radian.
acotd	- Inverse cotangent, result in degrees.
acoth	- Inverse hyperbolic cotangent.
acsc	- Inverse cosecant, result in radian.
acscd	- Inverse cosecant, result in degrees.
acsch	- Inverse hyperbolic cosecant.
asec	- Inverse secant, result in radians.
asecd	- Inverse secant, result in degrees.
asech	- Inverse hyperbolic secant.
asin	- Inverse sine, result in radians.
asind	- Inverse sine, result in degrees.
asinh	- Inverse hyperbolic sine.
atan	- Inverse tangent, result in radians.
atan2	- Four quadrant inverse tangent.
atand	- Inverse tangent, result in degrees.
atanh	- Inverse hyperbolic tangent.
betaincinv	- Inverse incomplete beta function.

3. Operações Básicas e Variáveis em MATLAB

3.1. Operações Aritméticas

O MATLAB possui todas as operações básicas da matemática (adição, subtração, multiplicação, divisão, exponenciação) podendo ser utilizado como uma simples máquina de calcular. Aqui estão alguns exemplos:

Tabela 1.1 – Operações aritméticas entre dois escalares		
Operação	Forma Algébrica	MATLAB
Adição	a + b	a + b
Subtração	a - b	a – b
Multiplicação	a x b	a*b
Divisão Direita	a÷b	a/b
Divisão Esquerda	b÷a	a\b
Exponenciação	ab	a^b

Hierarquia em Operações Aritméticas

Sabendo que várias operações podem ser combinadas em uma simples expressão aritmética, é importante conhecer a ordem nas quais as operações serão executadas. A tabela 1.2 contém a ordem de prioridade das operações aritméticas no MATLAB.

Tabela 1.2– Hierarquia em operações aritméticas	
Prioridade	Operação
1	Parênteses
2	Exponenciação, esquerda à direita
3	Multiplicação e Divisão, esquerda à direita
4	Adição e Subtração, esquerda à direita

Exemplos:

```
» 56/8
ans = 7

» 8\56   % aqui a divisão é da direita para esquerda!
ans =
        7
```

```
» 4^2*250 + 2*100

ans =

        4200

»4^2*(250 + 2)*100

ans =

       403200
```

3.2. Variáveis em MATLAB

A variável pode ser definida como uma letra ou um conjunto de caracteres, havendo o caso sensitivo, isto é, uma variável em letra minúscula é diferente daquela mesma em letra maiúscula. Se for usar mais de uma palavra para representar uma variável, deve ser usado o sinal de sublinhado para ligar os nomes que representarão a variável. As variáveis devem ter no máximo 19 caracteres, caso passe, o MATLAB ignora os caracteres restantes.

Considerando a seção anterior vamos mostrar que podemos efetuar os mesmos cálculos utilizando variáveis:

```
» parafusos=12
parafusos =
     12

» porcas=8
porcas =
     8

» itens= parafusos +porcas
itens =
     20

» custo_total= parafusos *5+porcas*4
custo_total =
     92
```

Tabela 1.3–Variáveis Especiais

Ans	Variável onde são guardados, por defeito, os resultados das operações - **ans** é o diminutivo de Answer.
PI	Valor de π = 3.1416.
Eps	*Unidade de arredondamento da máquina*, i.e., o menor valor que adicionado a 1 representa um número maior que 1
Flops	Contador do número de operações efetuadas. Estamos a falar de operações em vírgula flutuante.
Inf	Representa $+\infty$, isto é, 1/0
NaN	**N**ot-**a**-**N**umber, símbolo que representa 0/0 ou outra expressão não determinada.
i,j	Estas variáveis são inicialmente agrupadas ao valor $\sqrt{-1}$.
clock	Exibe a hora atual em um vetor linha de seis elementos contendo ano, mês, dia, hora, minute e segundos.

MATLAB *WORKSPACE*

O MATLAB recorda-se de todos os comandos que vão sendo introduzidos ao longo de uma sessão, permitindo que os utilizadores repitam ou aproveitem comandos inseridos noutras alturas. De igual modo, todas as variáveis que vão sendo definidas ao longo da sessão ficam disponíveis para serem utilizadas em ocasiões futuras.
O "local" onde esta informação está guardada designa-se por *MATLAB workspace*.

Em seguida enumeram-se algumas das coisas que podemos fazer, relacionadas com o
workspace:

- Se quisermos obter uma lista com as variáveis presentes no *workspace* basta utilizar o comando **who**. Também podemos utilizar o comando **whos** que juntamente com os nomes das variáveis refere, também, qual a memória que cada uma ocupa assim como a sua dimensão - o que é muito útil se as variáveis forem matrizes.

```
>> who

Your variables are:

ans         custo_total  itens       parafusos    porcas
```

- Também é possível gravar o conteúdo do *Workspace* para um ficheiro. Para isso podemos utilizar o menu **[File]** **[Save Workspace As…]** ou utilizar os comandos **save** e **load**. As variáveis presentes no *Workspace* podem ser gravadas em formato binário ou formato ascii.
  Se utilizarmos apenas o comando **save** sem especificar qual o nome do ficheiro em que pretendemos guardar a informação, ela será gravado no ficheiro **matlab.mat**. (Para obter uma explicação mais completa deste comando escreva **help save**)

```
>> save

Saving to: matlab.mat

>> save meu
% grava as variáveis em format binário para o ficheiro meu.mat

>> save dados parafusos porcas custo_total -ascii
% as variáveis foram gravadas em formato ascii
% podemos escolher quais as variáveis que queremos gravar
```

- Podemos remover alguma ou todas as variáveis presentes no *Workspace* utilizando o comando **clear**.

```
>> who

Your variables are:

ans         custo_total  itens       parafusos    porcas

>> clear parafusos
>> who

Your variables are:

ans         custo_total  itens       porcas

>> % se escrevermos o comando clear sem nenhum argumento o MATLAB
apaga todas as variáveis
```

3.3. Formatos Numéricos

No MATLAB podemos apresentar os números segundo diversos formatos. Aqui estão alguns exemplos:

```
» preco=1/3

preco =

    0.3333

» format long preco % Utiliza 16 dígitos

» preco

preco =

    0.33333333333333

» format short e % Utiliza 5 dígitos mais expoente

» preco

preco =

    3.3333e-001

» format long e % Utiliza 16 dígitos mais expoente

» preco

preco =

3.333333333333333e-001

» format hex % O número é escrito em formato hexadecimal

» preco

preco =

3fd5555555555555

» format bank % Escreve o número utilizando duas casas decimais

» preco

preco =

0.33

» format rat % Utilizando frações

» preco

preco =

1/3
```

» format short % Utiliza 5 dígitos

» preco

preco =

0.3333

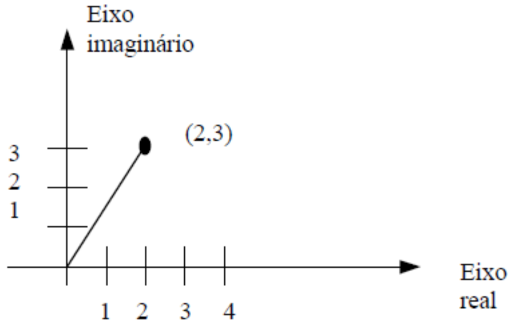
Nota: A representação interna dos números não é alterada quando se utilizam estes comandos.

3.4. Números Complexos

No MATLAB a definição de números complexos faz-se de uma maneira natural, apesar disso, eles podem ser definidos utilizando vários métodos:

Coordenadas polar e retangular

Podemos representar um número complexo em um plano com eixos real e imaginário. Os números reais representam o eixo x, e os números imaginários representam o eixo y, e os números com partes real e imaginária representam o resto do plano.
Quando representamos um número complexo com uma parte real e imaginária, como 2+i3, estamos usando uma notação retangular. A figura 1.1 mostra que o número complexo pode ser escrito com um ângulo θ e raio r em relação à origem. Esta forma é chamada de notação polar, e o ponto 2 + i3 pode ser representado em notação polar com um ângulo de 0,98 radianos e um raio 3,6.



Conversão

- retangular a polar

$r = \sqrt{(a^2 + b^2)}$
$\theta = \mathrm{arctg}(b/a)$

- polar a retangular

$a = r \cos \theta$
$b = r \text{ sen } \theta =$

• Definição de um complexo utilizando o i para identificar a parte imaginária.

```
» c1=1-2i

c1 =

   1.0000 - 2.0000i
```

• Definição de um complexo utilizando o j para identificar a parte imaginária.

```
» c1=1-2j   % o j também serve

c1 =

   1.0000 - 2.0000i
```

• Definição de um complexo utilizando o sqrt(-1) para identificar a parte imaginária.

```
» c2=3*(2-sqrt(-1)*3)

c2 =

   6.0000 - 9.0000i
```

• Sempre que aparecem raízes de números negativos então o MATLAB considera esse valor como um complexo.

```
» c3 = sqrt(-2)

c3 =

       0 + 1.4142i
```

• Definição de um complexo em função de outro complexo.

```
» c4=6+sin(.5)*i % neste caso foi necessário por sin(.5)*I

c4 =

   6.0000 + 0.4794i
```

As operações aritméticas entre complexos são escritas de forma semelhante ao que se fazia para os reais.

```
» c6=c1+c2

c6 =

  7.0000 -11.0000i

» c7=(c1+c2)/c3

c7 =

  -7.7782 -4.9497i
```

Para o MATLAB o resultado de uma operação entre números complexos é um complexo.

```
» c8=i^2   % o quadrado de i é o real -1

c8 =

 -1.0000 + 0.0000i
```

Apesar de i^2= -1 ser um real o **MATLAB** mantêm a parte imaginária do número igual a zero. Para eliminar a parte imaginária de um número complexo utiliza-se a função real.

```
» c9 =  real(c6)

c9 =

    7
```

Apresentam-se, agora, as funções utilizadas para estabelecer a correspondência entre a representação algébrica ($z = a+ bi$) e a representação polar ($z = r (\cos \theta + \text{sen } \theta)$, em que $r = |z|$):

• A função **abs** determina o valor absoluto de um complexo.

```
» c1

c1 =

  1.0000 - 2.0000i

» mag_c1 = abs(c1)

mag_c1 =

   2.2361
```

• A função **angle** determina o argumento de um complexo em radianos.

```
» angle_c1=angle(c1)

angle_c1 =

  -1.1071

» deg_c1=angle_c1*180/pi

deg_c1 =

 -63.4349
```

Outras duas funções utilizadas com números complexos são:

• A função **conj** dá-nos o complexo conjugado de um número complexo.

```
» conj(c1)

ans =

  1.0000 + 2.0000i
```

• A função **imag** dá-nos a parte imaginária de um complexo.

```
» imag_c1=imag(c1)

imag_c1 =

   -2
```

• A função **real** dá-nos a parte real de um imaginário.

```
» real_c1=real(c1)

real_c1 =

    1
```

3.5. Funções Matemáticas

Em seguida apresenta-se um quadro com as principais funções matemáticas que o MATLAB possui.

TRIGONOMETRICAS	
`sin`	Seno
`sinh`	Seno hiperbólico
`asin`	Arco cujo o seno é
`asinh`	Arco cujo seno hiperbólico é
`cos`	Co-seno
`cosh`	Co-seno hiperbólico
`acos`	Arco cujo o co-seno é
`acosh`	Arco cujo co-seno hiperbólico é
`tan`	Tangente
`tanh`	Tangente hiperbólica
`atan`	Arco cuja tangente é
`atanh`	Arco cuja tangente hiperbólica é
`sec`	Secante
`sech`	Secante hiperbólica
`asec`	Arco cujo co-seno hiperbólico é
`asech`	Arco cujo co-seno hiperbólico é
`csc`	Co-secante
`csch`	Co-secante hiperbólica
`acsc`	Arco cuja co-secante é
`acsch`	Arco cuja co-secante hiperbólica é
`cot`	Co-tangente
`coth`	Co-tangente hiperbólica
`acot`	Arco cuja co-tangente é
`acoth`	Arco cuja co-tangente hiperbólica é

EXPONENTIAL	
`exp`	Exponencial
`log`	Logaritmo natural
`log10`	Logaritmo de base 10
`sqrt`	Raiz quadrada

COMPLEXAS	
`abs`	Valor Absoluto
`angle`	Argumento (em radianos)
`conj`	Complexo conjugado
`imag`	Parte imaginaria
`real`	Parte real

NUMÉRICAS	
`round`	Arredonda para o inteiro mais próximo
`rem`	Resto da divisão
`sign`	Sinal de um número

Alguns exemplos de aplicação dessas funções matemáticas são apresentados em seguida:

```
» x=sqrt(2)/2
x =
0.7071

» y=asin(x)
y =
0.7854

» y_deg=y*180/pi
y_deg =
45.0000

» z=rem(23,4)
z =
3

» z1=23/4
z1 =
5.7500
```

» a=exp(c1)
a =
-1.1312 - 2.4717i

» sign(1.2)
ans =
1 % a resposta é 1 pois o número é positivo

» sign(-23.4)
ans =
-1 % a resposta é –1 quando o número é negativo

» sign(0)
ans =
0

4. Matrizes e Vetores

4.1. Vetores

Definição de vetor

Quando se pretende introduzir um vetor deve fazer-se:

```
>> a=[1 2 3 4 5 6]

a =

    1     2     3     4     5     6
```

Também é possível definir um novo vetor à custa de outro já existente:

```
>> x= [0 .1*pi .2*pi .3*pi .4*pi .5*pi .6*pi .7*pi .8*pi .9*pi pi]

x =

  Columns 1 through 9

         0    0.3142    0.6283    0.9425    1.2566    1.5708    1.8850    2.1991    2.5133

  Columns 10 through 11

    2.8274    3.1416

>> y=sin(x)
```

```
y =

  Columns 1 through 9

         0    0.3090    0.5878    0.8090    0.9511    1.0000    0.9511    0.8090    0.5878

  Columns 10 through 11

    0.3090    0.0000
```

Na definição de vetores um símbolo muito utilizado é o " **:** ". Aqui estão alguns exemplos da sua utilização.

```
x=1:5 % começa em 1 e termina em 5 com incrementos de 1

x =

     1     2     3     4     5
```

Se pretendermos utilizar um incremento diferente fazemos:

```
>> y=0:0.1:1 % começa em 0 e termina em 1 com incrementos de 0.1

y =

  Columns 1 through 9

         0    0.1000    0.2000    0.3000    0.4000    0.5000    0.6000    0.7000    0.8000

  Columns 10 through 11

    0.9000    1.0000
```

Também é possível utilizar incrementos negativos:

```
z=6:-1:1

z =

     6     5     4     3     2     1
```

Existem duas funções que podemos utilizar para criar vectores.

```
>> l=linspace(0,pi,11)

l =

  Columns 1 through 9

         0    0.3142    0.6283    0.9425    1.2566    1.5708    1.8850    2.1991    2.5133

  Columns 10 through 11
```

```
   2.8274    3.1416

>> g=logspace(0,2,11)

g =

  Columns 1 through 9

    1.0000    1.5849    2.5119    3.9811    6.3096   10.0000   15.8489   25.1189   39.8107

  Columns 10 through 11

   63.0957  100.0000
```

Tabela 2.1 -Métodos para Definir Vectores	
x=[2 2*pi sqrt(2) 2-3j]	Cria o vetor **x** com os elementos especificados. (Os elementos podem ser expressões ou números complexos).
x=ni:nf	Cria um vetor começando no número **ni** e terminando em **nf**, com incrementos de 1.
x=ni:i:nf	Cria um vetor começando no número **ni** e terminando em **nf**, com incrementos de **i**.
x=linspace(primeiro,ultimo,n)	Cria um vetor começando no **primeiro** e terminando no **último**, com n elementos.
x=logspace(primeiro,ultimo,n)	Cria um vetor começando em 10^**primeiro** e terminando em 10^**último**, com n elementos.

Endereçamento dos Elementos de um Vetor

Podemos também mudar e adicionar valores no vetor S usando uma referência entre parênteses. Assim, o seguinte comando:

```
S = [ 3.0 1.5 3.1];

S (2) = -1.0;
```

Muda o segundo valor do vetor S de 1.5 para –1.0.

A ordem do vetor pode ser alterada. Se executarmos o seguinte comando:

```
S(4) = 5.5;
```

Então a matriz S terá quatro valores em vez de três. Se executarmos o comando:

```
S(8) = 9.5;
```

Então a matriz S terá 8 elementos, e os valores de S(5), S(6) e S(7) são automaticamente nulos, já que não foram atribuídos valores para eles.

O ponto-e-vírgula no final de cada declaração de atribuição anterior tem uma função especial: ele *suprime o eco automático de valores* que ocorre normalmente quando uma expressão é avaliada em uma declaração de atribuição. Se uma declaração e atribuição é digitada sem o ponto-e-vírgula, os resultados da declaração são automaticamente exibidos na Janela de Comandos.

```
>> a=[1 2 3 4 5 6]

a =

   1   2   3   4   5   6
```

Se um ponto-e-vírgula é colocado no final da declaração, o eco desaparece. O eco é uma forma excelente para verificar rapidamente seu trabalho, mas ele atrasa seriamente a execução de programas MATLAB. Por essa razão, normalmente suprimimos o eco o tempo todo.

4.2. Matrizes

Suponha que queiramos agora criar as matrizes A, B e C usando o MATLAB. Há vários métodos de definição de matrizes no MATLAB. Vejamos cada um:

Modo mais simples:

Nome da matriz = [a_{11} a_{12} a_{13} …a_{1n} ; a_{21} a_{22} a_{23} … a_{2n} ; … ; a_{m1} a_{m2} a_{m3} … a_{mn}];

Assim, se quisermos as matrizes A, B e C abaixo:

A = [3.5]   B = [1.5 3.1]   C = $\begin{bmatrix} 0 & 0 & 2 \\ 1 & 1 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}$.

Representaremos por:

```
A = [ 3.5];
B = [1.5, 3.1];
C = [0,0,2; 1,1,0; 1,1,0;1,0,0];
```

O nome da matriz deve começar com uma letra e conter no máximo 19 caracteres que podem ser números, letras ou caractere sublinhado, e aparece ao lado esquerdo do sinal de igual. O lado direito contém os dados entre colchetes por ordem de linhas. O ponto-e-vírgula separa as linhas, e os valores das linhas podem estar separados por vírgulas ou por espaços. O valor pode conter um sinal de + ou -, e um ponto decimal, mas não pode conter uma vírgula, como 32,154.

Se quisermos, por exemplo, definir um vetor-linha F com 10 valores, também podemos fazer:

```
F = [1 52 64 197 42 –42 55 82 22 109]

F = [1 52 64 197 42 –42, …55 82 22 109]
```

Esta forma é muito usada quando a linha de uma matriz é extensa. Podemos terminar uma linha com uma vírgula seguida de três ou mais pontos, e continuar a entrar com os valores restantes na próxima linha da área de trabalho do MATLAB.

Podemos também definir uma matriz usando outra que já definida. Por exemplo, considere as seguintes matrizes:

```
B = [1.5, 3.1];
S = [3.0 B];
```

Estes comandos equivalem a:

```
S = [3.0 1.5 3.1];
```

4.3. Operações com Matrizes

Matrizes Transpostas

A transposta de uma matriz é uma nova matriz onde as colunas são formadas pelas linhas da matriz original.

Exemplo 1

```
>> A=[1 2 3;4 5 6; 7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9

>> T=A'

T =

     1     4     7
     2     5     8
     3     6     9
```

Para vetores tudo se passa de modo semelhante,

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

x =

    -1
     0
     2
```

Adição e Subtração

Considerando os vetores A e B vamos determinar A+B.

```
>> A=[1 2 3;4 5 6; 7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9

>> B=[1 4 7; 2 5 8; 3 6 0]

B =

     1     4     7
     2     5     8
     3     6     0


>> A+B
```

```
ans =

   2    6    10
   6    10   14
   10   14    9
```

Multiplicação

A multiplicação de duas matrizes corresponde ao somatório de produtos das linhas i da primeira matriz e das colunas j da Segunda matriz. Como o somatório de produtos requer que os vetores tenham o mesmo número de elementos, então o número de colunas de A deve ser igual ao número de linhas de B.

```
>> C=A*B

C =

   14   32   23
   32   77   68
   50  122  113
```

O primeiro elemento do produto C = A.B é:
(1).(1) + (2).(2) + (3).(3) = 14

Divisão

Existem dois tipos de divisão: a divisão à esquerda (A\b) e a divisão à direita (A/b). Podemos ver qual a diferença entre estes dois tipos de divisão através de um exemplo.
Considere o seguinte sistema de equações:

$$\underset{A}{\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 0 \end{bmatrix}} \cdot \underset{x}{\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}} \underset{=}{=} \underset{b}{\begin{bmatrix} 366 \\ 351 \\ 804 \end{bmatrix}}$$

```
>> A=[1 2 3; 4 5 6; 7 8 0]

A =

    1    2    3
    4    5    6
    7    8    0
```

```
>> b=[366;804;351]

b =

  366
  804
  351
```

Divisão à esquerda (x = A\b é solução para A * x = b), ou seja, a divisão indireta entre duas matrizes (A\b) é equivalente a multiplicar a matriz B pela inversa de A.

```
>> x=A\b

x =

  25.0000
  22.0000
  99.0000
```

Outro exemplo:

```
>>A = [ 9 4 0 ; 3 6 4 ; 0 9 2 ] ;
>> B = [ 8 3 1 ; 10 8 3 ; 4 9 8 ] ;
>> B\A
ans =

  1.92000    0.44667   -0.38667
 -2.88000   -0.25333    1.41333
  2.28000    1.18667   -1.14667
```

Divisão à direita (y = A/b é solução para x * A = b), ou seja, a divisão direta (A/B) entre duas matrizes é equivalente a multiplicar a matriz A pela inversa de B.

```
>> y=A/b
??? Error using ==> mrdivide
Matrix dimensions must agree.

>> x*A
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

Devido as dimensões das matrizes não é possível realizar a divisão à direita teria que se modificar as dimensões de b, fazendo a sua transformada por exemplo.
A relação entre estas duas divisões é dada por: A\b = (b'/A')'

```
>> (b'/A')'

ans =
  25.0000
   22.0000
   99.0000
```

Outro exemplo:

```
>>A =  [ 9 4 0 ;  3 6 4  ; 0 9 2 ] ;
>> B =  [ 8 3 1 ; 10 8 3 ; 4 9 8 ] ;
>> B/A

ans =

   0.7625    0.3792    -0.2583
   0.8625    0.7458     0.0083
  -0.2750    2.1583    -0.3167
```

4.4. *Operações com Arrays*

Quando se utiliza o termo “operações com *arrays*” pretende-se referir que as operações aritméticas são feitas de elemento para elemento.

Adição e Subtração

Para a adição e subtração as operações com *array* e as operações com matrizes são iguais. (Utiliza-se os mesmos símbolos)

Multiplicação e Divisão

A multiplicação de *array* é efetuada elemento a elemento, sendo representada por “ .* ”. Se A e B têm dimensões iguais então C = A .* B resulta numa matriz em que cada um dos seus elementos é igual ao produto dos elementos individuais de A e B, nas mesmas posições.

```
>> A=[3 1 5]

A =

     3     1     5

>> B=[4 1 3]

B =

     4     1     3
```

```
>> C=A.*B

C =

   12     1    15
```

No que diz respeito à divisão; se tivermos D = A ./ B ou E = A .\ B cada elemento de D e E é obtidos através da divisão (à esquerda ou à direita) envolvendo os elementos respectivos de A e B.

```
>> D=A./B

D =

   0.7500    1.0000    1.6667

>> E=A.\B

E =

   1.3333    1.0000    0.6000
```

Potenciação

A potenciação elemento a elemento é efetuada utilizando os símbolos “ .^ ”.

```
>> P=A.^B

P =

   81     1   125
```

Contudo, o expoente poderá ser um escalar.

```
>> Pe=A.^3

Pe =

   27     1   125
```

Mas também podemos ter a base como um escalar.

```
>> Ps=3.^A

Ps =

   27     3   243
```

Operações com Arrays	
Sabendo que : **A = [a_1 a_2 … a_n]; B = [b_1 b_2 … b_n]; c – Escalar**	
Adição com um escalar	**A+c = [a1+c a2+c … an+c]**
Multiplicação com um escalar	**A*c = [a1*c a2*c … an*c]**
Adição	**A+B = [a1+b1 a2+b2 … an+bn]**
Multiplicação	**A.*B = [a1.*b1 a2.*b2 … an.*bn]**
Divisão à esquerda	**A.\B = [a1.\b1 a2.\b2 … an.\bn]**
Divisão à direita	**A./B = [a1./b1 a2./b2 … an./bn]**
Potenciação	**A.^c = [a1.^c a2.^c … an.^c]** **c.^A = [c.^a1 c.^a2.^c … c.^an]** **A.^B = [a1.^b1 a2.^b2 … an.^bn]**

4.5. Manipulações dos Elementos de uma Matriz

Podemos também mudar e adicionar valores na matriz usando uma referência entre parênteses. Assim, o seguinte comando para uma matriz A é:

```
>> A = [3 5 10 ; 0  0 3; 3 9  5]

A =
 3    5    10
 0    0     3
 3    9     5
```

>> A(3,3)=9 % modifica o elemento na 3ª linha, 3ª coluna para 9

```
A =

   3    5    10
```

```
     0     0     3
     3     9     9
```

Também podemos fazer:

```
>> A(2,2)=A(1,2)+A(3,2)

A =

     3     5    10
     0    14     3
     3     9     5
```

Outra manipulação dos elementos de uma matriz que o MATLAB permite é a seguinte:

b=A(:) % transformamos uma matriz num vetor coluna

```
b =

     3
     0
     3
     5
    14
     9
    10
     3
     5
```

Se quisermos criar uma matriz B invertendo a ordem das linhas de A, fazemos:

>> B=A(3:-1:1,1:3) % escolhemos as linhas, começando na 3 e acabando na
    % escolhemos as coluna, começando na 1 e acabando na 3

```
B =

     3     9     5
     0    14     3
     3     5    10
```

De modo semelhante podemos obter uma submatriz de A.

```
>> C=A(1:2,2:3)

C =
     5    10
    14     3
```

Manipulação dos Elementos de uma Matriz	
A(l,c)	Resulta uma submatriz de **A** com as linhas definas pelo vetor **l** e com as colunas definas (ou indexadas) pelo Vetor **c**.
A(l,:)	Obtemos uma submatriz de **A** com as linhas definas pelo vetor **l** e com todas as colunas de **A**.
A(:,c)	Resulta uma submatriz de **A** com todas as linhas de **A** e com as colunas definas (ou indexadas) pelo vector **c**.
A(:,c)	Obtemos um vetor coluna com todos os elementos de **A** tendo estes sido retirados coluna a coluna da matriz **A**.

4.6. Matrizes Especiais e Funções com Matrizes

De seguida apresentam-se algumas das matrizes especiais que é possível criar utilizando o MATLAB.

Matrizes Especiais

Magic Square

Uma matriz *magic square* de ordem n é uma matriz n x n constituída de números inteiros de 1 a n^2. Os elementos a_{ij} da matriz estão dispostos de forma tal que o somatório de cada linha é igual ao somatório de uma coluna.

Forma Geral: *magic (n)* matriz *square magic* de ordem n.

Assim, para saber o quadrado mágico de ordem 3 , o prompt do MATLAB deve apresentar:

magic (3)

Zeros

Esta função gera uma *matriz zero*, isto é, uma matriz cujos elementos a_{ij} são nulos.
Forma Geral: zeros(n) Gera uma matriz zero, quadrada, de ordem n.

zeros(m,n) Gera uma matriz zero de ordem m x n.

Ones

A função ones gera uma matriz cujo valor dos elementos a_{ij} é unitário.

Argumento: ones(n) Gera uma matriz quadrada de ordem n.

ones(m,n) Gera uma matriz de ordem m x n.

Eye

A matriz identidade pode ser gerada pelo MATLAB através da função *eye.* Uma matriz identidade é uma matriz escalar de qualquer ordem cujos elementos a_{ij} são iguais a 1 para i = j.
Apresenta o mesmo formato que as funções anteriores. O formato "eye(n)" gera uma matriz identidade de ordem n. Já o formato "eye (m,n)" gera uma matriz de ordem m x n .

Pascal

Cria uma matriz cujas diagonais lembram o triângulo de Pascal. Assim, se usarmos o comando pascal(5), a seguinte matriz é gerada:

1	1	1	1	1
1	2	3	4	5
1	3	6	10	15
1	4	10	20	35
1	5	15	35	70

Funções com Matrizes

O MATLAB possui inúmeras funções com matrizes. No quadro seguinte vamos apresentar apenas algumas.

Funções com Matrizes	
det(A)	Determinante
expm(A)	Matriz exponencial
logm(A)	Matriz logaritmo
inv(A)	Inversa da matriz **A**
d=eig(A) **[V,D]=eig(A)**	Valores próprio e vetores próprios.
poly(A)	Polinómio característico

5. Arquivos M – File

5.1. Primeiros Programas

Quando o número de comandos a serem introduzidos é muito grande e também quando queremos reavaliar as expressões entretanto introduzidas torna-se mais prático utilizar ficheiros de texto com comandos de MATLAB denominados *Script files* (ou *m-files*).
Para criarmos um novo *script* (ou *m-file*) basta procurar o comando **[New]** localizado no menu **[File]** e seguidamente escolher **[M-file]**.
Como um *m-file* é um ficheiro de texto então pode ser feito em qualquer editor de texto – o ficheiro tem de ter a extensão *.m*.
Para executar um *m-file* basta introduzir o seu nome, por exemplo:

```
» exemplo
```

● O MATLAB procura o ficheiro *exemplo.m* e executa todos os comandos como se eles fossem inseridos directamente no *command window*.
Ao utilizar *m-files* tenha em mente que:

Os comandos presentes no *m-file* têm acesso às variáveis anteriormente definidas no *workspace*.

 As variáveis definidas no *m-file* passam a fazer parte do *workspace* e podem ser utilizadas após a execução do *m-file*.

 O comando **echo on** diz ao MATLAB para fazer o eco dos comandos que vai lendo e executando. O comando **echo off** faz o contrário.

Exemplo de um *m-file*:

```
%Exemplo1 – m-files

cadernos=4;

canetas=input(' Introduza o nº de canetas > ');

itens=cadernos+canetas

custo_total=cadernos*250+canetas*100
```

A execução deste *m-file* produz os seguintes resultados:

```
» exemplo1

Introduza o nº de canetas > 4

itens =

8
custo_total =

1400
```

FUNÇÕES PARA OS *M-FILES*

disp(*variável*)	Mostra o valor de uma variável sem apresentar o seu nome.
echo	Controla o eco dos comandos, presentes no *m-file*, que vão sendo executados. (**echo on** e **echo off**)
input	Espera pela introdução de um valor pelo utilizador.
keyboard	Interrompe a execução de um *m-file* dando liberdade ao utilizador para executar outros comandos. Retoma-se a execução do *m-file* fazendo **return**..
disp	faz com que o texto apareça na tela do MATLAB
pause(n)	Há uma pausa de n segundos na execução.
waitforbuttonpress	Existe uma pausa na execução do *m-file* até que se carregue numa tecla do mouse ou do teclado.

Muitas vezes precisamos fazer um programa interativo de forma que as entradas possam ser dadas via teclado. Para isso, usamos os comandos **input** e **disp**.

6. Polinômios

Representação

No Matlab, um polinômio é representado por um vetor-linha contendo os coeficientes do polinômio em ordem decrescente. Por exemplo, o polinômio de 4º grau: $x^4 - 3x^3 + 10x^2 - 7x - 48$ é representado pelo vetor:

```
p = [1 –3 10 –7 –48]
```

Raízes

Para calcular as raízes de um polinômio utiliza-se o comando **roots**:

```
>> r = roots(p)

r =

  0.8995 + 3.3273i

  0.8995 - 3.3273i

  2.6984

 -1.4973
```

que, para o exemplo, apresenta duas raízes reais (2,6984 e –1,2973) e duas raízes complexas (0,8995 + i 3,3273 e 0,8995 – i 3,3273). Se forem conhecidas as raízes de um polinômio, então o comando **poly** retorna os coeficientes desse polinômio:

```
>> poly(r)

 ans =

    1.0000   -3.0000   10.0000   -7.0000  -48.0000
```

Observe que neste exemplo, os valores são os coeficientes do vetor p. Entretanto, se o argumento de poly() for uma matriz quadrada, então este comando retorna os coeficientes do polinômio característico dessa matriz.

Exemplo:

```
>> A = [1 2 3; 4 5 6; 7 8 9];

>> pp = poly(A)

 pp =

    1.0000  -15.0000  -18.0000    0.0000
```

Produto de Polinômios

Se tivermos dois polinômios y1 e y2 e quisermos calcular o produto y1*y2, utiliza-se o comando **conv**:

```
 >>  y1 = [1 1 9]

 y1 =

     1     1     9

 >> y2 = [2 -4 0 1]

 y2 =

     2    -4     0     1

 >> conv(y1,y2)

 ans =

     2    -2    14   -35     1     9
```

O resultado da operação conv(y1,y2) contém os coeficientes do polinômio:

$$(x^2 + x + 9) \cdot (2x^3 - 4x^2 + 1) = 2x^5 - 2x^4 + 14x^3 - 35x^2 + x + 9$$

Divisão de polinômios

A divisão de polinômios é feita com o comando **deconv**:

```
>> z1 = [1 2 1 -3 1]

z1 =

 1    2    1   -3    1

>> z2 = [-1 1 3]

z2 =

 -1    1    3

>> [q,r] = deconv(z1,z2)

q =

 -1   -3   -7

r =

  0    0    0   13   22
```

No caso da divisão do polinômio ($x^4 + 2x^3 + x^2 – 3x + 1$) por (- x2 + x + 3), o resultado é o polinômio $–x^2 - 3x – 7$ com resto igual a 13x + 22. O Matlab retorna dois vetores: o vetor [q], que contém os coeficientes do quociente, e o vetor [r], que contém os coeficientes do resto da divisão, de tal forma que:

z1 = conv(q, z2) + r

A operação de divisão de polinômios também é denominada deconvolução polinomial.

Avaliação de polinômios

Consideremos a função polinomial

$$f(x) = 2x^4 - 5x^3 + 8x^2 - 10x + 40 .$$

Para avaliar numericamente uma função polinomial pode-se fazê-lo de duas formas: se o valor da variável independente x for um escalar basta apenas escrever a equação na forma algébrica que o Matlab avalia a expressão e retorna o valor do polinômio:

```
>> x = 2;
```

```
>> f = 2*x^4 - 5*x^3 + 8*x^2 - 10*x + 40

 f =

    44
```

Se, no entanto, a variável x for um vetor contendo um intervalo de valores, então será necessário utilizar o operador ponto-escalar, como no exemplo:

```
 >> x = 0:0.5:2;

>> f = 2*x.^4 - 5*x.^3 + 8*x.^2 - 10*x + 40

 f =

   40.0000   36.5000   35.0000   36.2500   44.0000
```

O segundo método consiste na utilização do vetor-linha p contendo os coeficientes do polinômio. Para avaliar numericamente o polinômio para um dado valor ou conjunto de valores x, utiliza-se a função **polyval**:

```
 >> x = 0:0.5:2;

>> p = [2 -5 8 -10 40];

>> y = polyval(p,x)

 y =

   40.0000   36.5000   35.0000   36.2500   44.0000
```

Derivação de Polinômio

Para a derivação de um polinômio, o MATLAB tem a função **polyder**.
Ex.:

```
» g=[1 6 20 48 69 72 44];      % x^6+6x^5+20x^4+48x^3+69x^2+72x+44
» h=polyder(g)
h =
6 30 80 144 138 72
```

Resumo das funções vistas neste módulo:

poly: cálculo do polinômio característico e remontagem de um polinômio a partir das raízes

roots: raízes de polinômio

conv: produto de polinômios

deconv: divisão de polinômios

polyder derivação de polinônios

7. Interpolação e Ajuste de Curva

A interpolação é definida como sendo uma forma de estimar os valores de uma função entre aqueles dados por algum conjunto de pontos de dados. A interpolação é uma ferramenta valiosa quando não se pode calcular rapidamente a função nos pontos intermediários desejados. Por exemplo, isto ocorre quando os pontos de dados resultam de medições experimentais ou de procedimentos computacionais demorados.

Nesta seção vamos apresentar dois tipos de interpolação. A interpolação linear, que considera que os valores intermediários caem em uma linha reta entre os pontos definidos. Neste método se torna claro que, à medida que se têm mais pontos de dados e a distância entre eles diminui, a interpolação linear se torna mais precisa. E a interpolação spline, que considera que alguma curva suave se ajusta aos pontos, onde esta suposição é a de que um polinômio de terceira ordem, isto é, um polinômio cúbico seja usado para modelar cada segmento entre pontos consecutivos e que a inclinação de cada polinômio cúbico se ajuste nos pontos de dados.

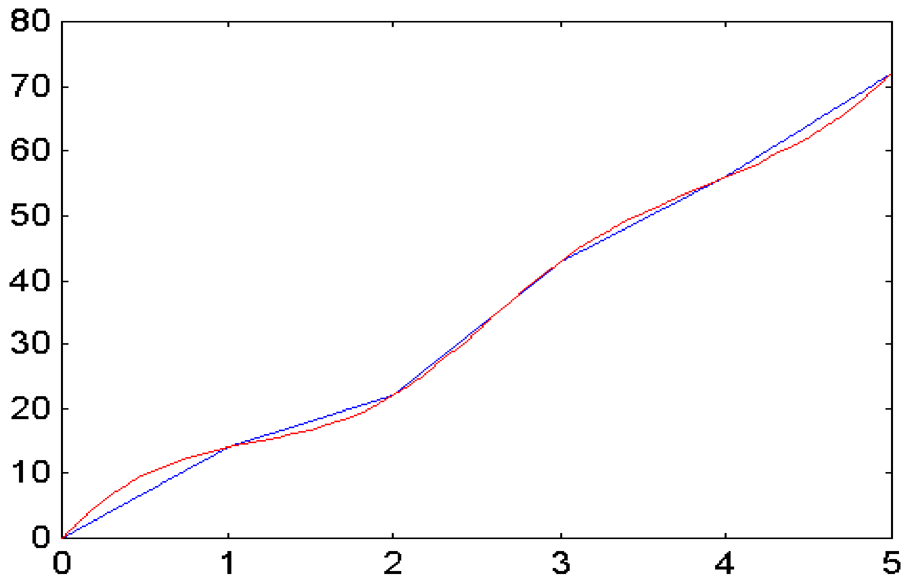


Gráfico 5.1- Interpolação linear

7.1. Interpolação Linear

Supondo que tenhamos apenas duas coordenadas de uma função qualquer e, que podemos estimar seu comportamento linearmente, ou seja através de uma reta entre esses pontos. Então poderemos assim determinar o comportamento da função em qualquer ponto deste intervalo por meio de uma simples semelhança de triângulos, onde a equação geral é:

$$f(b) = f(a) + \frac{b-a}{c-a}\left(f(c) - f(a)\right)$$

A interpolação linear é possível no MATLAB através do uso dos comandos *table1* e *table2*.

Comando table1

Este comando proporciona a interpolação linear em uma dimensão usando para isto uma tabela contendo as informações a serem trabalhadas. O primeiro argumento deste comando se refere à tabela contendo as informações. O segundo se refere ao valor de x para o qual queremos interpolar o valor da função.

O comando irá até a primeira coluna da tabela e achar os dois pontos consecutivos, entre os quais estará o nosso ponto a ser interpolado. O comando então acha o valor da função no ponto escolhido. É importante notar que na hora de alocar os valores na tabela, eles devem estar ordenados crescentemente ou decrescentemente, e o valor a ser interpolado deverá estar entre o primeiro e último valores da primeira coluna da tabela, caso contrário surgirá uma mensagem de erro!

Exemplo 1

Supondo que queiramos determinar o comportamento térmico da cabeça de um cilindro a ser implementado num carro. Supondo também que os valores experimentais referentes ao Tempo e a Temperatura sejam;

Tempo, s	Temp., F
0	0
1	20
2	60
3	68
4	77
5	110

Para alocarmos estas informações devemos usar uma matriz, onde o tempo será preenchido na primeira coluna através dos seguintes comandos:

dado1(:,1) = [0,1,2,3,4,5] ;

dado2(:,2) = [0,20,60,68,77,110] ;

Podemos usar o comando ***table1*** para interpolar a temperatura correspondente a um determinado tempo no intervalo de 0 a 5 segundos:
y1 = table1 (dado1, 2.6);
y2 = table1 (dado1, 4.9);
Os valores correspondentes serão y1 = 64.8 e y2 = 106.7.

Supondo agora que medimos a temperatura em três pontos do cilindro:

Tempo, s	T1	T2	T3
0	0	0	0
1	20	25	52
2	60	62	90
3	68	67	91
4	77	82	93
5	110	103	96

Guardando estas informações numa matriz, com as informações do tempo na primeira coluna:

```
dado2(:,1) = [ 0,1,2,3,4,5] ;

dado2(:,2) = [0,20,60,68,77,110] ;

dado2(:,3) = [0,25,62,67,82,103];

dado2(:,4) = [0,52,90,91,93,96];
```

Para determinar valores das temperaturas nestes três pontos no tempo de t = 2.6 s, usamos os seguinte comando:

```
temps = table1 (dado2, 2.6);
```

Onde temps será um vetor contendo os três valores da temperatura: 64.8, 65.0 e 90.6.

Comando table2

Esse comando possibilita a interpolação bidimensional usando valores da primeira coluna e da primeira linha da tabela. É importante perceber que tanto os elementos da primeira coluna quanto os elementos da primeira linha devem estar ordenados crescentemente ou decrescentemente e que os valores de x e de y devem permanecer entre os limites da tabela.
Supomos agora que iniciamos um determinado processo incrementando uma velocidade constante dada em rotações por minuto, enquanto medimos a temperatura em um ponto da cabeça do cilindro. Então, se iniciarmos o processo e incrementarmos uma velocidade 2000 rpm em 5 segundos e registrarmos os valores de temperatura. Da mesma forma podemos continuar registrando os valores de temperaturas para os vários valores de velocidade:

Tempo, s	V1=2000	V2=3000	V3=4000	V4=5000	V5=6000
0	0	0	0	0	0
1	20	110	176	190	240
2	60	180	220	285	327
3	68	240	349	380	428
4	77	310	450	510	620
5	110	405	503	623	785

Desta forma podemos estimar a temperatura da cabeça do cilindro em qualquer tempo entre 0 e 5 segundos, e em qualquer velocidade entre 2000 e 6000 rpm.

Ao invés de calcularmos, o que seria bem mais complicado pode interpolar a função em questão.
Podemos agora guardar estas informações numa matriz dado3, e então usar o comando ***table2*** para calcular esta informação para nós:
Note que agora nós preenchemos as linhas com as informações da tabela, no exemplo anterior nós preenchemos as colunas.

```
dado3(1,:) = [0,2000,3000,4000,5000,6000];

dado3(2,:) = [0,0,0,0,0,0];

dado3(3,:) = [1,20,110,176,190,240];

dado3(4,:) = [2,60,180,220,285,327];

dado3(5,:) = [3,68,240,349,380,428];

dado3(6,:) = [4,77,310,450,510,620];

dado3(7,:) = [5,110,405,503,623,785];

temp = table2(dado3,3.1,3800)
```

A resposta será mostrada em temp = 336.68 F.

Comando spline

Uma spline cúbica é uma curva suave construída passando através do conjunto de pontos.
A curva entre cada par de pontos é determinada por um polinômio do terceiro grau, que é calculado para fornecer uma curva suave entre os pontos ao invés de ligá-los simplesmente.
O comando ***spline*** realiza no MATLAB uma spline cúbica. O primeiro argumento do comando *spline* é o x, o segundo é o y e o terceiro contêm o valor do(s) ponto(s) aonde se deseja o valor da função. Lembrando que novamente os valores de x devem ser ordenados ou crescentemente ou decrescentemente, caso contrário surgirá uma mensagem de erro!

Exemplo 2

Supondo que queiramos usar a spline cúbica para calcular a temperatura na cabeça do cilindro no tempo t = 2.6 segundos, podemos usar os seguintes comandos:

```
x = [0,1,2,3,4,5];
```

```
y = [0,20,60,68,77,110];
temp1 = spline(x,y,2.6)
```

O valor de temp1 será 67.3.

Se quisermos usar estes processos para calcularmos a temperatura em diferentes momentos podemos usar os seguintes comandos:

```
temp2 = spline(x,y,[2.6,4.9]);
temp2 = [67.3,105.2]
```

Se quisermos ainda plotar uma curva spline abrangendo outro intervalo de valores, podemos gerar um vetor x como o terceiro argumento do comando *spline*.

Exemplo 3

```
x = [0,1,2,3,4,5];
y = [0,20,60,68,77,110];
newx = 0: 0.1 :5;
newy = spline(x,y,newx);
axis([-1,6,-20,120]);
plot (x,y,newx,newy,x,y,´o');
title (' Interpolação Spline ');
xlabel(' Tempo,s ');
ylabel(' Graus, F ');
grid;
```

Note que na interpolação linear, o gráfico de x e y percorre as coordenadas por meio de retas, enquanto que o gráfico de newx e newy representa a spline definida por interpolação cúbica.

7.2. Ajuste de Curvas pelo Método dos Mínimos Quadrados

Supondo que tenhamos um conjunto de pontos originados de um determinado experimento e que queiramos plotar o seu gráfico. Se tentarmos traçar uma única reta entre esses pontos, somente um par destes pontos irá fazer parte da reta. O método dos mínimos quadrados poderá ser usado neste caso para achar uma única reta que mais se aproxime de todos os pontos. Embora essa reta seja a melhor aproximação possível, pode acontecer da reta não passar efetivamente por nenhum ponto.
Note que este método é muito diferente da interpolação porque esta passará por todos os pontos.
Vamos partir primeiro para a discussão do ajuste da reta para um conjunto de pontos e depois para o ajuste do polinômio através do conjunto de pontos.

Regressão Linear

É o processo que determina a equação linear, ou seja, a função mais aproximada do comportamento dos pontos, que é calculada através do somatório dos mínimos quadrados das distâncias entre a reta e os pontos. Como exemplo vamos ainda considerar aqueles valores de temperaturas do cilindro:

```
x = [0,1,2,3,4,5];
y = [0,20,60,68,77,110];
axis([-1,6,-20,120]);
```

Se simplesmente plotarmos o gráfico através do comando:
plot(x,y,x,y, ‘o’);

Ele ligará os pontos. Mas, se ao invés disso, estimarmos o comportamento da função em y1 = 20*x, e aí sim plotarmos este gráfico:

```
plot(x,y1,x,y, ‘o’)
```

Para medirmos a qualidade desta estimativa, devemos determinar a distância no eixo vertical de cada ponto à reta estimada e somá-las através do comando sum. Observe que somamos os quadrados das distâncias para evitar que algum valor seja anulado devido aos sinais.

```
somadist = sum ((y - y1) .^ 2);
```

Para achar a reta mais perto de todos os pontos devemos achar a menor soma dos quadrados das distâncias. Para isto devemos escrever a equação geral da reta : y = mx + b.
Os valores de m e b poderão ser calculados através do comando *polyfit*

Comando polyfit

Este comando acha os coeficientes do polinômio que estamos procurando. Mas, para isto devemos especificar o grau do polinômio. Este comando possui três argumentos: primeiro as coordenadas x e y, e depois o grau do polinômio.

Exemplo:

```
x = [0,1,2,3,4,5];
y = [0,20,60,68,77,110];
coef = polyfit(x,y,1);
```

```
m = coef (1);
b = coef (2);
ybest = m*x+b;
somadist = sum ((y - ybest) .^ 2 );
axis([-1,6,-20,120]);
plot(x,ybest,x,y, 'o' );
title (' ')
xlabel ('X'); ylabel('Y');
grid;
```

Comando polyval

Este comando é empregado para estimar o mínimo polinômio quadrado de um conjunto de pontos. O primeiro argumento deste comando conterá os coeficientes do polinômio, o segundo argumento será um vetor com os valores de x para os quais desejamos o valor da função.

Exemplo:

```
ybest = polyval (coef,x);
```

8. Comandos de Fluxo e Operadores Lógicos e Relacionais

8.1. Operadores Relacionais

O MATLAB tem operadores relacionais que podem ser usados para comparar duas matrizes de mesma ordem ou para comparar uma matriz e um escalar, como os mostrados a seguir:

Operador	**Descrição**
<	Menor que
<=	Menor ou igual a
>	Maior que
>=	Maior ou igual a
= =	Igual a (no sentido de condição)
~ =	Não igual a

A finalidade dos operadores é fornecer respostas a perguntas do tipo falso/verdadeiro.

Assim, se a comparação for verdadeira, atribui-se o valor **1**; se for falsa, o valor **0**.
Considere a expressão lógica a seguir:
a < b
Se a e b forem escalares, então o valor da expressão será 1 (verdadeira) se a for menor que b; caso contrário, a expressão será 0 (falsa). Se a e b forem vetores com os valores a seguir:
a = [2 4 6]
b = [3 5 1]
Então, o valor de a < b será o vetor [1 1 0], enquanto o valor de a ~ = b será [1 1 1].

8.2 - Operadores Lógicos

Podemos combinar expressões usando os operadores lógicos do MATLAB. Os operadores são representados pelos seguintes símbolos.

Operadores	**Descrição**
&	e
|	ou
~	não

Quando duas expressões são unidas por **e**, o resultado será 1 (verdadeiro) se ambas expressões forem verdadeiras, para expressões unidas por **ou**, o resultado será 1 (verdadeiro) se uma ou ambas expressões forem verdadeiras. Assim, para a seguinte expressão lógica.

a < b & b < c

O resultado será 1 (verdadeiro) somente se a < b < c; e falso (0) para todos resultados diferentes. A operação só será válida se as matrizes resultantes (a < b e b <c) tiverem o mesmo tamanho.

Ciclo For

O loop **for** possibilitam que uma série de comandos seja repetida por um número de vezes fixo e predefinido. Vale ressaltar que o comando for não pode ser encerrado atribuindo-se valores ao contador dentro do loop.

O ciclo for é dividido em três partes:

- A primeira parte (i=1) é realizada uma vez, antes de o ciclo ser inicializado.

- A segunda parte é o teste ou condição que controla o ciclo, (i<=5). Esta condição é avaliada; se verdadeira, o corpo do ciclo (X(i) =i^2) é executado.

● A terceira parte acontece quando a condição se torna falsa e o ciclo termina. O comando **end** é usado como limite inferior do corpo do ciclo.

São comuns construções em que conjuntos de ciclos **for** são usados principalmente com matrizes:

Ex:

```
>> for i = 1:8
for j= 1:8,
A(i,j)= i+j;
B(i,j)=  i-j;
end
end
>> C=A+B
C =
 2   2   2   2   2   2   2   2
 4   4   4   4   4   4   4   4
 6   6   6   6   6   6   6   6
 8   8   8   8   8   8   8   8
10  10  10  10  10  10  10  10
12  12  12  12  12  12  12  12
14  14  14  14  14  14  14  14
16  16  16  16  16  16  16  16
```

Ciclo While

No ciclo **while** apenas a condição é testada. Por exemplo, na expressão:

Ex:

```
>>a = 1; b = 15;
while a<b,
clc
a = a+1
```

```
b = b-1

pause(1)

end

a =

 8

b =

 8
```

Estrutura If – Else – End:

Utilizamos esta estrutura quando queremos condicionar a execução de uma dada instrução ou comando ao valor (verdadeiro ou falso) de uma da expressão relacional ou lógica.
O quadro seguinte apresenta a sintaxe da estrutura IF – ELSE - END.

```
 if   <condição>
      <comandos 1>
else
      <comandos 0>
End
```

Se o resultado da expressão lógica <condição > for 1 (verdadeiro) então a lista <comandos 1> será executada. Se <condição> for 0 (falso) então será a lista <comandos 0> a ser executada.
Os **Comandos** são executados apenas se a **expressão** for verdadeira.

Vejamos aqui exemplos que mostram essas funções:

EX$_1$:

```
a = input('Entre com o valor de a : ');
if a > 0
b = log(a)
else
b = exp(a)
end
```

Por exemplo, o programa do arquivo **estcond2.m** (modo como ele foi salvo) quando executado resultará:

```
>> estcond2
Entre com o valor de a : 5
b = 1.6094
```

Se a for positivo, então o logaritmo natural de a será atribuído a b e se a for negativo ou nulo, então b será igual ao exponencial de a.

EX2:

```
% calculo da velocidade de um carro a uma distância d do edifício;

for d = 1:20

if d < 10

v = 0.425 + 0.00175*d^2

elseif d == 10

v = 0

elseif d > 10

v = 0.625 + 0.12*d - 0.00025*d^2

end

end
```

Após o ultimo comando **end**, aparecerão todos os resultados de velocidade para a dada distância.

9. Gráficos

Podemos construir gráficos no MATLAB através de simples comandos. Para plotar gráficos 2D, utilizamos os comandos listados na tabela a seguir:

Comando	Descrição do Comando
Plot	plotar linear
logplot	plotar em escala loglog
semilogx	plotar em escala semilog
semilogy	plotar em escala semilog
fill	desenhar polígono 2D
polar	plotar em coordenada polar
bar	gráfico de barras
stem	seqüência discreta
stairs	plotar em degrau
errorbar	plotar erro
hist	plotar histograma
rose	plotar histograma em ângulo
compass	plotar em forma de bússula
feather	plotar em forma de pena

fplot	plotar função
comet	plotar em trajetória de cometa

Podemos determinar os estilos e cores das linhas utilizadas para plotar o gráfico. Veja nas tabelas a seguir os estilos e cores que podemos controlar:

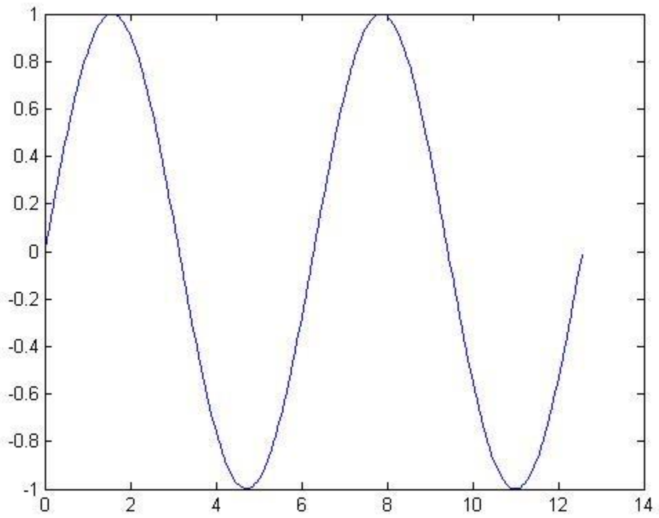
Tipo de Linha	
-	-------------------
_	_____________
-.	-.-.-.-.-.-.-.-.-.-.
.	....
Tipo de Ponto	
.	....
*	*************
o	Oooooooooooooooooooo
+	++++++++++
x	Xxxxxxxxxxxxxx
Cores	
y	Amarelo
m	Lilás
c	azul claro
r	Vermelho
g	Verde
b	azul escuro
w	Branco
k	Preto

Vejamos alguns exemplos simples que demonstram a aplicação de algumas dessas funções:

Ex$_1$:

```
>> x=0:0.05:4*pi;

>> y=sin(x);

>> plot(x,y)

>>
```

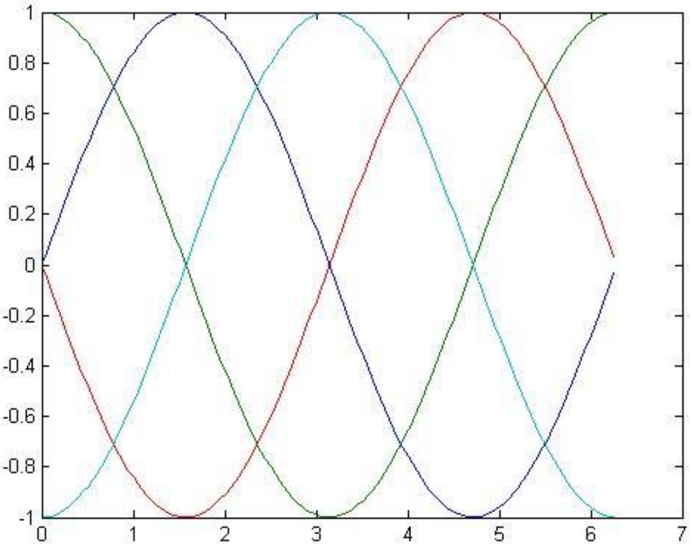
Aparecerá:



Ex2:

```
>> x=0:0.05:2*pi;
>> plot(x,sin(x),x,cos(x),x,sin(x+pi),x,cos(x+pi));
>>
```

Aparecerá:



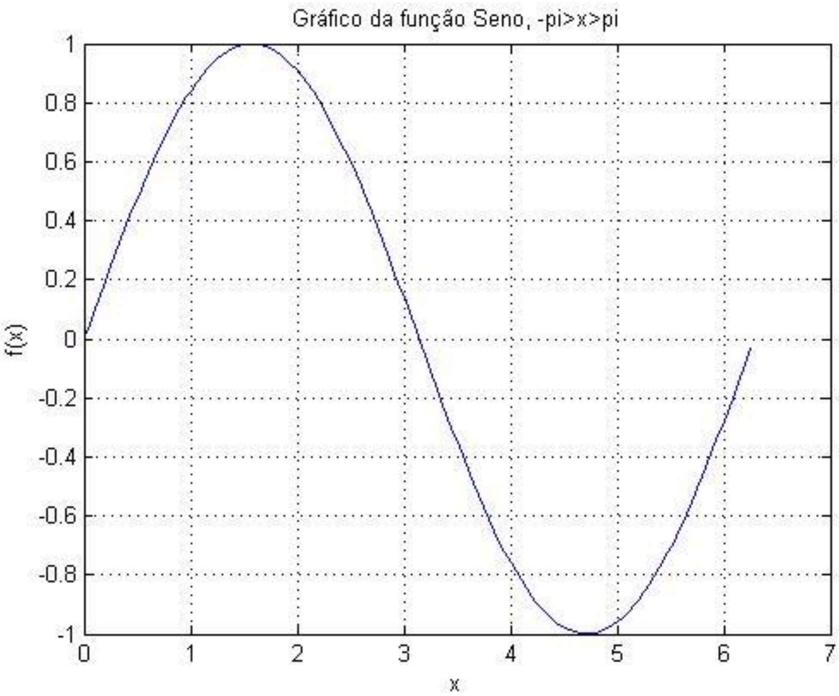
É possível acrescentar informações a um gráfico, através dos seguintes comandos.

Comando	Descrição do Comando
title	título do gráfico
xlabel	título do eixo x
ylabel	título do eixo y
zlabel	título do eixo z
text	inserir anotação no gráfico
gtext	inserir anotação com o mouse
grid	linhas de grade

Vejamos mais alguns exemplos:

```
>> x:-pi:0.01:pi;

>> y=sin(x);

>> plot(x,y);

>> title('Gráfico da função Seno, -pi>x>pi');

>> xlabel('x');

>> ylabel('f(x)');

>> grid;

>>
```

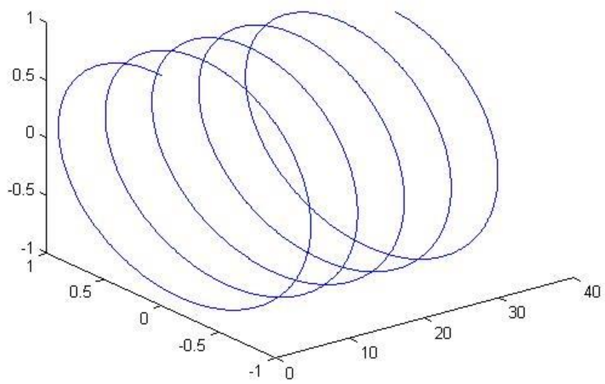
Aparecerá:



Algumas funções que plotam o gráfico 3D:

Comando	Descrição do Comando
plot3	plotar em espaço 3D
fill3	desenhar polígono 3D
Comet	plotar em 3D com trajetória de cometa
Contour	plotar contorno 2D (projeção)
contour3	plotar contorno 3D
Clabel	plotar contorno com valores
Quiver	plotar gradiente
Mesh	plotar malla 3D
Meshc	combinação mesh/contour
Surf	plotar superfície 3D
Surfc	combinação surf/contour
Slice	plota visualização volumétrica
Cylinder	gerar cilindro
Sphere	gerar esfera

Vejamos um exemplo:



10. Integração Numérica

A integração e diferenciação são conceitos fundamentais usados para resolver um grande número de problemas na Engenharia e na Ciência. Enquanto muitos destes problemas se usam de soluções analíticas, muitos requerem soluções numéricas para serem entendidos.

10.1. Integração Numérica

A integral de uma função f(x) no intervalo [a,b], é definida como sendo a área sob a curva percorrida por f(x) entre a e b.

k = $\int_a^b f(x)dx$

O MATLAB possui três comandos para calcular a área sob uma função, em um domínio finito, que são: **trapz, quad e quad8.**

10.2. Regra do Trapézio

Quando a área sob a curva pode ser representada por trapézios e o intervalo [a,b], dividido em n partes iguais, a área aproximada poderá ser calculada através da seguinte fórmula:

Kt =((b – a/2n)* (f(x_0) + 2 f(x_1) + ... + 2f(x_n - 1) + f(x_n))

onde os valores de x_i representam os pontos no final da cada trapézio e x_0 = a e x_n = b.

A estimativa da integral melhora quando usarmos um maior número de componentes (como por exemplo trapézios), para aproximar a área sob a curva, pois quanto menor for o intervalo da função a curva tende a uma reta.

Função **trapz**: numericamente avalia um integral usando a regra dos Trapézios. **T=trapz(x,y)** – aproxima a integral duma função definida pelos pontos com abcissas x e ordenadas y usando a regra dos trapézios.

Ex:

1. Dada a função a função f(x) definida pontualmente através da seguinte tabela:

x	f(x)
-1	-2
-0.5	12
0	-6
0.5	10
1	11

e I= $\int_{-1}^{1} f(x)dx$**,** utilize o Matlab para determinar um valor aproximado de I pela regra dos trapézios.

Solução em MATLAB:

```
>> x=[-1 -0.5 0 0.5 1]

x =

  -1.0000   -0.5000         0    0.5000    1.0000

>> y=[-2 12 -6 10 11]
```

```
y =

 -2   12   -6   10   11

>> I=trapz(x,y)

I =

10.2500

>>
```

10.3. Regra de Simpson

O MATLAB possui dois comandos para desenvolver a integração numérica. O comando **quad** usa uma forma adaptada da regra de Simpson, enquanto o comando **quad8** usa uma forma adaptada da regra de Newton-Cotes. O comando **quad8** funciona melhor em certas funções com certos tipos de singularidades como, por exemplo:

k= $\int_0^1 \sqrt{x}\,dx$**,** lembrando que uma singularidade é um ponto no qual uma função ou sua derivada não são definidas ou tendem para o infinito. Ambas as funções escrevem na tela uma mensagem quando detectam uma singularidade, mas ainda assim o valor estimado da integral é retornado.

A forma mais simples do comando **quad** requer três argumentos: o primeiro argumento é o nome da função no MATLAB que reconhece a função que estamos tratando; o segundo e o terceiro argumento são os limites inferior e superior a e b da integral.

11. Diferenciação Numérica

A derivada de uma função f em um ponto pode ser descrita graficamente como a inclinação da reta que tangencia a função naquele ponto. Pontos da função onde a derivada é zero são chamados pontos críticos. São pontos onde a tangente é representada por uma linha horizontal e que, por isso, definem o local de máximo e de mínimo da função.

Podemos perceber ao analisar uma determinada função num determinado intervalo que o sinal da derivada pode mudar, e, se esse sinal muda, significa que dentro deste intervalo existe local de máximo e local de mínimo.

Podemos também analisar uma função pela sua derivada segunda. De modo que, se a derivada segunda de um ponto crítico é positiva, então o valor da função naquele ponto significa um local de mínimo. Da mesma forma, se a derivada segunda de um ponto crítico é negativa, então a função possui um local de máximo.

11.1. Comando diff

O comando diff calcula a diferença entre dois pontos adjacentes num vetor, gerando um novo vetor com a diferença (Se o comando diff for aplicado a uma matriz, ele irá operar como se cada coluna da matriz fosse um vetor). Por exemplo, assumindo que o vetor x seja [0,1,2,3,4,5], e que o vetor y seja [2,3,1,5,8,10]. O vetor gerado por diff(x) será [1,1,1,1,1], enquanto que o gerado por diff(y) será [1,-2,4,3,2].

A derivada dy será calculada por diff(y) ./ diff(x). Note que estes valores de dy estarão corretos para ambas as formas de diferenças, backward ou forward. A diferença entre esses dois métodos para o cálculo da derivada é determinada pelos valores de x que correspondem à derivada dy. Se os valores correspondentes de x forem [1,2,3,4,5] então dy é calculado pela diferença backward; mas se os valores de x forem [0,1,2,3,4] então dy será calculado pelo método da diferença forward.

Supondo que desejamos analisar a função dada pelo seguinte polinômio:

$f(x) = x^5 - 3x^4 - 11x^3 + 27x^2 + 10x - 24$

Assumindo que queiramos calcular o valor de sua derivada no intervalo [-4,5], usando o método da diferença backward.

Chamando f ’(x) de df e, xd os valores de x da derivada.

Temos no MATLAB que:

```
x = -4:0.1:5;

f = x .^5 - 3 * x .^4 - 11 * x.^3 + 27 * x .^2 + 10 * x - 24;

df = diff(y) . / diff(x);

xd = x(2:length(x) );

plot(f,x)

plot(df,xd)

axis([-4 5 -800 600]);

plot(f)

axis([-4 5 -200 1400]);
```

```
plot(df)
```

Podemos marcar os locais dos pontos críticos para essa função com os seguintes comandos:

```
produto = df(1 : length(df) - 1 ) .* df(2 : length(df) );

critico = xd (find (produto < 0) )
```

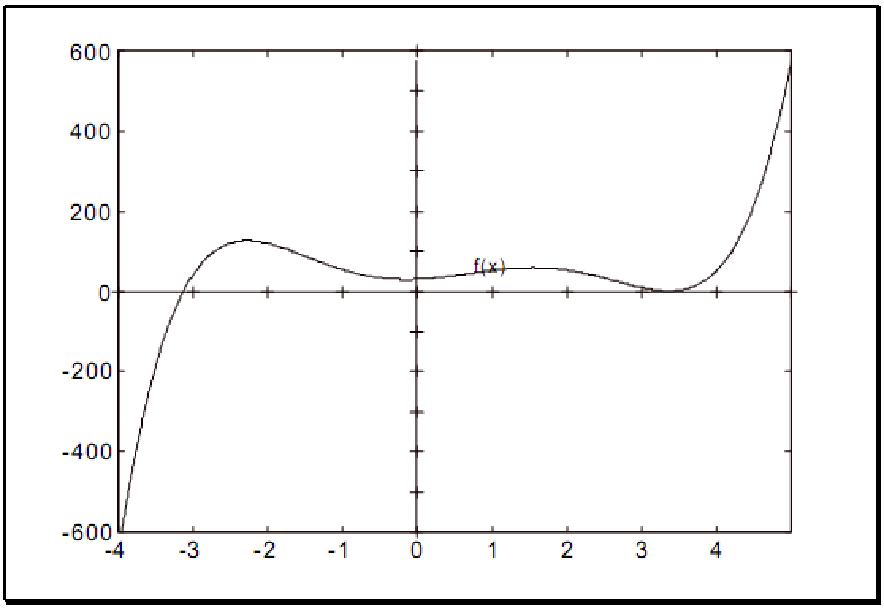
O comando **find** determina os índices dos locais do produto para os quais a derivada df(k) é igual a zero; esses índices são então usados com o vetor contendo os valores de xd para determinar os locais de pontos críticos.

Outro exemplo:

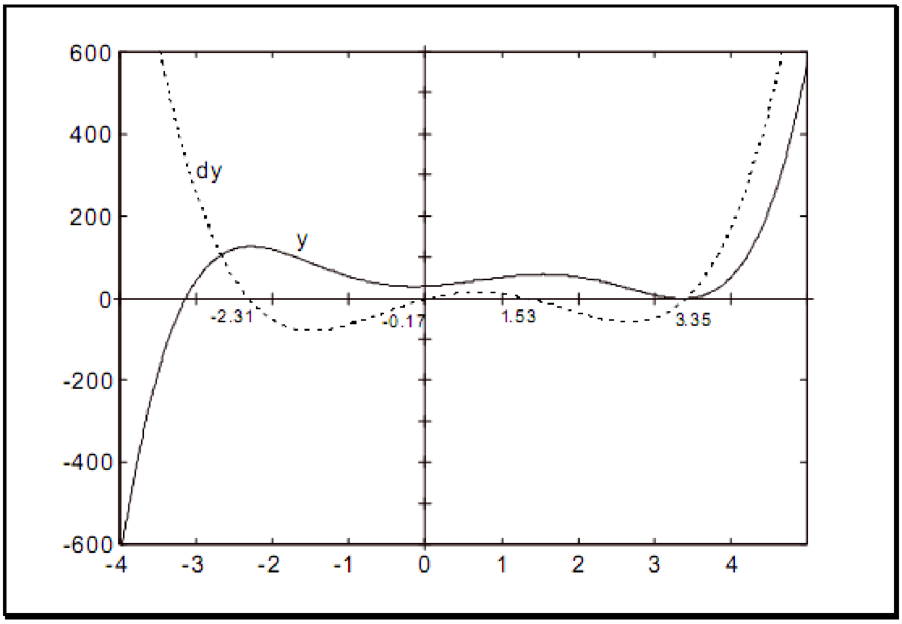
Seja a função definida pelo seguinte polinômio:

$f(x) = x^5 - 3x^4 - 11x^3 + 27x^2 + 10x - 24$

Graficamente




```
>> x = -4:0.1:5;

>> y = x.^5 - 3*x.^4 - 11*x.^3 + 27*x.^2 + 10*x -24;

>> dy = diff(y)./diff(x);
```



Esse gráfico representa a função e sua respectiva derivada.

Apêndice

Tabelas de Referência

O MATLAB apresenta várias categorias principais de funções. Algumas das funções são incorporadas no próprio interpretador, enquanto outras encontram-se sob a forma de arquivos M. As funções de arquivos M, assim como os arquivos M contendo texto de ajuda para as funções incorporadas, estão organizadas em alguns diretórios, cada um deles contendo os arquivos associados a uma dada categoria. O comando help do MATLAB apresenta uma tabela on-line dessas categorias principais.

Principais Categorias de Funções MATLAB

color	Controle de cores e luminosidade.
datafun	Análise de dados e funções da transformação de Fourier.
demos	Demonstrações e exemplos.
elfun	Funções de matemática elementar.
elmat	Manipulação de matrizes e matriz elementares.
funfun	Função de Função - Métodos Numéricos não-lineares.
general	Comandos gerais.
graphics	Comandos gerais de gráficos.
iofun	Funções de baixo nível de entrada e saída de arquivos

As páginas seguintes contêm tabelas de funções em cada uma dessas áreas específicas. Caso seja executado o comando help com algum dos nomes de diretórios listados no lado esquerdo dessa tabela, o Matlab apresentará uma versão on-line das tabelas dentro daquela área.

Comandos de Aplicação Geral

Controle de Comandos e Funções

`demo`	Executa demonstrações.
`expo`	Executa programa EXPO de demonstração do MATLAB
`help`	Documentação on-line.
`info`	Informação sobre MATLAB e The MathWorks.
`lasterr`	Última mensagem de erro gerada.
`lookfor`	Procura palavra-chave entre os itens de ajuda.
`path`	Controle de procura de caminhos do MATLAB.
`subscribe`	Torna aprovado um usuário MATLAB no MathWorks.
`type`	Mostra o conteúdo de um arquivo M-File.
`ver`	Versão do MATLAB e ToolBox corrente.
`version`	Número de versão do MATLAB corrente.
`what`	Lista arquivos M, MAT e MEX.
`whatsnew`	Mostra arquivos LEIA-ME para MATLAB e ToolBox
`which`	Arquivos e funções locais.

Trabalhando com Arquivos e Ambiente Operacional	
cd	Muda o diretório de trabalho corrente.
cedit	Coloca parâmetros em comandos editados(somente UNIX).
delete	Apaga arquivos.
diary	Salva sessão de textos de MATLAB.
dir	Lista o diretório.
getenv	Obtém o valor de ambiente.
hostid	Número de identificação do servidor anfitrião MATLAB.
Ls	Lista o diretório.
Matlabroot	Diretório raiz da instalação MATLAB.
Pwd	Mostra o corrente diretório de trabalho.
tempdir	Nome do sistema do diretório temporário.
tempname	Único nome para arquivo temporário.
terminal	Obtém o tipo de terminal gráfico.
unix	Executa comando do sistema operacional; retornando o resultado.
!	Executa comando do sistema operacional

Controle de Janelas do Windows	
clc	Limpa janela de comando.
echo	Repete comandos para dentro de arquivos de texto.
format	Formato de saída do número.
home	Envia o cursor para a base, casa
more	Controle da listagem de informações em janelas de comando.

Iniciando e Saindo do MATLAB

matlabrc	Executa arquivos *.M.
quit	Fecha o MATLAB.
startup	Executa arquivos *.M quando MATLAB é invocado.

Operadores e Caracteres Especiais

+	Adição.
-	Subtração.
*	Multiplicação matricial.
.*	Multiplicação de vetorial.
^	Potência matricial.
.^	Potência vetorial.
kron	Produto de tensor Kronecker.
\	Corte ou divisão à esquerda.
/	Corte ou divisão à direita.
./	Divisão vetorial.
:	Dois pontos.
( )	Parênteses.
[ ]	Colchetes.
.	Ponto Decimal.
. .	Diretório raiz.
. . .	Continuação.
,	Vírgula.
;	Ponto-e-vírgula.
%	Comentário, observação.
!	Ponto de exclamação.
¢	Transposição.
.'	Transposição Vetorial.
=	Indicação.
==	Igualdade.
< >	Operadores relacionais.

Funções Lógicas

all	Verdadeiro se todos elementos do vetor forem verdadeiros.
any	Verdadeiro se algum elemento do vetor for verdadeiro.
Exist	Verifica se existe variáveis ou funções.
find	Encontra índices dos elementos não nulos.
finite	Verdadeiro para elementos finitos.
isempty	Verdadeiro para matrizes vazias.
ishold	Verdadeiro se hold estiver ligado.
isieee	Verdadeiro para IEEE pontos flutuantes aritméticos.
isinf	Verdadeiro para elementos infinitos.
Isletter	Verdadeiro para caracteres alfabéticos.
isnan	Verdadeiro para um não número.
isreal	Verdadeiro se todos elementos da matriz forem reais.
issparse	Verdadeiro para matrizes esparsas.
isstr	Verdadeiro para texto.

Construção de Linguagem e Depuração

MATLAB como uma Linguagem de Programação

eval	Executa funções em formato texto no MATLAB.
Feval	Executa funções especificadas nas variáveis texto.
function	Adiciona novas funções.
global	Define variáveis globais.
nargchk	Número de validade de argumentos colocados.

Controle de Fluxo

break	Execução terminal de um Loop.
else	Usado com if.
elseif	Usado com if.
end	Termina com o campo de ação de comandos for, while e if.
error	Mostra mensagem e aborta funções.
for	Repete declarações num especificado número de vezes.
if	Executa declarações condicionais.
return	Retorna para funções invocadas.
while	Repete declarações num indefinido número de vezes.

Entrada Interativa

input	Preparação para entrada de usuário.
keyboard	Declara o teclado como se fosse um arquivo texto.
menu	Menu geral de escolhas para entrada do usuário.
pause	Espera pela resposta do usuário.

Depuração

dbclear	Remove ponto de parada.
dbcont	Execução resumida.
dbdown	Muda o contexto local da estação de trabalho.
dbquit	Sai do modo debug.
dbstack	Lista quem chamou quem.
dbstatus	Lista todos pontos de parada.
dbstep	Executa uma ou mais linhas.
dbstop	Coloca um ponto de parada.
dbtype	Lista arquivo *.M com número de linhas.
dbup	Muda o contexto local da estação de trabalho

Manipulação de Matrizes

Matrizes Elementares	
eye	Matriz Identidade.
gallery	Teste de matrizes - condição matricial e autovalores.
Linspace	Vetor linearmente esparsado.
logspace	Vetor logaritmicamente esparsado.
meshgrid	Matrizes X e Y para plotes 3-D.
ones	Matriz de elementos unitários.
rand	Distribui uniformemente números ao acaso.
randn	Distribui normalmente números ao acaso.
zeros	Matriz de elementos nulos.
:	Vetor regularmente esparsado.

Variáveis e Constantes Especiais	
ans	Mais recente resposta.
computer	Escrita computacional.
eps	Relativa precisão no ponto-flutuante.
flops	Operações enumeradas de ponto flutuante.
i,j	Unidade imaginária.
inf	Infinito.
NaN	Não é número.
nargin	Número de entradas de argumentos da função.
nargout	Número de saídas de argumentos da função.
pi	3.1415926535897 ...
realmax	Maior número de ponto flutuante.
realmin	Menor número de ponto flutuante.

Data e Hora	
clock	Relógio.
cputime	Tempo decorrido em unidades de CPU.
date	Calendário.
etime	Função do tempo decorrido.
tic, toc	Funções de cronometragem

Manipulação Matricial

Matrizes Especiais	
compan	Matriz companheira
hadamard	Matriz Hadamard.
hankel	Matriz Hankel.
hilb	Matriz Hilbert.
invhilb	Matriz inversa de Hilbert.
magic	Matriz quadrada cujas as somas das linhas e colunas são iguais.
pascal	Matriz Pascal.
rosser	Problema clássico de teste de simetria dos autovalores.
toeplitz	Matriz Toeplitz.
vander	Matriz Vandermonde.
wilkinson	Autovalores obtidos para a matriz de Wilkinson

Funções Matemáticas

Funções de Matemática Elementar

Funções de Matemática Elementar	
abs	Valor absoluto.
acos	Inversa do coseno.
acosh	Inversa do coseno hiperbólico.
acot	Inversa da cotangente.
acoth	Inversa da cotangente hiperbólica.
acsc	Inversa da cosecante.
acsch	Inversa da cosecante hiperbólica.

ceil	Inteiro próximo a mais infinito.
conj	Conjugado de um número complexo.
cos	Coseno.
cosh	Coseno hiperbólico.
cot	Cotangente.
coth	Cotangente hiperbólica.
csc	Cosecante.
csch	Cosecante hiperbólico.
exp	Exponencial.
Fix	Inteiro próximo a 0.
gcd	Grande divisor comum.
imag	Parte imaginária de um número complexo.
lcm	Menor múltiplo comum.
log	Logaritmo natural.
log10	Logaritmo na base10.
real	Parte real de um número complexo.
rem	Resto da divisão.
round	Inteiro mais próximo.
sec	Secante.
sech	Secante hiperbólica.
sign	retorna o sinal de um número. Ex.: sign(1.2)=1,sign(-23.4)=-1e sign(0)=0
sin	Seno.
sinh	Seno hiperbólico.
sqrt	Raiz quadrada.
tan	Tangente.
tanh	Tangente hiperbólica.

Funções Especializadas da Matemática

Funções de Matemática Especializada	
bessel	Funções Bessel.
besseli	Funções Bessel modificada de primeiro tipo.
besselj	Funções Bessel de primeiro tipo.

Continua.......

besselk	Funções Bessel modificada de segundo tipo.
bessely	Funções Bessel de segundo tipo.
beta	Funções beta.
betainc	Funções beta incompleta.
betaln	Logaritmo da função beta.
ellipj	Funções elípticas de Jacobian.
ellipke	Integral elíptica completa.
erf	Função de erro.
erfc	Função de erro complementar.
erfcx	Escala complementar da função de erro.
erfinv	Função inversa de erro.
expint	Integral exponencial.
gamma	Função gama.
gammainc	Função gama incompleta.
gammaln	Função logarítmica de gama.
legendre	Funções associadas Legendre.
log2	Separação de números de ponto flutuante.
pow2	Escala de números de ponto flutuante.
rat	Aproximação racional.
rats	Saída racional.

Funções Matriciais

Análise Matricial	
cond	Número da condição matricial.
det	Determinante.
etree	Árvore eliminatória de uma matriz.
norm	Matriz ou vetor normal.
null	Espaço nulo.
orth	Ortogonalização.
rcond	Estimação condicional recíproca LINPACK.
rank	Número de linhas ou colunas linearmente independentes.
rref	Reduzir linhas da forma ECHELON.
subspace	Ângulo entre dois sub-espaços.
trace	Soma dos elementos diagonais.

Equações Lineares	
chol	Fatorização CHOLESKY.
inv	Matriz inversa.
lscov	Últimos quadrados na presença de covariance.
lu	Fatores de eliminações Gausianas.
nnls	Matrizes quadradas não negativas.
pinv	Pseudo-inversa.
qr	Decomposição ortogonal - triangular
\ e /	Solução de equação linear

Autovalores e Valores Singulares	
balance	Escala diagonal para melhorar precisão de autovalores.
cdf2rdf	Forma da diagonal complexa para real forma de bloco diagonal.
eig	Autovalores e autovetores.
hess	Forma Hessenberg.
poly	Polinômio característico.
qz	Autovalores gerais.
rsf2csf	Forma de diagonal de bloco para forma de diagonal complexa.
schur	Decomposição Schur.
svd	Decomposição de valor singular.

Funções Matriciais

Análise de Dados e as Funções da Transformada de Fourier

Operações Básicas

cumprod	Produto cumulativo de elementos.
cumsum	Soma acumulativa de elementos.
max	Maior componente.
mean	Média ou valor significativo.
median	Mediana
min	Menor componente.
prod	Produto de elementos.
sort	Organiza em ordem ascendente.
std	Desvio padrão.
sum	Soma de elementos.
trapz	Integração numérica usando método trapezoidal.

Diferenças Finitas

del2	Ponto cinco discreto Laplaciano.
diff	Função diferencial e derivada aproximada.
gradient	Gradiente aproximado.

Correlação

corrcoef	Coeficientes correlacionados.
cov	Matriz covariante

Transformada de Fourier

abs	Magnitude.
angle	Ângulo de fase.
cplxpair	Organiza números para dentro de um par conjugado complexo.
fft	Discreta transformada de Fourier.
fft2	Discreta transformada de Fourier bidimensional.
fftshift	Muda a freqüência zero para centro do espectro.
ifft	Inversa discreta da transformada de Fourier.
ifft2	Inverso bidimensional da discreta transformada de Fourier.
nextpow2	Potência de 2 superior mais próxima.
unwrap	Remove o ângulo de fase saltando através de limites de 360°

Funções Vetoriais

cross	Produto de vetores.
dot	Produto escalar.

Funções Polinomiais e Interpolares

Polinômios

conv	Multiplicação polinomial.
deconv	Divisão polinomial.
poly	Construção polinomial com raízes especificas.
polyder	Derivada polinomial.
polyeig	Solução polinomial para problemas de autovalores.
polyfit	Ajuste polinomial para dados.
polyval	Cálculo do grau polinomial.

Interpolação de Dados	
griddata	Rede de dados.
interp1	Interpolação unidimensional.
interp2	Interpolação bidimensional.
interpft	Interpolação unidimensional usando método FFT.

Função – Função

Função - Função - Métodos Numéricos não lineares	
fmin	Função minimizada de uma variável.
fmins	Função minimizada de várias variáveis.
fplot	Funções de plotagem.
fzero	Encontra zero da função de uma variável.
ode23	Resolve equações diferenciais pelo método de baixa ordem.
ode 45	Resolve equações diferenciais pelo método de alta ordem.
quad	Avaliação numérica da integral pelo método de baixa ordem.
quad8	Avaliação numérica da integral pelo método de alta ordem.

Funções Matriciais Esparsadas

Matrizes Elementares Esparsadas	
spdiags	Matriz esparsada formada por diagonais.
speye	Matriz identidade esparsada.
sprandn	Matriz esparsada casual.
sprandsym	Matriz esparsada simetricamente casual.

Tudo para Conversão Esparsada	
find	Encontra índices de entradas não nulas.
full	Converte matriz esparsa em matriz completa.
sparse	Cria matriz esparsa de não nulos e índices.
spconvert	Converte para formato externo de matriz esparsa.

Trabalhando com Entradas Não nulas de Matrizes Esparsas	
issparse	Verdadeiro se matriz é esparsa.
nnz	Número de entradas não nulas.
nonzeros	Entradas não nulas.
nzmax	Soma de distribuição armazenada para entradas.
spalloc	Destina memória para entradas não nulas.
spfun	Aplica função para entradas não nulas.
spones	Substitui entradas não nulas por um.

Visualizando Matrizes Esparsadas	
gplot	Plota gráfico, como em "teoria gráfica".
spy	Visualiza estrutura esparsada.

Reordenando Algoritmos	
colmmd	Mínima extensão da coluna.
colperm	Ordena colunas baseadas em contador não nulo.
dmperm	Decomposição Dulmage-Mendelsohn.
randperm	Permutação aleatória de vetores.
symmmd	Mínima extensão simétrica.
symrcm	Ordenando a reversa de Cuthill-McKee.

Norma, Número Condicional e Linha	
condest	Estimativa 1 - condição de norma.
normest	Estimativa 2 - norma.
sprank	Linha estrutural.

Diversos	
spaugment	Sistema ampliado da última forma quadrática.
spparms	Estabelece parâmetros para rotinas de matrizes esparsadas.
symbfact	Análise de fatorização simbólica.

Gráficos Bidimensionais

Gráficos Elementares X-Y

fill	Desenha polígonos bidimensionais preenchidos.
loglog	Plota em escala logarítmica.
plot	Plota em escala linear.
semilogx	Plota em escala linear, somente com o eixo x logaritmizado.
semilogy	Plota em escala linear, somente com o eixo y logaritmizado.

Gráficos Especiais X-Y

bar	Plota em barras gráficas.
comet	Plota em comet animado.
compass	Plota em compass.
errorbar	Plota em erros de barra.
feather	Plota em Feather.
fplot	Plota funções.
hist	Plota em historiogramas.
polar	Plota em coordenadas polares.
rose	Plota em ângulos historiogrâmicos.

Anotações Gráficas

grid	Rede de linhas.
gtext	Lugar do texto com o mouse.
legend	Adiciona legenda para plotar.
text	Anotação de texto.
title	Título do gráfico.
xlabel	Classificação do eixo X.
ylabel	Classificação do eixo Y.

Conversão de Sistemas de Coordenadas

cart2pol	Coordenadas cartesianas para coordenadas polares.
pol2cart	Coordenadas polares para coordenadas cartesianas.

Diversos

zoom	Zoom de aproximação (in) e afastamento (out).

Gráficos Tridimensionais

Comandos de Linha e Área Preenchida	
fill3	Desenha polígonos 3-D preenchidos em espaço 3-D.
plot3	Plota linhas e pontos em espaço 3-D.

Contorno e outras Plotagens Bidimensionais de dados Tridimensionais	
clabel	Classifica a elevação do plot de contorno.
comet3	Plot de comet animado.
contour	Plot de contorno.
contour3	Plot de contorno tridimensional.
contourc	Cálculo do plot de contorno (usado pelo contorno).
image	Mostra imagem.
imagesc	Dados em escala e mostra como imagem.
pcolor	Plota um tabuleiro de damas.

Plotagem de Superfície e Malha	
mesh	Malha da superfície em 3-D.
meshc	Combinação de gráfico de malha/contorno.
meshz	Malha 3-D com plano zero.
slice	Visualização de gráfico volumétrico.
surf	Superfície 3-D sombreada.
surfc	Combinação de plot de superfície/malha.
surfl	Superfície 3-D sombreada com luminosidade.
waterfall	Plota superfícies. Semelhante ao comando mesh.

Aparência Gráfica	
axis	Escala e aparência de eixo.
caxis	Pseudocores de eixo escalar.
colormap	Tabela de cores.
hidden	Retirada da linha de malha oculta .
shading	Sombreamento de cores.
view	Especificação do ponto de vista de um gráfico 3-D.
viewmtx	Visualização de matrizes transformadas.

Anotação Gráfica	
grid	Rede de linhas.
legend	Adiciona legenda no gráfico
text	Anotação.

Objetos 3-D

cylinder Cilindro comum.
sphere Esfera comum.

Conversão de Sistemas de Coordenadas

cart2sph Coordenadas cartesianas para polar.
sph2cart Coordenadas polares para cartesianas.

Funções Gráficas

Criação e Controle de Janelas

capture Captura tela de uma figura atual (somente UNIX).
clf Limpa figura atual.
close Fecha figura.
figure Cria figura (janela gráfica).
gcf Dá mobilidade a figura atual.
graymon Determina as propriedades padrão da figura para monitores com escala de cinza.
newplot Determina os eixos corretos e a figura para novos gráficos.
refresh Redesenha a atual figura da janela.
whitebg Altera figura para cores de fundo.

Criação e Controle de Eixos

axes	Cria eixos em posições arbitrárias.
axis	Controla escala e aparência de eixos.
caxis	Controla pseudocores de escala de eixos.
cla	Limpa eixos atuais.
gca	Torna eixo manuseável.
hold	Controla o gráfico atual.
ishold	Verdadeiro se o controle estiver ligado.
subplot	Cria eixos em várias posições.

Caixas de Diálogos

uigetfile	Recupera nome do arquivo para abrir uma caixa de diálogo.
uiputfile	Recupera nome do arquivo para escrever numa caixa de diálogo.

axes Cria eixos.

Impressão e Armazenagem

orient	Mostra orientações do papel.
print	Imprime gráficos ou salva-os em arquivos.
printopt	Configuração local da impressora.

surface Cria superfície.
text Cria texto.

Filmes e Animações

getframe	Mostra estrutura do filme.
movie	Roda as estruturas do filme gravado.
moviein	Inicializa a memória da estrutura do filme.

drawnow	Começa eventos gráficos pendentes.
findobj	Encontra objetos com propriedades específicas.
gco	Torna um objeto manuseável.
get	Dá as propriedades de um objeto.
reset	Refaz as propriedades de um objeto.
rotate	Rotaciona um objeto.

Diversos

ginput	Entrada do gráfico pelo mouse.
ishold	Retorna ao estado conservado.
rbbox	Caixa de borracha para região selecionada.
waitforbuttonpress	Espera pelo pressionamento de uma tecla sobre a figura.

Controle de Cores e Funções de Luminosidade

Controle de Cores

caxis	Pseudocores da escala de eixos.
colormap	Tabela de consulta de cores.
shading	Modo de sombreamento de cores.

Mapeamento de Cores

bone	Escala cinza com uma matiz azul.
contrast	Acentuação de contraste em escala cinza.
cool	Sombra de cyan e magenta.
copper	Matiz linear usando tonalidades cooper.
flag	Alternando vermelho, branco, azul e preto.
gray	Escala de cinza linear.
hsv	Valor de saturação de tonalidade.
hot	Preto-vermelho-amarelo-branco.
jet	Variação do HSV. (sem contorno)
pink	Pasteuriza sombras de rosa.
prism	Prisma de cores.

Funções relacionadas com Mapeamento de Cores
`brighten` Brilho ou escuridão.
`colorbar` Mostra mapeamento de cores como escala de cores.
`hsv2rgb` Conversão de HSV para RGB.

Funções relacionadas com Mapeamento de Cores

brighten	Brilho ou escuridão.
colorbar	Mostra mapeamento de cores como escala de cores.
hsv2rgb	Conversão de HSV para RGB.
rgb2hsv	Conversão de RGB para HSV.
rgbplot	Plota o mapeamento de cores.
spinmap	Gira o mapeamento de cores.

`surfl` Superfícies 3-D sombreadas com luminosidade.
`surfnorm` Superfícies normais.

Luminosidade

diffuse	Reflexo difuso.
specular	Reflexo refletido.
surfl	Superfícies 3-D sombreadas com luminosidade.
surfnorm	Superfícies normais.

Funções Sonoras

Funções Gerais de Som

saxis	Som em eixos escalares.
sound	Converte vetor para som.

SPARCstation - Funções Sonoras Específicas

auread	Lê arquivos de som .au
auwrite	Escreve arquivos de som .au
lin2mu	Conversão de linear para função mi.
mu2lin	Conversão de mi para linear.

Funções de Som .wav

wavread	Carrega MS-Windows 3.1 .wav no formato de arquivo de som.
wavwrite	Salva MS-Windows 3.1 .wav no formato de arquivo de som.

Funções de Texto

Gerais

abs	Converte texto para valores numéricos.
blanks	Cria texto de vazios.
deblank	Remove arrastando espaços brancos e nulos de textos.
eval	Executa frases com expressão MATLAB.
findstr	Encontra uma letra em um texto.
isstr	Verdadeiro para texto.
setstr	Converte valores numéricos para texto.
str2mat	De texto matricial para letras individuais.
string	Sobre caracteres texto no MATLAB.
strrep	Procura e substitui texto.
strtok	Primeiro toma como texto.

Comparação Frasal

isletter	Verdadeiro para caracteres alfabético.
lower	Converte texto para uma caixa menor.

Conversão de Texto para Número

int2str	Converte inteiro para texto.
num2str	Converte número para texto.
sprintf	Converte número para texto sob um controle formatado.
sscanf	Converte texto para número sob um controle formatado.
str2num	Converte texto para número.

Funções de Arquivos de Entrada e Saída de Baixo Nível

Abrindo e Fechando Arquivo

fclose	Fecha arquivo.
fopen	Abre arquivo.

Entrada e Saída Não-formatada

Fread	Lê dado binário de arquivo.
fwrite	Escreve dado binário para arquivo.

Entrada e Saída Formatada

fgetl	Lê linha de arquivo, descarta novas linhas de caracteres.
fgets	Lê linha de arquivo, permanece novas linhas de caracteres.
fprintf	Escreve dado formatado para arquivo.
fscanf	Lê dado formatado para arquivo.

Posição em Arquivos

feof	Teste para fim de linha.
ferror	Pergunta a situação do erro de arquivo I/O.
frewind	Refaz arquivo.
fseek	Mostra o indicador da posição do arquivo.
ftell	Determina o indicador da posição do arquivo.

Conversão de Texto	
sprintf	Escreve dados formatados para letras.
sscanf	Lê letras sob um controle formatado.

Arquivos Especiais de Entrada e Saída	
csvread	Lê um arquivo de valores separados por (,).
cswrite	Escreve um arquivo de valores separados por (,).
uigetfile	Recupera nome de arquivo para abrir caixa de diálogo.
uiputfile	Recupera nome de arquivo para escrever numa caixa de diálogo.
wk1read	Lê um arquivo Lotus 1-2-3 .wk1
wk1write	Escreve um arquivo Lotus 1-2-3 .wk1