UNESP - Universidade Estadual Paulista
"Julio de Mesquita Filho"
Faculdade de Engenharia
Câmpus de Ilha Solteira – SP

INTRODUÇÃO AO MATLAB

Autores:
Jean Marcos de Souza Ribeiro
Marcio Roberto Covacic

Ilha Solteira - SP

ÍNDICE

1 INTRODUÇÃO 4

1.1 ENTRANDO NO MATLAB 4
1.2 COMO O MATLAB TRABALHA 4
1.3 AMBIENTE DE TRABALHO DO MATLAB 5
1.4 ARQUIVOS .M 5
1.5 COMANDOS BÁSICOS 6
1.5.1 Comandos de propósito gerais 6
1.5.2 Comandos do sistema operacional 7
1.6 ERROS NO MATLAB 7
1.7 HELP 7
1.8 PRIMEIRO EXEMPLO 8

2 MATRIZES, VETORES E ESCALARES 9

2.1 DEFININDO MATRIZES NO MATLAB 9
2.2 OPERADOR DOIS PONTOS (:) 11
2.3 COMANDO SIZE 12
2.4 COMANDO INPUT 12
2.5 COMANDO *FORMAT* 12
2.6 COMANDO *DISP* 13
2.7 COMANDO *FPRINTF* 13
2.8 GRÁFICOS X-Y 14

3 CÁLCULOS FUNDAMENTAIS E MATRIZES ESPECIAIS 16

3.1 VALORES ESPECIAIS E MATRIZES ESPECIAIS 16
3.1.1 Valores especiais 16
3.1.2 Matrizes especiais 16
3.2 OPERAÇÕES ENTRE ESCALARES 17
3.2.1 Hierarquia em Operações Aritméticas 18
3.3 OPERAÇÕES DE CONJUNTOS 19
3.4 FUNÇÕES ELEMENTARES 19
3.4.1 Funções Matemáticas Elementares 20
3.4.2 Funções Trigonométricas 21
3.4.3 Funções Hiperbólicas 22
3.4.4 Funções de arquivo M 22
3.5 NÚMEROS COMPLEXOS 23
3.5.1 Operações Aritméticas com Números Complexos 23
3.5.2 Coordenadas polar e retangulares 23

4 CONTROLE DE FLUXO 25

4.1 OPERADORES LÓGICOS E RELACIONAIS 25
4.1.1 Operadores Relacionais 25
4.2 OPERADORES LÓGICOS 25
4.3 TOMADA DE DECISÕES 26
4.3.1 Estrutura If – Else – End 26
4.4 LOOP FOR 28
4.5 LOOPS WHILE 28

5 OPERAÇÕES COM MATRIZES 30

5.1 OPERAÇÕES COM MATRIZES 30
5.1.1 Matrizes Transpostas 30
5.1.2 Somatório de Produtos 30
5.1.3 Multiplicação de Matrizes 31
5.1.4 Matriz Inversa 31
5.1.5 Determinante 32

5.2 MANIPULAÇÕES COM MATRIZES 32
5.2.1 Comando rot90 ..... *32*
5.2.2 Comando fliplr *32*
5.2.3 Comando flipud *32*
5.2.4 Comando diag *32*
5.2.5 Comando triu *33*
5.2.6 Comando tril *33*
6 GRÁFICOS 34
6.1 GRÁFICOS X – Y 34
6.1.1 Coordenadas Retangulares *34*
6.2 GRÁFICOS POLARES 35
6.2.1 Coordenadas polares *35*
6.3 GRÁFICOS DE BARRAS E DEGRAUS 36
6.4 OPÇÕES 36
6.4.1 Estilo de linha e marcação *37*
6.4.2 Escala *37*
6.4.3 Subplot *38*
6.4.4 Controle de tela *38*

CAPÍTULO 1

1 INTRODUÇÃO

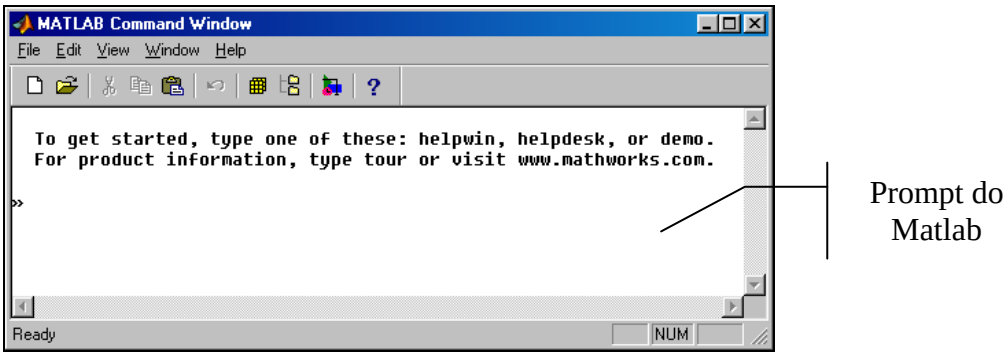
O MATLAB é uma linguagem de programação e uma ferramenta de cálculo muito útil para engenharia em geral. Todos programas são abertos, isto é, o algoritmo utilizado na solução de um problema é conhecido e pode ser modificado. Este fato é, talvez, a maior virtude do MATLAB, pois possibilitou que fossem preparados vários aplicativos, os *toolboxes*, usando a plataforma MATLAB como base. Temos aplicativos na área de controle, análise de sinais, equações diferenciais, robótica, análise modal, finanças, estatística, etc.

O MATLAB é muito fácil de aprender, fácil de utilizar e, quando bem utilizado, aumenta em muito a produtividade. É também uma ferramenta profissional usada mundialmente, principalmente por engenheiros, sobretudo pelos que têm contato com trabalhos experimentais.

A melhor forma de aprender MATLAB é utilizando-o. A experiência mostra que uma vez vencido o acanhamento inicial, o medo de enfrentar algo novo, o trabalho se torna muito agradável.

1.1 ENTRANDO NO MATLAB

Quando entramos a primeira vez no MATLAB, o símbolo » aparece na tela (chamada de prompt do MATLAB). Este símbolo indica que você pode escrever um comando. Os comandos em MATLAB podem terminar com “;” ou não. Quando um comando termina com “;” ele é executado mas o conteúdo das variáveis envolvidas não são mostrados na tela. Para que o resultado apareça na tela é necessário omitir o “;”.



1.2 COMO O MATLAB TRABALHA

Por causa de sua grande variedade de utilizações e possibilidade de programação, o MATLAB trabalha sob forma de linha de comandos. Sua janela é simples e

possui um menu bem básico. Todos os comandos podem ser digitados no *prompt* do MATLAB, e os programas são guardados em arquivos especiais.

Por ex.: para somar dois números do MATLAB simplesmente digita-se:

```
» 2+2
ans =
      4
»
```

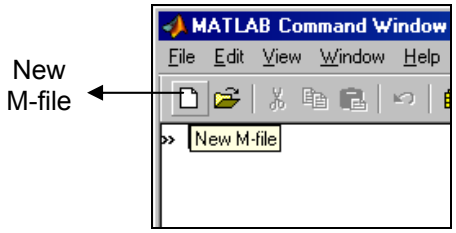
Qualquer comando que desejarmos executar deve ser digitado diretamente no *prompt* e o resultado sai no ambiente de trabalho ou em janelas que são abertas automaticamente no próprio MATLAB, que são outra forma de interface que o MATLAB utiliza para mostrar ou receber dados. Por exemplo, um gráfico.

1.3 AMBIENTE DE TRABALHO DO MATLAB

O ambiente de trabalho do MATLAB é na própria janela do programa sob forma de *prompt* onde o usuário entra com dados e comandos. A janela da área de trabalho possui uma barra de rolagem que permite acompanhar todas as saídas produzidas pelo MATLAB enquanto o usuário está trabalhando. A janela de trabalho também possibilita a edição simples do Windows, ou seja, o usuário pode colar texto para o *prompt* e utiliza-lo como comando, da mesma forma, pode-se também copiar textos ou saídas do MATLAB para outros programas do Windows.

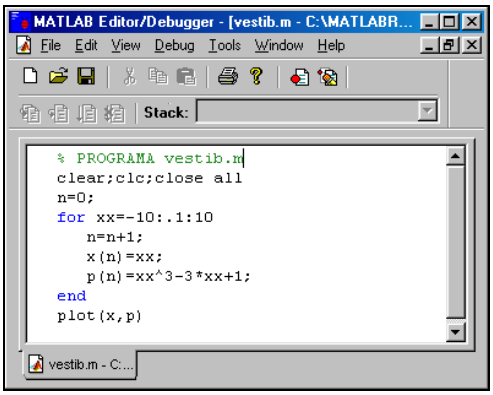
1.4 ARQUIVOS .M

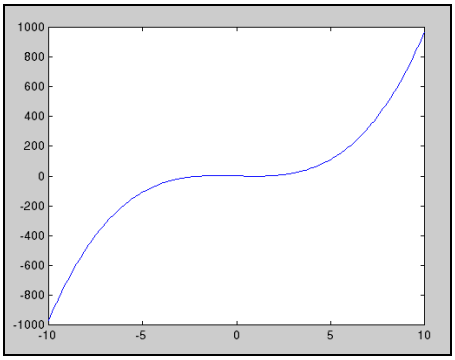
O MATLAB permite que os comandos sejam executados de forma seqüencial em forma de programa, constituindo-se assim de uma linguagem interpretada. Para se abrir o editor de programação basta clicar no menu New M-file.



As seqüências de comandos são gravadas como texto em um arquivo especial com extensão .m e são executados simplesmente digitando-se o nome do arquivo direto no prompt.

Ex.:
» vestib





Para que um arquivo seja executado dessa forma, o mesmo deve estar no diretório atual do MATLAB ou então estar em um dos diretórios denominados path. Para saber em que diretório você se encontra atualmente, basta digitar o comando “cd”.

Por ex.:

```
» cd
C:\MATLABR11\work
»
```

Para mudar o diretório de trabalho, por exemplo, para sua pasta de arquivos, basta digitar o comando “cd caminho”. Por exemplo,

```
» cd c:\usuarios
```

1.5 COMANDOS BÁSICOS

Os comandos no MATLAB podem ser divididos em vários “tipos” conforme seu propósito.

1.5.1 Comandos de propósito gerais

Alguns comandos são de uso geral, e normalmente auxiliam na utilização da área de trabalho, são eles:

clc – limpa a tela.
disp – exibir um texto na área de trabalho.
echo – ligar ou desliga a exibição dos comandos na execução de um programa.
format – ajusta a exibição de casas decimais e formatos numéricos para FORMA.
help comando – mostra, na área de trabalho, o help do mando solicitado.
who / **whos** – mostra as variáveis que estão atualmente na memória.
clear – apaga uma ou mais variáveis da memória.
lookfor – utilizado para localizar comandos, baseado em uma palavra chave.
% - insere um comentário ao arquivo **.m**.

1.5.2 Comandos do sistema operacional

O MATLAB possibilita a execução de alguns comandos do sistema operacional, através de comandos próprios.

dir – mostra o conteúdo do diretório corrente.
what – mostra quais os nomes de arquivos presentes que tem a extensão .m e podem ser executados pelo MATLAB.
cd – muda o diretório corrente.
type – mostra na área de trabalho o conteúdo de um arquivo.

1.6 ERROS NO MATLAB

No MATLAB os comandos e operações são avaliados e o resultado é posto na área de trabalho, caso o usuário digite um comando que não exista, ou uma expressão que não faça sentido para o MATLAB, o mesmo envia para área de trabalho uma mensagem de erro referente a operação que o usuário executou.

Esta mensagem de erro é um comentário especificando o erro cometido e, normalmente, seguido da expressão digitada erroneamente pelo usuário. Alguns erros são comuns no MATLAB e vale a pena lista-los:

```
??? Undefined function or variable xxx.
```

Este erro é causado devido a digitação de um nome de programa, função ou variável que não existe no MATLAB, no caso de ser um programa o mesmo pode não estar no diretório corrente (*Path*).

```
??? 23+4#
      |
Missing operator, comma, or semi-colon.
```

Este erro ocorre quando o usuário digita um símbolo ou operador que não faz sentido para o MATLAB. Neste caso o MATLAB aponta para onde está localizado o caractere inválido.

```
??? a+b*(2+5)-(3+5*(5-9)
                  |
A closing right parenthesis is missing.
Check for a missing ")" or a missing operator.
```

Este erro significa que algum parêntese está aberto errado ou está faltando na expressão como um todo. Existem outras mensagens de erro que o usuário irá conhecer ao se familiarizar melhor com o *software*.

1.7 HELP

Todos os comandos do MATLAB estão presentes no *help* do mesmo. Todos os comandos possuem uma sintaxe que deve ser obedecida, e um número de parâmetros que

deve ser utilizado. O comando “help comando” mostra na área de trabalho um resumo de como se deve operar com o comando especificado.

O *help* é muito útil para obtenção de informações sobre comandos que venham a ser utilizados durante um trabalho.

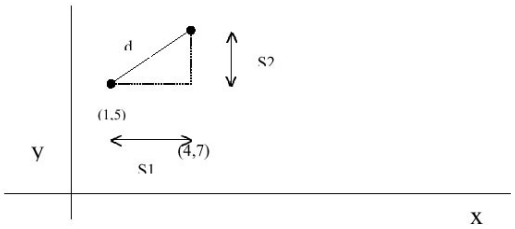
1.8 PRIMEIRO EXEMPLO

Suponha que os pontos p1 e p2 tenham as seguintes coordenadas:

p1= (1,5), p2= (4,7)

Queremos calcular a distância entre os dois pontos, que é a hipotenusa de um triângulo retângulo, conforme mostra a figura a seguir. Usando o Teorema de Pitágoras, podemos calcular a distância *d* com a seguinte equação:

$$d = \sqrt{s1^2 + s2^2}$$
$$d = \sqrt{(4-1)^2 + (7-5)^2}$$
$$d = \sqrt{13}$$
$$d = 3{,}61$$



No próximo capítulo, falaremos sobre os comandos MATLAB. Contudo, da solução você pode ver que os comandos são muito similares às equações usadas durante a execução de um cálculo manualmente. O sinal de percentagem é usado para anteceder comentários que explicam os comandos MATLAB.

```
%
% Este programa calcula e imprime a
% distância, em linha reta, entre dois pontos.
p1 = [1,5]; % ponto 1 inicial
p2 = [4,7]; % ponto2 inicial
d = sqrt (sum ((p2-p1).^2)) % calcular distância
```

O passo final em nosso processo de solução de problemas é testar a solução. Quando os comandos MATLAB, na solução, são executados, o computador mostra a seguinte saída:

```
d = 3.6056
```

CAPÍTULO 2

2 MATRIZES, VETORES E ESCALARES

A capacidade de visualização dos dados é um fator importante na solução de problemas de engenharia. *Matriz* é uma tabela de números dispostos em m linhas e n colunas. Assim, um simples número pode ser considerado uma matriz com uma linha e uma coluna, uma coordenada x-y pode ser considerada uma matriz com uma linha e duas colunas, e um grupo de quatro coordenadas x-y-z pode ser considerada uma matriz com quatro linhas e três colunas. Como exemplo, temos:

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

Se uma matriz contiver m linhas e n colunas, então conterá um total de m,n elementos. Cada elemento da matriz é indicado por índices, aij. O primeiro, i, indica a linha, o segundo, j, indica a coluna onde o elemento se encontra. Assim, o elemento a12 da matriz B é 3.1. Se o número de linhas e colunas forem iguais, então dizemos que a matriz é uma matriz quadrada. Se a matriz tiver apenas uma linha e uma coluna, podemos dizer que o valor é um escalar, se a matriz contiver apenas uma linha ou uma coluna, ao matriz é chamada vetor-linha ou vetor-coluna, respectivamente.

2.1 DEFININDO MATRIZES NO MATLAB

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:

Primeiro modo

Modo mais simples:

Nome da matriz = [a11 a12 a13 …a1n ; a21 a22 a23 … a2n ; … ; am1 am2 am3 … amn];

Assim, as matrizes A, B e C serão representadas por:

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

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.

Quando definimos uma matriz, o MALTAB imprime o valor da matriz na próxima linha a menos que coloquemos um ponto-e-vírgula depois da definição. Tente entrar com as matrizes A, B e C sem o ponto-e-vírgula.

Segundo modo

Você também pode definir uma matriz digitando uma cada linha separadamente. Como exemplo, a matriz C:

```
» C = [ -1 0 0
1 1 0
1 –1 0
0 0 2];
```

Terceiro modo

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.

Quarto modo

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];
```

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

```
» S (2) = -1.0;
```

muda o segundo valor da matriz S de 1.5 para –1.0. A ordem da matriz 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.

2.2 OPERADOR DOIS PONTOS (:)

Suponha que queiramos armazenar a primeira coluna da matriz data1 em um vetor x, e a segunda coluna em um vetor y. O uso do operador dois pontos (:) é útil na criação de matrizes ou vetores. Dependendo do argumento, pode significar todas as linhas ou todas as colunas da matriz-referência. Para o nosso exemplo, temos:

```
» data1 = [0.0,0.0; 0.1 0.2; 0.3 0.6];
» x = data1 ( : , 1);
» y = data1 ( : , 2 );
```

Os elementos do vetor x correspondem à primeira coluna de data1. O segundo comando cria um vetor y cujos elementos correspondem à segunda coluna da matriz data1.

Se quiséssemos criar um vetor z cujos elementos sejam os elementos da primeira linha da matriz data1, devemos fazer:

```
» z = data1(1, : );
```

Se o operador dois pontos for usado na seguinte notação:

```
» H = 1 : 8;
```

A matriz H contém os valores 1, 2, 3, 4, 5, 6, 7 e 8. O operador " : " entre os dois números inteiros gera todos os inteiros entre os dois números especificados. Se for usado para separar três números, os dois pontos gerarão valores entre o primeiro e terceiro números, usando o segundo número como incremento. A notação abaixo gera um vetor-linha denominado TEMPO que contém os números de 0.0 à 5.0 com incrementos de 0.5:

```
» TEMPO = 0.0 : 0.5 : 5.0;
```

O incremento também pode ser um valor negativo como:

```
» VALORES = 10 : -1: 0;
```

Os elementos de VALORES são 10, 9, 8, 7, 6, … 0. O operador dois pontos pode também ser usado para selecionar uma sub-matriz de uma outra matriz. Por exemplo, considere a matriz abaixo:

```
» C = [-1,0,0;1,1,0; 1,-1,0; 0,0,2];
```

Se executarmos os comandos:

```
» PARTE_1 = C ( : , 2:3);
» PARTE_2 = C (3:4, 1:2);
```

Definimos as matrizes:

```
PARTE_1 = [ 0 0; 1 0; -1 0; 0 2];
PARTE_2 = [1 –1; 0 0];
```

E x e r c í c i o s

Determine as ordens e o conteúdo das matrizes abaixo. Use a matriz G como referência.

$$G = \begin{bmatrix} 0,6 & 1,5 & 2,3 & -0,5 \\ 8,2 & 0,5 & -0,1 & -2,0 \\ 5,7 & 8,2 & 9,0 & 1,5 \\ 0,5 & 0,5 & 2,4 & 0,5 \\ 1,2 & -2,3 & -4,5 & 0,5 \end{bmatrix}$$

Verifique suas respostas usando o MATLAB.

1. A = G (:, 2);
2. B = G (4, :);
3. C = [10 : 15];
4. D = [4:9; 1:6];
5. E = [-5,5];
6. F = [0.0:0.1:1.0];
7. T1 = G (4:5,1:3);
8. T2 = G (1:2:5, :);

2.3 COMANDO SIZE

O comando size retorna a dimensão da matriz. Por exemplo:

```
» [n,m]=size(G)
n=
    5
m=
    4
```

2.4 COMANDO INPUT

Você pode entrar com os valores da matriz, via teclado, utilizando o comando input que mostra um texto e então espera-se por uma entrada. Considere o comando:

```
» z = input ( ‘Valores de z: ‘);
```

Quando este comando é executado, o texto “ Valores de z: ” é mostrado na tela. O usuário pode entrar com uma expressão como [5.1 6.3 -18.0] o qual especifica valores para o vetor z. Já que o comando input termina com um ponto-e-vírgula, os valores de z não são imprimidos quando o comando é executado.

2.5 COMANDO *FORMAT*

Suponha os comandos abaixo:

```
» a = [1 2 3];
» c = 2*a
```

```
» T = [ 1.1 2.4 3.7];
» U = 2*T
```

```
c =                                U =
   2 4 6                              2.2000 4.8000 7.4000
```

Por definição, se o elemento de uma matriz for um número inteiro, o MATLAB apresenta o resultado como número inteiro. Se o elemento for um número real, o MATLAB apresenta-o com cinco dígitos significativos, ou seja, quatro dígitos à direita do ponto decimal. Podemos alterar o formato numérico utilizando o comando format.

Exemplo: Seja uma variável A que armazene a raiz quadrada de 2.

```
» A = sqrt(2)
```

De acordo com o formato numérico escolhido, a variável A pode estar apresentada sob a forma:

Comando MATLAB	*Variável A*	*Descrição*
Format long	1.41421356237310	16 dígitos
Format short	1.4142	5 dígitos – formato numérico padrão
Format short e	1.4142e+000	5 dígitos - notação científica
Format long e	1.414213562373095e+000	16 dígitos – notação científica
format +	+	"+" para valores positivos e "-" para valores negativos
format rat	1393/985	aproximação racional
format hex	3ff6a09e667f3bcd	formato hexadecimal

2.6 COMANDO *DISP*

Quando quisermos exibir o conteúdo de uma matriz sem imprimir seu nome ou imprimir um pequeno texto, usamos o comando *dis*p. Assim, se a variável *temp* contiver um valor de temperatura em graus Celsius, podemos imprimir o valor em uma linha de comando e a unidade na linha posterior:

```
disp(temp); disp ('graus Celsius')
```

Se o valor de *temp* for 78, então a saída será:

```
78 graus Celsius
```

2.7 COMANDO *FPRINTF*

O comando *fprintf* nos permite imprimir textos e conteúdo de matrizes. Podemos também especificar o formato numérico . Sua forma geral é:

```
fprintf (formato, matriz)
```

O modo formato contém o texto e as especificações que são:

%e indica que os valores da matriz serão impressos em notação exponencial

%f indica que os valores da matriz serão impressos em notação decimal ou em notação fixa, isto é, o usuário pode especificar o número de algarismos significativos juntamente com o ponto decimal.

%g pode indicar as duas formas acima, dependendo de qual delas será a mais curta.

O modo matriz denota a variável cuja matriz está armazenada.

Um simples exemplo de aplicação do comando fprintf é mostrado abaixo:

```
» fprintf ('A temperatura é %f graus Celsius \n', temp)
```

A saída seria:

```
A temperatura é 78.0000 graus Celsius
```

Se modificarmos o comando para esta forma:

```
» fprintf ('A temperatura é \n %f graus Celsius \n', temp)
```

Então, a saída seria:

```
A temperatura é
78.0000 graus Celsius
```

Os formatos específicos %f, %e, e %g também podem conter informação para especificar o número de casas decimais a imprimir e o número de algarismos significativos, juntamente com o ponto decimal, conforme explicado no início da seção. Considere o seguinte comando:

```
» fprintf ('A temperatura é %4.1f graus Celsius \n', temp)
```

A saída mostrará o valor de *temp* com 4 algarismos, sendo que um destes será um ponto decimal, conforme mostramos abaixo:

```
A temperatura é 78.0 graus Celsius
```

2.8 GRÁFICOS X-Y

Suponhamos que queremos plotar os valores de uma matriz em vez de imprimi-los. Podemos usar o MATLAB para plotar gráficos. Nesta seção, mostraremos como gerar um simples gráfico x-y de dados armazenados em dois vetores. Então, sem conhecer alguns comandos, você pode imediatamente começar usando o MATLAB para gerar gráficos.

Suponha que queira plotar os dados de temperatura a seguir coletados em uma experiência de física:

Tempo, s	Temperatura, °C
0	54.2
1	58.5
2	63.8
3	64.2
4	67.3
5	71.5
6	88.5
7	90.1
8	90.6
9	89.5
10	90.4

Suponha também que os dados relativos ao tempo estejam armazenados em um vetor denominado x, e que os relativos à temperatura estejam armazenados em um vetor denominado y. Para plotar estes pontos, simplesmente usamos o comando *plot* , onde *x e y* são vetores-linha ou vetores-coluna.

```
plot (x, y)
```

O gráfico é gerado automaticamente. A prática mostra que um bom gráfico deve incluir unidades, título e uma breve descrição. Logo, podemos aperfeiçoá-lo como os seguintes comandos:

Title Adiciona um título ao gráfico.

Xlabel Inclui uma descrição na direção do eixo-x

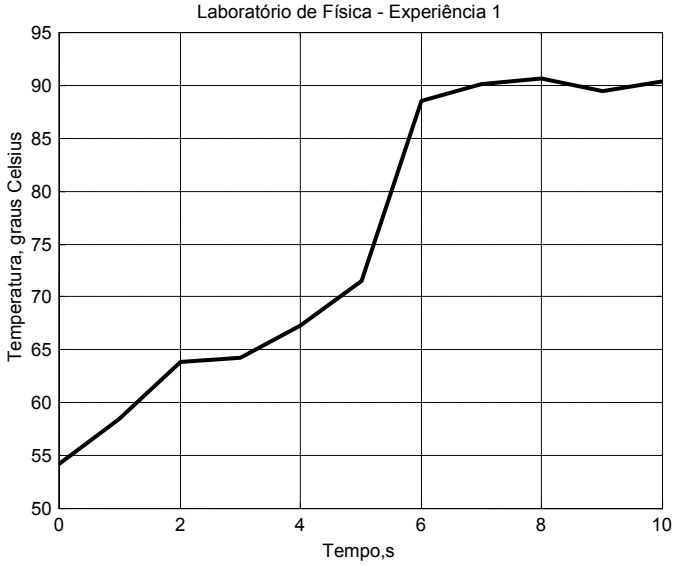
Ylabel Inclui uma descrição na direção do eixo-y

Grid Adiciona linhas de grade ao gráfico

Whitebg Muda a cor de fundo do gráfico para branco.

Assim,

```
» plot (x,y), …
» title (‘Laboratório de Física - Experiência 1’), …
» xlabel (‘Tempo, s’), …
» ylabel (‘Temperatura, graus Celsius’), …
» grid
» whitebg
```



A virgula e os três pontos usados depois dos quatro comandos são usados para que o MATLAB execute os seis comandos em uma única vez. Para aprender mais opções para gerar gráficos x-y.

CAPÍTULO 3

3 CÁLCULOS FUNDAMENTAIS E MATRIZES ESPECIAIS

As operações de adição, subtração, multiplicação e divisão são a maioria das operações fundamentais usadas por engenheiros e cientistas. Podemos executar outras operações de rotina, como o cálculo da raiz quadrada ou o logaritmo de um valor ou a tangente de um ângulo. Estas operações podem ser executadas sobre um valor simples (um escalar), aplicadas a uma lista de valores (vetor), ou aplicadas a um grupo de valores armazenados em uma matriz. Neste capítulo aprenderemos como executar todas estas operações e funções. E também, aprenderemos como usar números complexos no MATLAB.

3.1 VALORES ESPECIAIS E MATRIZES ESPECIAIS

O MATLAB contém um grupo de constantes pré-definidas, valores e matrizes especiais úteis para uso em programas do MATLAB.

3.1.1 Valores especiais

π	**pi**	O valor de π é automaticamente armazenado nesta variável.
$\sqrt{-1}$	**i , j**	Estas variáveis são inicialmente agrupadas ao valor √-1.
∞	**Inf**	Esta variável representa no MATLAB o valor infinito.
Not-a-number	**NaN**	Ocorre em grande parte quando a expressão é indefinida, como a divisão por zero.
	Clock	Exibe a hora atual em um vetor linha de seis elementos, contendo ano, mês, dia, hora minuto e segundos.
	Date	Exibe a data atual como por exemplo, 06-May-2002.
	ans	Variável utilizada para armazenar valores resposta de expressão sem variável.

3.1.2 Matrizes especiais

O MATLAB contém um grupo de funções que geram matrizes especiais. Algumas destas matrizes tem aplicação específica às técnicas numéricas discutidas posteriormente.

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. Forma Geral:

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. Forma Geral:

eye(n) - gera uma matriz identidade de ordem n.

eye (m,n) - gera uma matriz de ordem m x n.

3.2 OPERAÇÕES ENTRE ESCALARES

Cálculos aritméticos são identificados usando expressões. Uma expressão pode ser tão simples como uma constante, ou pode ter matrizes e constantes combinadas com operações aritméticas. Nesta seção, discutiremos operações envolvendo somente escalares. Na seção posterior, estendemos as operações incluindo operações elemento por elemento entre escalares e matrizes ou entre duas matrizes.

As operações aritméticas entre dois escalares são mostradas na tabela a seguir.

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	$\frac{a}{b}$	a/b
Divisão Esquerda	$\frac{b}{a}$	a\b
Exponenciação	a^b	a^b

Uma expressão pode ser resolvida e armazenada em uma variável específica, como no comando seguinte, o qual especifica que os valores em *a* e *b* serão adicionados, e a soma armazenada em uma variável x:

```
» x = a + b
```

Este comando deve ser interpretado como o valor em *b* adicionado ao valor em a, e a soma é armazenado em x. Se nós interpretamos os comandos desta forma, então nós preocupamos pelo seguinte comando MATLAB válido.

```
» count = count + 1
```

É óbvio que esta instrução não é um comando algébrico válido, mas o MATLAB explica que 1 é adicionado ao valor em *coun*t, e o resultado será armazenado nesta variável. Ou seja, o valor em *count* será acrescido de 1 (ou incrementado por 1).

É importante reconhecer que uma variável pode armazenar somente um valor por vez. Por exemplo, suponha que as seguintes instruções serão executadas seguidamente;

```
» Time = 0.0
» Time = 5.0
```

O valor 0.0 é armazenado na variável time quando a primeira instrução é executado e então substituído pelo valor 5.0 quando a segunda instrução é executada. Quando você entra com uma expressão sem especificar uma variável para armazenar o resultado, o mesmo é automaticamente armazenado em uma variável denominada *an*s. Cada vez que um novo valor é armazenado em *an*s, o valor anterior é perdido.

3.2.1 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 3.2 contém a ordem de prioridade das operações aritméticas no MATLAB. Note que esta prioridade também segue a prioridade algébrica padrão.

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

Uma maneira fácil para ter certeza que os cálculos são feitos na ordem que você quer é adicionar parênteses extras. Se uma expressão é longa, divida-a em várias expressões. Por exemplo, considere a seguinte equação:

$$f = \frac{x^3 - 2x^2 + 6{,}3}{x^2 + 0{,}5005x - 3{,}14}$$

O valor de *f* poderia ser calculado usando os seguintes comandos, onde x é um escalar:

```
» numerador = x^3 - 2*x^2 + 6.3
» denominador = x^2 + 0.5005*x - 3.14
» f = numerador/ denominador
```

É melhor usar várias equações que são mais fáceis de compreender que apenas uma, que requer maior cuidado na hora de imaginar a ordem das operações.

3.3 OPERAÇÕES DE CONJUNTOS

Uma operação de conjunto é uma operação elemento por elemento. Por exemplo, suponha que A e B sejam vetores-linha com cinco elementos. Um modo de gerar um novo vetor C com valores que sejam produtos dos valores correspondentes em A e B é o seguinte:

```
» C(1) = A(1)*B(1);
» C(2) = A(2)*B(2);
» C(3) = A(3)*B(3);
» C(4) = A(4)*B(4);
» C(5) = A(5)*B(5);
```

Estes comandos são essencialmente comandos escalares porque cada comando multiplica um simples valor por um outro e armazena o produto em um terceiro valor. Para indicar que executamos uma multiplicação elemento por elemento entre duas matrizes de mesma ordem, usamos um ponto antes da operação. Assim, os cinco comandos acima podem ser substituídos pelo seguinte:

```
» C = A .*B;
```

Se omitirmos o ponto estaremos executando uma operação matricial. Operações matriciais é o tema que será discutido em outro capítulo. Para as operações de adição e subtração, as operações de conjunto e matriciais são idênticas, e então não precisamos distinguí-las. Contudo, as operações de conjunto para multiplicação, divisão e exponenciação são diferentes das operações matriciais para multiplicação, divisão e exponenciação e por isso devemos usar o ponto quando queremos especificar uma operação de conjunto.

Uma operação elemento por elemento, ou operações de conjuntos, aplicam-se não somente para operações entre duas matrizes de mesma ordem como também em operações entre um escalar e um não escalar. Contudo, a multiplicação de uma matriz por um escalar e a divisão esquerda de uma matriz por um escalar podem ser escritas de um modo ou de outro. Assim, os dois comandos em cada grupo de comandos abaixo são equivalentes para uma matriz não escalar A.

```
B = 3*A;
B = 3.*A;
C = A/5;
C = A ./5;
```

As matrizes resultantes B e C terão a mesma ordem de A.

3.4 FUNÇÕES ELEMENTARES

As expressões aritméticas raramente requerem outros cálculos que não sejam a adição, subtração, multiplicação, divisão, e exponenciação. Por exemplo, muitas expressões requerem o uso de logaritmos, exponenciais, e funções trigonométricas. O MATLAB nos

permite usar funções para executar estes tipos de cálculos em vez de nos exigirem calculá-los usando operações aritméticas básicas. Por exemplo, se quisermos calcular o seno de um ângulo e armazenar o resultado em b, podemos usar o seguinte comando:

```
» b = sin(angle);
```

A função *sin* supõe que o argumento está em radianos. Se o argumento contém um valor em graus, podemos convertê-lo de graus para radianos dentro da função referência:

```
» b = sin (angle*pi/180);
```

Poderíamos também fazer a conversão em comandos separados:

```
» angle_radians = angle*pi/180;
» b = sin(angle_radians);
```

Estes comandos são válidos se angle é um escalar ou se angle é uma matriz. Se angle for uma matriz, então a função será aplicada elemento por elemento aos valores na matriz.

Agora que já vimos vários exemplos de funções, iniciaremos uma revisão das regras relativas às funções. Uma função é uma referência que representa uma matriz. Os argumentos ou parâmetros da função estão contidos em parênteses seguindo do nome da função. Uma função pode não conter argumentos, um argumento ou muitos argumentos, dependendo de sua definição. Por exemplo, pi é uma função que não tem argumento; quando usamos a função referência pi, o valor para pi automaticamente substitui a função referência. Se uma função contém mais que um argumento, é muito importante dar os argumentos em ordem correta.

Algumas funções também exigem que os argumentos estejam em unidades específicas. Por exemplo, as funções trigonométricas supõem que os argumentos estão em radianos. No MATLAB, algumas funções usam o número de argumentos para determinar a saída da função. Por exemplo, a função zeros pode ter um ou dois argumentos, pelos quais determinamos a saída.

Uma função referência não pode ser usada ao lado esquerdo de um sinal de igualdade, desde que este represente um valor e não uma variável. Funções podem aparecer à direita de um sinal de igualdade e em expressões. Uma função de referência pode também ser parte do argumento de uma outra função de referência. Por exemplo, o seguinte comando calcula o logaritmo do valor absoluto de x:

```
» log_x = log(abs(x))
```

Quando uma função é usada para calcular o argumento de uma outra função, tenha certeza de fechar o argumento de cada função em seu próprio grupo de parênteses. Esta acomodação da função é também chamada composição de funções. Nomes de funções devem estar em letras minúsculas.

3.4.1 Funções Matemáticas Elementares

As funções matemáticas elementares incluem funções para executar um número de cálculos comuns como o cálculo de valor absoluto e a raiz quadrada. Além disso, também

incluímos um grupo de funções usadas em arredondamentos. Mostraremos a seguir uma lista destas funções com uma breve descrição:

abs (x) Calcula o valor absoluto de x.

sqrt(x) Calcula a raiz quadrada de x.

round(x) Arredonda o valor de x para o inteiro mais próximo.

fix(x) Arredonda o valor de x para o inteiro mais próximo de zero.

floor(x) Arredonda o valor de x para o inteiro mais próximo de $-\infty$

ceil(x) Arredonda o valor de x para o inteiro mais próximo de $+\infty$

sign(x) Se x é menor que zero, a função retorna ao valor – 1; se x for igual a zero, retorna ao valor zero; caso contrário, a função retorna ao valor 1.

rem(x,y) Retorna o resto da divisão $\frac{x}{y}$. Por exemplo, rem (25,4) é 1, e rem (100,21) é 16.

exp(x) Esta função retorna ao valor de e^x, onde e é a base para logaritmo natural ou aproximadamente 2.718282.

log(x) Retorna $ln(x)$, o logaritmo natural de *x* para a base e.

log10(x) Retorna $log_{10} x$, ou seja, o logaritmo de *x* na base 10.

3.4.2 Funções Trigonométricas

As funções trigonométricas supõem que os ângulos estejam representados em radianos. A seguir uma lista de funções trigonométricas com uma breve descrição:

sin(x) Calcula o seno de x, em radianos.

cos(x) Calcula o cosseno de x, em radianos.

tan(x) Calcula a tangente de x, em radianos.

asin(x) Calcula o arcoseno de x, onde x deve estar entre –1 e 1. A função apresenta um ângulo em radianos entre - pi/2 e pi/2.

acos(x) Calcula o arcocosseno de x, onde x deve estar entre –1 e 1. A função apresenta um ângulo em radianos entre 0 e pi.

atan(x) Calcula o arcotangente de x, onde x deve estar entre –1 e 1. A função apresenta um ângulo em radianos entre - pi/2 e pi/2.

atan2(x,y) Calcula o arcotangente do valor de y/x. A função apresenta um ângulo em radianos estará entre – pi e pi, dependendo dos sinais de x e y.

As outras funções trigonométricas podem ser calculadas usando as seguintes equações:

sec x = 1/ cos x

csc x = 1 / sin x

cot x = 1 / tan x

3.4.3 Funções Hiperbólicas

sinh(x) Calcula o seno hiperbólico de x.

cosh(x) Calcula o cosseno hiperbólico de x.

tanh(x) Calcula a tangente hiperbólica de x.

asinh(x) Calcula o seno hiperbólico inverso de x.

acosh(x) Calcula o cosseno hiperbólico inverso de x.

atanh(x) Calcula a tangente hiperbólica inversa de x.

3.4.4 Funções de arquivo M

O MATLAB apresenta uma estrutura que nos permite criar funções sob a forma de arquivos M. Como exemplo, considere uma função que esteja em um arquivo-M denominado circum.m:

```
function c = circum( r)
% CIRCUM Circunferência de um círculo de raio r.
% Para matrizes, CIRCUM ( r ) retorna uma matriz
% que contêm as circunferências de círculos com raios iguais
% aos valores no vetor original.
c = pi*2*r;
```

Assim, se o prompt do MATLAB apresentar:

```
» r = [0 1.4 pi];
» a = circum(r);
```

Os elementos da matriz A corresponderão as circunferências de círculos de raios 0, 1.4 e pi, respectivamente.

Para esta função também são válidos os comandos:

```
» a = 5.6;
» disp (circum(a))
» c = [1.2 3; 5 2.3];
» circum (c) ;
```

Assim, *circum* passa a ser uma função MATLAB assim como *one*s, *sin* e outras. A parte comentada no arquivo circum.m é usada quando digitarmos help circum no prompt do MATLAB.

3.5 NÚMEROS COMPLEXOS

As soluções de muitos problemas de engenharia como sistema de controle para um braço mecânico ou análise da estabilidade de um circuito elétrico envolvem a busca das raízes de uma equação da seguinte forma:

$y = f(x)$

onde as raízes são os valores de x para qual y é igual a zero.

Considere a forma geral para um polinômio de grau n:

$a_1x^n + a_2x^{n-1} + a_3x^{n-2} + \ldots + n_{a-1}x^2 + a_nx + a_{n+1} = 0$

Um polinômio de grau n terá n raízes, sendo que algumas podem ser raízes múltiplas ou raízes complexas. Nesta seção discutiremos as operações com números complexos e as funções MATLAB que os usam.

3.5.1 Operações Aritméticas com Números Complexos

Os comandos MATLAB reconhecem os números complexos usando i para representar √-1. O MATLAB também reconhece o uso de j para representar √-1. Esta notação é mais usada na Engenharia Elétrica. O comando a seguir define uma variável complexa:

```
» x = 1 - 0.5*i;
```

Quando executamos operações entre dois complexos, o MATLAB automaticamente executa os cálculos necessários. Se uma operação for entre um número real e um complexo, o MATLAB supõe que a parte imaginária do número real é igual a zero. O MATLAB inclui várias funções que são específicas aos números complexos:

real (x) Calcula a parte real do número complexo x.

imag(x) Calcula a parte imaginária do número complexo x.

conj(x) Calcula o conjugado do número complexo x.

abs(x) Calcula o módulo do número complexo x.

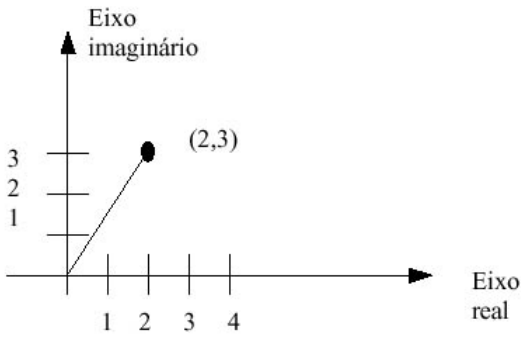
angle(x) Calcula o ângulo usando o valor de atan2 (imag(x), real(x)), e portanto o ângulo está entre –pi e pi.

Estas funções tornam mais fácil converter o complexo da forma polar para retangular.

3.5.2 Coordenadas polar e retangulares

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 a seguir 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 raio3,6.



Conversão

- retangular a polar

$$r = \sqrt{a^2 + b^2}$$

$$\theta = \tan^{-1}\left(\frac{b}{a}\right)$$

- polar a retangular

$$a = r\cos\theta$$

$$b = r\,\text{sen}\,\theta$$

Se x é um número complexo, então o módulo e a fase podem ser calculados com os seguintes comandos:

```
» r = abs(x);
» theta = angle(x);
```

Para calcular o número complexo usando módulo e fase determinados, usamos o comando:

```
» y = r*exp (i*theta);
```

Podemos calcular a parte real e a parte imaginária de um complexo com os comandos:

```
» a = real (x);
» b = imag (x);
```

CAPÍTULO 4

4 CONTROLE DE FLUXO

4.1 OPERADORES LÓGICOS E RELACIONAIS

4.1.1 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].

4.2 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 *o*u, 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 "0" (falso) para todos resultados diferentes. Além disso, a operação só será válida se as matrizes resultantes (a < b e b < c) tiverem o mesmo tamanho.

4.3 TOMADA DE DECISÕES

4.3.1 Estrutura If – Else – End

```
if expressão
   Comandos
end
```

Se a expressão lógica for verdadeira, os comandos entre *if* e *end* são executados. Como exemplo, temos:

```
if a < 50
   count = count +1;
   sum = sum + a;
end
```

Suponha que *a* seja um escalar. Se *a* < 50, então count é incrementada por 1 e *a* é adicionada à sum; caso contrário, os comandos não serão executados. A próxima estrutura contém um parâmetro *if* dentro de outro parâmetro *if*:

```
if expressão 1
   grupo de comandos A
      if expressão 2
         grupo de comandos B
      end
   grupo de comandos C
end
grupo de comandos D
```

Se a expressão 1 for verdadeira, os grupos de comandos A e C são executados. Se a expressão 2 também for verdadeira, o grupo de comandos B é executado antes do grupo de comandos C. Se a expressão 1 for falsa, imediatamente salta-se para os comandos D. Como exemplo, temos:

```
if a < 50
   count = count + 1
   sum = sum + a;
   if b > a
```

```
      b = 0;
   end
end
```

Novamente, suponha que *a* e *b* sejam escalares. Então, se a < 50 aumentaremos *count* por 1 e adicionaremos *a* à *su*m. Se b > a, então b será igual a zero. Se *a* não for menor que 50, então pula-se diretamente para o segundo end. Se a e nem b forem escalares, então b é maior que a somente se cada par de elementos correspondentes de a e b são valores nos quais b > a. Se a ou b é um escalar, então a matriz é comparada ao escalar.

Instrução *Else*

Esta instrução permite que executemos um comando se a expressão lógica é verdadeira e um diferente comando se a expressão é falsa. A forma geral do comando *if* combinada à instrução *else* é mostrada a seguir:

```
if expressão
   grupo de comandos A
else
   grupo de comandos B
end
```

Se a expressão lógica é verdadeira, então o grupo de comandos A é executado. Caso contrário, o grupo de comandos B é executado.

Quando há muitas alternativas a serem executadas, pode ser mais difícil determinar quais expressões lógicas devem ser verdadeiras (ou falsas) para executar cada grupo de comandos. Neste caso, a cláusula elseif é freqüentemente usada para simplificar o programa lógico:

```
if expressão 1
   grupo de comandos A
elseif expressão 2
   grupo de comandos B
elseif expressão 3
   grupo de comandos C
end
```

Se a expressão 1 for verdadeira, somente o grupo de comandos A é executado. Se a expressão 1 for falsa e a expressão 2 for verdadeira, então somente o segundo grupo de comandos é executado. Se as expressões 1 e 2 forem falsas e a expressão 3 for verdadeira, então somente o grupo de comandos C é executado. Se mais de uma expressão lógica for verdadeira, a primeira que for verdadeira determina qual grupo de comandos será executado. Se nenhuma das expressões lógica for verdadeira, então nenhum dos comandos dentro da estrutura if é executado.

4.4 LOOP FOR

O MATLAB contém dois comandos para gerar loops, o comando *for* e o comando *while*. Nesta seção, discutiremos o comando *for*, e na próxima seção discutiremos o comando *while.*

O comando *for* tem a estrutura a seguir:

```
for variável = expressão
   Grupo de comandos A
end
```

Os comandos entre as instruções *for* e *end* são executados uma vez para cada coluna da expressão matricial. A cada iteração, a variável é atribuída para a próxima coluna da matriz, isto é, durante o i-ésimo ciclo do loop, temos que variável = expressão matricial (: , i). Veja o exemplo a seguir:

Suponha que temos um grupo de valores que representam a distância de um táxi à torre mais próxima. Queremos gerar um vetor que contenha as respectivas velocidades. Se o táxi está a 10 metros do edifício, usamos a equação:

$$\text{velocidade} = 0{,}425 + 0{,}00175\text{d}^2$$

Se o táxi estiver a uma distância maior que 10 metros, use a equação a seguir:

$$\text{velocidade} = 0{,}625 + 0{,}12\text{d} - 0{,}00025\text{d}^2$$

Como a escolha da equação da velocidade depende do valor de d, devemos determinar separadamente *d(1*), *d(2*), e assim por diante. Logo, usaremos um loop, conforme descrito a seguir.

```
for d = 1:25
   if d < = 10
      velocidade = 0.425 + 0.00175*d^2
   else
     velocidade = 0.625 + 0.12*d – 0.00025*d^2
end
```

4.5 LOOPS WHILE

O loop *while* é uma importante estrutura para repetição de um grupo de comandos quando a condição especificada for verdadeira. O formato geral para esta estrutura de controle é:

```
while expressão
   grupo de comandos A
end
```

Se a expressão for verdadeira, então o grupo de comandos A é executado. Depois destes comandos serem executados, a condição é novamente questionada. Se for verdadeira, o grupo de comandos é novamente executado. Quando a condição for falsa, o controle pula para o comando posterior ao comando end. As variáveis modificadas no grupo de comandos A devem incluir as variáveis na expressão, ou o valor da expressão nunca será

mudado. Se a expressão for verdadeira (ou é um valor não-nulo), o loop torna-se um loop infinito. Lembre-se que você pode usar ^*c* (Ctrl+C) para sair um loop infinito.

Exemplo:

```
x = [ 1 2 3 4 5 6 7 8 9 ];
sum = 0;
k = 1;
while x (k) > =0 & k < size (x,2)
   sum = sum + x(k);
   k = k + 1;
end
```

CAPÍTULO 5

5 OPERAÇÕES COM MATRIZES

Uma matriz é um conveniente meio para representar dados experimentais. Nos capítulos anteriores, nós discutimos cálculos matemáticos e funções que poderiam ser aplicadas elemento a elemento presente nas matrizes. Neste capítulo, nós apresentaremos um conjunto de operações e funções que podem ser aplicadas à matrizes como um todo, ao invés de lidarmos com os elementos individualmente. Vamos primeiro considerar um conjunto de operações matemáticas aplicados à matrizes. E depois vamos considerar um grupo de funções que ajudam na manipulação das matrizes.

5.1 OPERAÇÕES COM MATRIZES

5.1.1 Matrizes Transpostas

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

Exemplo:

$$A = \begin{bmatrix} 2 & 5 & 1 \\ 7 & 3 & 8 \\ 4 & 5 & 21 \\ 16 & 13 & 0 \end{bmatrix} \qquad At = \begin{bmatrix} 2 & 7 & 4 & 16 \\ 5 & 3 & 5 & 13 \\ 1 & 8 & 21 & 0 \end{bmatrix}$$

Podemos notar que o elemento da posição (3,1) foi movido para a posição (1,3). De fato, quando se acha a matriz transposta de uma matriz temos a troca de elementos das posições (i,j) para as posições (j,i).

No MATLAB a matriz transposta é denotada por A'.

Por exemplo:

```
» B=A'
```

5.1.2 Somatório de Produtos

É a soma escalar de dois vetores do mesmo tamanho.

Somatório de produtos = $A.B = \sum_{i=j}^{N} ai\ bi$

Exemplo

A = [4 –1 3] e B = [-2 5 2]

A . B = (4) . (-2) + (-1) . (5) + (3) . (2)

A . B = (-8) + (-5) + (6)

A . B = -7

5.1.3 Multiplicação de Matrizes

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.

Se A tem 2 linhas e 3 colunas, e B tem 3 linhas e 3 colunas, então o produto A.B terá 2 linhas e 3 colunas.

Exemplo

$$A = \begin{bmatrix} 2 & 5 & 1 \\ 0 & 3 & -1 \end{bmatrix} \quad B = \begin{bmatrix} -1 & 0 & 2 \\ -1 & 4 & -2 \\ 5 & 2 & 1 \end{bmatrix}$$

O primeiro elemento do produto C = A.B é (2).(-1) + (5).(-1) + (1).(5) = -2. Logo a matriz C será:

$$C = \begin{bmatrix} -2 & 22 & -5 \\ -8 & 10 & -7 \end{bmatrix}$$

Neste exemplo não se pode ter B.A pois o número de colunas de B não é igual ao número de linhas de A.

No MATLAB podem ser usados os seguintes comandos:

```
» A = [2 5 1;0 3 –1];
» B = [1 0 2;-1 4 –2;5 2 1];
» C = A * B
```

5.1.4 Matriz Inversa

Por definição o inverso de uma matriz quadrada A é a matriz A^{-1}. Se considerarmos duas matrizes A e B:

$$A = \begin{bmatrix} 2 & 1 \\ 4 & 3 \end{bmatrix} \quad B = \begin{bmatrix} 1.5 & -0.5 \\ -2 & 1 \end{bmatrix}$$

Quando calculamos os produtos A.B e B.A e obtemos as matrizes:

$$AB = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad BA = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Temos que as matrizes A e B são inversas, ou seja, $A = B^{-1}$ e $B = A^{-1}$.

No MATLAB, para obtermos uma matriz inversa devemos fornecer a matriz original A e executar o comando *inv(A*).

5.1.5 Determinante

Seja a matriz

$$A = \begin{bmatrix} 1 & 3 \\ -1 & 5 \end{bmatrix}$$

O determinante de A = |A| é definido pela expressão:

$a_{11} \cdot a_{22} - a_{21} \cdot a_{12}$

No MATLAB, o comando utilizado para se achar o determinante de uma matriz é *det(A).*

5.2 MANIPULAÇÕES COM MATRIZES

5.2.1 Comando *rot90*

Uma matriz A pode sofrer uma rotação de 90^0 usando-se o comando *rot9*0.

Exemplo

$$A = \begin{bmatrix} 2 & 1 & 0 \\ -2 & 5 & -1 \\ 3 & 4 & 6 \end{bmatrix}$$

$$B = \text{rot90(A)}$$

$$B = \begin{bmatrix} 0 & -1 & 6 \\ 1 & 5 & 4 \\ 2 & -2 & 3 \end{bmatrix}$$

5.2.2 Comando *fliplr*

Esse comando troca o lado esquerdo com o direito de uma matriz.

5.2.3 Comando *flipud*

Esse comando troca a parte de cima com a parte de baixo de uma matriz.

5.2.4 Comando *diag*

Esse comando extrai os elementos da diagonal principal da matriz A e os coloca em um vetor coluna. Desta forma, temos:

$$A = \begin{bmatrix} -1 & 0 & 2 \\ -1 & 4 & -2 \\ 5 & 2 & 1 \end{bmatrix}$$

B = diag(A)

$$B = \begin{bmatrix} -1 \\ 4 \\ 1 \end{bmatrix}$$

5.2.5 Comando *triu*

Este comando trata uma matriz preenchendo com zeros nos lugares dos antigos elementos localizados abaixo da diagonal principal.

Exemplo

$$A = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 3 & 6 & 9 & 12 \\ 4 & 3 & 2 & 1 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

B = triu(A)

$$B = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 0 & 6 & 9 & 12 \\ 0 & 0 & 2 & 1 \\ 0 & 0 & 0 & 4 \end{bmatrix}$$

5.2.6 Comando *tril*

É similar ao comando triu, porém essa função mantém a matriz da diagonal principal para baixo.

CAPÍTULO 6

6 GRÁFICOS

Engenheiros usam gráficos para analisar e resolver problemas e situações. Por isso é muito importante aprendermos a interpretar e gerar gráficos e suas formas. Neste capítulo vamos aprender como o MATLAB pode nos ajudar a gerar gráficos.

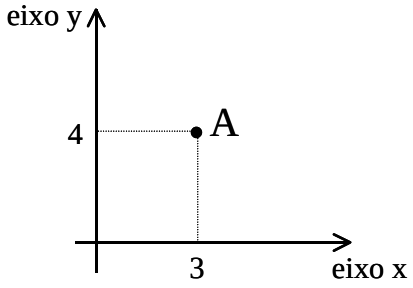
6.1 GRÁFICOS X – Y

É muito comum engenheiros e cientistas usarem gráficos x - y. Os dados que nós plotamos são usualmente lidos por um arquivo ou calculados em nossos programas.

Geralmente assumimos que valores de x representam variáveis independentes e que valores de y representam variáveis dependentes. Os valores de y podem ser calculados usando as funções de x, ou os valores de x e y podem ser retirados de experiência.

6.1.1 Coordenadas Retangulares

Os pontos retangulares identificam os pontos no sistema de coordenadas cartesianas com suas posições ao longo dos eixos horizontal e vertical como na figura a seguir.



Legenda

Os comandos para se adicionar títulos, linhas de grade e inserir textos estão relacionados a seguir:

title('text') Este comando escreve títulos no topo do gráfico plotado.

xlabel('text') Este comando escreve um texto abaixo do eixo x do gráfico plotado.

ylabel('text') Este comando escreve um texto ao lado do eixo y do gráfico plotado.

text(x, y, 'text') Este comando escreve um texto na tela do gráfico no ponto específico das coordenadas (x, y) usando os eixos dos gráficos. Se x e y são vetores o texto é escrito a cada ponto.

text(x, y, 'text', sc) Este comando escreve um texto na tela do gráfico no ponto especificado pelas coordenadas (x, y), assumindo que a esquina esquerda inferior é

(0,0), e a esquina direita superior é (1,1).

gtext(‘text’) Este comando escreve um texto nas posições indicadas na tela do gráfico pelo mouse.

Comandos de plotar

Geralmente assumimos que y e x são eixos divididos com o mesmo intervalo de espaço. Esses gráficos são chamados de lineares. Às vezes temos que usar uma escala logarítmica em um ou ambos os eixos.

Os comandos para plotar gráficos lineares e logarítmicos são:

plot(x,y) Este comando gera gráficos lineares com valores x e y, onde x representa a variável independente e y representa a variável dependente.

semilogx(x,y) Este comando gera gráfico usando escala linear para y e escala logarítmica para x.

semilogy(x,y) Este comando gera gráficos usando escala linear para x e escala logarítmica para y.

loglog(x,y) Este comando gera gráficos com escala logarítmica para ambos os eixos x e y.

6.2 GRÁFICOS POLARES

Gráficos polares são úteis quando valores são representados por ângulo e grandeza (magnitude). Por exemplo se medirmos a intensidade luminosa ao redor de uma fonte de luz, podemos representar a informação com um ângulo fixando eixos e magnitude representando intensidade.

6.2.1 Coordenadas polares

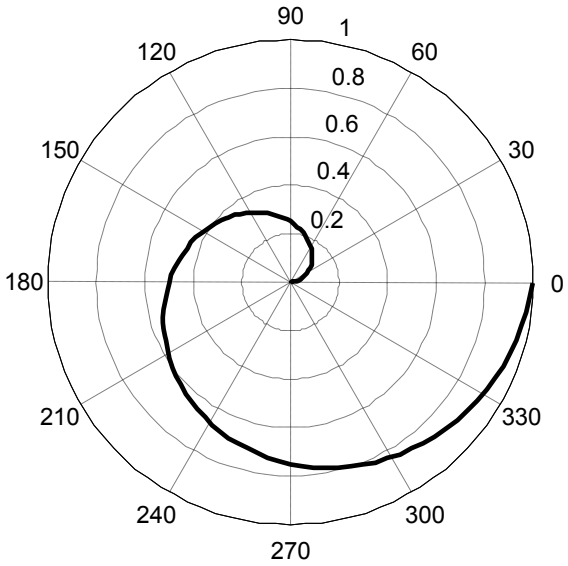
Um ponto é representado em coordenadas polares por um ângulo θ e uma magnitude r. O valor de θ é geralmente dado entre 0 e 2π. A magnitude é um valor positivo que representa a distância do eixo que fornece o angulo até o ponto.

O comando no MATLAB para gerar gráficos polares é:

polar(theta,r) - Este comando generaliza gráficos polares com ângulo θ (em radiano) e magnitude r correspondente.

Exemplo: Os comando para a construção do gráfico da próxima figura;

```
» theta = 0:2*pi / 100 : 2*pi;
» r = theta / (2*pi);
» polar(theta,r);
```



6.3 GRÁFICOS DE BARRAS E DEGRAUS

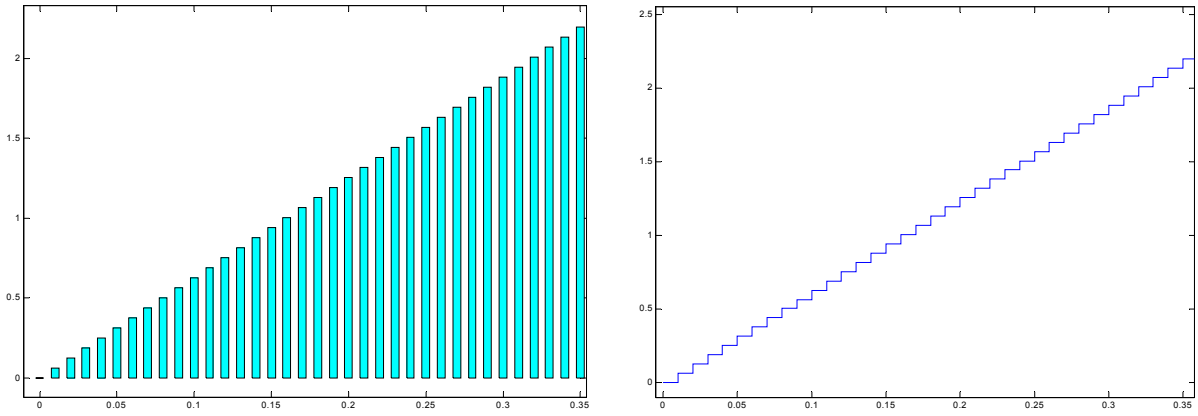
Os gráficos são similares, porém as linhas verticais que marcam o eixo x nos gráficos de barras são omitidas nos gráficos de degraus.

Comandos:

bar(x,y) - Este comando gera gráficos de barras com elementos do vetor y localizados no vetor x, contém o mesmo espaço entre os valores.

stairs(x,y) - Este comando gera um gráfico de degraus com os elementos do vetor y.

Exemplo: a figura a seguir mostra um gráfico de barra e um de degraus;



6.4 OPÇÕES

Gráficos Múltiplos => Para plotar curvas múltiplas no mesmo gráfico deve se usar vários argumentos no comando plotar como a seguir:

```
» plot(x, y, w, z);
```

Quando se executa este comando a curva correspondente a x, y e a curva correspondente a w, z são plotadas no mesmo gráfico. O MATLAB seleciona linhas diferentes para as curvas plotadas.

Outro comando bastante utilizado quando se deseja plotar mais de uma curva em um mesmo gráfico é o comando hold. Por exemplo,

```
» plot(x,y)
» hold on
» plot(w,z)
» hold off
```

Esses comandos efetuarão a mesma tarefa exemplificada anteriormente.

6.4.1 Estilo de linha e marcação

O comando plot(x, y) nos mostra uma linha plotada representando os vetores y e x, mas podemos selecionar outros tipos de linha. Também podemos selecionar plotar pontos ao invés de linhas. A seguir as diferentes opções de linhas e marcações:

Tipo de Linha	Indicador	Tipo de Ponto	Indicador
Solid	-	Point	.
Dashed	--	Plus	+
Dotted	:	Star	*
Dashdot	-.	Circle	o
		x-mark	x

O comando a seguir representa linha sólida com tipo de ponto x-mark

```
» plot(x, y, x, y, 'x')
```

Podemos também escolher as cores que serão usadas:

Cor	Indicadores
Vermelho	r
Verde	g
Azul	b
Branco	w
Invisível	i

O comando seguinte representa linha sólida azul para os vetores x, y e pontos vermelhos do tipo x-mark:

```
» plot(x, y, 'b', x, y, 'xr');
```

6.4.2 Escala

A escala dos eixos no MATLAB é automática, porém se você quiser ajustar a escala de seus eixos você pode usar o comando *axis*. Existem várias formas de se usar o comando *axis*:

axis(v) - v é um vetor de quatro elementos que contém a escala de valores, [xmin,xmax,ymin,ymax].

Esse comando tem um uso especial quando se quer comparar curvas de diferentes gráficos, pois pode ser difícil a comparação quando as curvas possuem diferentes eixos e escalas.

6.4.3 Subplot

O comando subplot é usado quando se quer visualizar dois ou mais gráficos ao mesmo tempo.

```
» Subplot(211), plot(x,y)
» Subplot(212), plot(y,x)
```

Esse comando significa que teremos 2 gráficos sendo o primeiro (plot(x,y)) colocado no canto superior esquerdo da tela e o segundo colocado no canto superior direito da tela.

6.4.4 Controle de tela

gcf Apresenta uma janela com gráfico;

clc Limpa a janela de comando;

clf Limpa a janela do gráfico;

figure Abre nova janela de gráfico (similar ao gcf).

E x e r c í c i o

Gerar 12 pontos de uma função para os valores de x começando de x=0 e incrementando de 0.5;

y = 5x.^2 :

a) Gerar o gráfico linear desta função;

b) Gerar o gráfico desta função com escala logarítmica x;

c) Gerar o gráfico desta função com escala logarítmica y;

d) Gerar o gráfico loglog desta função;