LCEE

Laboratório de Computação da Engenharia Elétrica - UFES

MANUAL DO MATLAB

Curso de MATLAB

Projeto REENGE - DEL

Elaboração/ Redação: Leonardo Pereira Bastos

Revisão: Prof. Celso Munaro

Data: março de 1997

Esta apostila foi elaborada dentro do projeto REENGE/DEL, cujo objetivo é melhorar as condições de ensino do curso de Engenharia Elétrica via suporte computacional adequado.

Índice:

PARTE I:

PARTE II: Informações Avançadas

1. Fatoração Triangular
2. Fatoração Ortogonal
3. Decomposição dos Autovalores

3. Polinômios e Processamento de Sinais

1. Representação Polinomial
2. Processamento de Sinais
3. Filtro de Dados
4. Funções de Função

1. Integração Numérica
2. Equações Não-Lineares e Funções de Otimização
3. Funções de Equações Diferenciais

5. Fluxo de Controle

1. Loop FOR
2. Loop WHILE
3. Comandos IF e BREAK

6. Arquivos-M: Scripts e Funções

1. Arquivos Script
2. Arquivos Função
3. Variáveis Globais
4. Strings de Texto
5. A Função eval

7. Arquivos de Disco

1. Importando e Exportando Dados

1. Importando Dados
2. Exportando Dados do MATLAB

8. O Debugger do MATLAB
9. Arquivos E/S

1. Abrindo e Fechando Arquivos
2. Leitura de Arquivos com Dados em Binário
3. Escrevendo em Arquivos com Dados em Binário
4. Escrevendo Arquivos de Texto Formatados e Strings
5. Lendo Arquivos de Texto Formatados e Strings

PARTE III: Gráficos Avançados

1. Gráficos Avançados

1. Gráficos 2-D

PARTE I

Informações Básicas

Esta primeira parte traz os capítulos de 1 a 6, e contém as informações necessárias para se começar a trabalhar com MATLAB.

Após esta parte o leitor deve estar apto a declarar variáveis no prompt, seja do tipo vetor ou matriz, realizar operações com estas variáveis, utilizar o comando Help e plotar gráficos.

Capítulo 1

Informações Iniciais

Este capítulo possui informações para a instalação do MATLAB versão 4.20 no seu computador.

1.1 Instalação

A instalação do MATLAB no computador é feita da seguinte forma:

Insira o disco 1 e chame o arquivo setup.exe. O programa de instalação o informará sobre o espaço disponível no disco rígido e o espaço necessário para a instalação do programa; caso o espaço disponível seja suficiente para a instalação, será perguntado um nome para o diretório onde o programa será instalado (o nome padrão do diretório é MATLAB).À medida que a instalação for sendo realizada, o programa de instalação pedirá o próximo disco (até o disco 5). No final da instalação, é perguntado se deseja-se instalar alguma toolbox. Responda não. A mesma resposta deve ser dada com respeito aos compiladores.

Quando a instalação terminar, insira o disco 6 e copie o diretório *control* para o caminho *c:/matlab/toolbox/matlab/*. Desta forma, a toolbox de controle ficará disponível no caminho *c:/matlab/toolbox/matlab/control/* Existe um arquivo no diretório principal que define os caminhos que o MATLAB segue (em quais subdiretórios o MATLAB procura os arquivos ou funções necessárias); este arquivo é uma espécie de *autoexec.bat* do MATLAB. Este arquivo é o matlabrc.m; ele pode ser modificado em qualquer editor de texto de modo a definir o caminho para o diretório da toolbox de controle.

Os caminhos são definidos na seguinte parte do arquivo

matlabpath([...

'C:\MATLAB\toolbox\local',...

';C:\MATLAB\toolbox\matlab\control',... ***(linha adicionada)***

';C:\MATLAB\toolbox\matlab\datafun',...

';C:\MATLAB\toolbox\matlab\elmat',...

]); onde a linha que deve ser adicionada está indicada acima.

Obs.: O MATLAB utiliza, geralmente, o Bloco de Notas do Windows para que se possa criar ou alterar arquivos .m. Por isso, quando a instalação do MATLAB terminar, deve-se fazer o seguinte:

- No Windows 3.x

No gerenciador de arquivos, clique uma vez sobre qualquer arquivo com extensão .m e, no menu *Arquivo*, escolha *Associar*. No item *Arquivos com extensão* coloque m (extensão dos arquivos que o MATLAB utiliza). No item *Associar com* escolha o Bloco de Notas. Clique em *OK* e pronto.

- No Windows 95

No Explorer, clique no menu *Exibir* e escolha *Opções*. Escolha em tipos de arquivo o ícone *Novo tipo*. Onde estiver *Descrição do tipo* escreva MATLAB; onde estiver escrito *Extensão associada* escreva m. Em seguida escolha *Novo* e preencha da seguinte forma:

Ações: Abrir

Aplicativo utilizado para executar a ação: c:\windows\notepad.exe

Clique em *OK* até que todas as janelas sejam fechadas.

1.2 Diretório

O diretório MATLAB é composto de vários subdiretórios. Os dois que possuem maior importância a nível de conhecimento do aluno são os descritos abaixo:

/bin	Diretório em que se encontra o arquivo matlab.exe, arquivo de execução do MATLAB.
/toolbox/matlab	Diretório onde estão localizados os subdiretórios de toolboxes (/matlab, /control, etc.).

Capítulo 2

Iniciando

Neste capítulo é apresentado o espaço de trabalho do MATLAB. É mostrado como se lidar com matrizes e como utilizar o comando Help para obter informações de utilização das funções.

2.1 Linha de Comando

Para editar comandos digitados erroneamente ou para chamar linhas de comandos anteriores, pode-se fazer uso das setas. Por exemplo, se foi digitado errado a função sqrt no comando

```
>> log(sqt(atan(2*(3+4))))
```

MATLAB responde com uma mensagem de erro do tipo

```
??? Undefined function or variable sqt.
```

Ao invés de se digitar a linha de comando novamente, tecla-se . O último comando que foi entrado é mostrado. Pode-se teclar para mover o cursor e inserir o r.

Os comandos que MATLAB executou durante uma sessão são armazenados até um certo limite. Pode ser utilizada uma chamada rápida ao invés da digitação dos comandos previamente digitados, através da especificação dos primeiros caracteres seguidos pela tecla . No exemplo abaixo, as letras plo são usadas para encontrar uma linha de comando que inicie com o comando plot executado anteriormente

```
>> plo
```

A tabela abaixo lista as teclas que podem ser usadas para edição na linha de comandos:

Ctrl-P	Chama uma linha de comando anterior.
Ctrl-N	Chama uma linha de comando posterior.
Ctrl-B	Move o cursor um caractere à esquerda.
Ctrl-F	Move o cursor um caractere à direita.
Delete	Move o cursor à esquerda, apagando um caractere.
Ctrl-L	Move o cursor uma palavra à esquerda.
Ctrl-R	Move o cursor uma palavra à direita.
Ctrl-A	Move o cursor para o começo da linha.
Ctrl-E	Move o cursor para o final da linha.
Ctrl-U	Cancela a linha.
Ctrl-D	Apaga o caractere que está sobre o cursor.
Ctrl-K	Apaga até o final da linha.

2.2 Matrizes Simples

MATLAB trabalha essencialmente com um tipo de objeto, uma matriz retangular numérica (real ou complexa). Em algumas situações, denominações específicas são atribuídas a matrizes 1 por 1, que são os escalares, e a matrizes com somente uma linha ou coluna, que são os vetores. Operações e comandos no MATLAB são aplicados de maneira natural, utilizando o conceito de matriz, como são indicados no papel.

Pode-se entrar com matrizes no MATLAB de diversas maneiras:

- Entrar com uma lista explícita de elementos.
- Gerar matrizes utilizando funções e linhas de comando do MATLAB.
- Criar matrizes em arquivos-M.
- Chamar matrizes de um arquivo de dados externo.

A maneira mais fácil de se declarar matrizes é fazendo a explicitação da lista de elementos na linha de comando, seguindo as convenções abaixo:

- Separar os elementos da lista de elementos através de espaços ou vírgulas.
- Colocar os elementos entre colchetes, [ ].
- Usar (;) ponto-e-vírgula para indicar o fim de uma linha.

Por exemplo, entrando com a linha de comando

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

resulta em

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

MATLAB salva a matriz A para que possa ser utilizada posteriormente. Pode-se, também, entrar matrizes de arquivos de extensão .m. Se um arquivo com o nome de matriz.m contém as três linhas de texto

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

então o comando

```
>> matriz
```

atribui à matriz A os dados correspondentes.

2.3 Elementos da Matriz

Elementos de matriz podem ser quaisquer expressões que o MATLAB permite; por exemplo

```
>> x = [-1.3 sqrt(3) (1+2+3)*4/5]
```

resulta em

```
>> x =
-1.3000 1.7321 4.8000
```

Elementos individuais de matrizes podem ser referenciados individualmente com índices dentro de parênteses, ( ). Continuando o exemplo anterior

```
>> x(5) = abs(x(1))
>> x =
-1.3000 1.7321 4.8000 0 1.3000
```

Perceba que o tamanho de x aumenta automaticamente para acomodar o novo elemento, e os elementos indefinidos são assumidos como sendo zero.

Pode-se construir grandes matrizes utilizando matrizes menores como elementos. Por exemplo, para acrescentar uma nova linha à matriz A:

```
>> r = [10 11 12];
```

```
>> A = [A; r]
```
Isto resulta em

```
>> A =
1 2 3
4 5 6
7 8 9
10 11 12
```

Matrizes menores podem ser extraídas de matrizes grandes utilizando os dois pontos (:). Por exemplo

```
>> A = A(1:3,:);
```

que fornece a seguinte matriz

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

faz com que retorne as três primeiras linhas e todas as colunas da matriz original A para a matriz A.

2.4 Linhas de Comando e Variáveis do MATLAB

As linhas de comando do MATLAB frequentemente são da forma

variável = expressão
ou simplesmente
expressão

Podem ser compostas expressões com operadores e outros caracteres especiais, com funções, e com nomes de variáveis. A execução da expressão produz uma matriz. A matriz é mostrada na tela e assume o nome da variável definida na linha de comando para que possa ser utilizada em uma outra situação. Se for omitido o nome da variável e o sinal =, MATLAB cria automaticamente uma variável com o nome ans. Por exemplo, digitando a expressão

```
>> 1900/81
>> ans =
23.4568
```

Uma linha de comando normalmente termina com a tecla **Enter**. Entretanto, se o último caractere for um ponto-e-vírgula (;), a variável que é produzida não é mostrada na tela. Se a expressão é complicada de tal forma que não cabe em uma linha, utiliza-se os três pontos (...) seguidos pela tecla **Enter** para indicar que a linha de comando continua na próxima linha. Pode-se formar nome de funções e variáveis com uma letra seguida por um número qualquer de letras e dígitos. No entanto, MATLAB guarda somente os 9 primeiros caracteres de um nome.

MATLAB entende de forma diferente letras maiúsculas e minúsculas. A e a não são a mesma variável. Os nomes das funções devem ser digitados em letras minúsculas.

2.5 Obtendo Informações do Espaço de Trabalho

Os exemplos previamente executados criaram variáveis que são armazenadas do espaço de trabalho do MATLAB. Para listar as variáveis armazenadas use o comando

```
>> who
```

Para verificar uma listagem mais completas sobre as variáveis armazenadas deve-se utilizar o comando whos, o qual traz informações sobre o nome, a dimensão da matriz correspondente, o número de elementos dessa matriz, entre outros.

2.6 O Comando Help

O comando help provê informação para a grande parte dos tópicos do MATLAB. O comando

```
>> help
```

sem nenhum argumento mostra uma lista de diretórios que contêm arquivos relacionados com MATLAB. Cada linha mostrada lista o nome de um diretório seguido por uma descrição do conteúdo do mesmo. Alguns diretórios são associados com os comandos básicos do MATLAB. Outros contêm toolboxes, tais como *control* e *signal*; esses possuem funções adicionais do MATLAB cobrindo áreas de aplicações mais especializadas.

Para uma lista de funções relacionadas a um determinado diretório, digita-se help seguido pelo nome do mesmo. Aparece uma lista de comandos, funções e símbolos específicos do MATLAB. O comando help seguido por estes comandos ou funções fornece informações sobre como utilizá-los em uma linha de comando.

Com o comando

```
>> help topico
```

topico deve ser um diretório ou o nome de uma função ou comando do MATLAB. Se for digitado

```
>> help inverse
```

MATLAB provavelmente irá responder

```
inverse not found.
```

Isto ocorre porque inverse não é o nome de uma função do MATLAB, a menos que tenha sido adicionada a uma toolbox pessoal.

2.7 Finalizando e Salvando o Espaço de Trabalho

Para finalizar o MATLAB digite quit ou exit. Terminando uma sessão do MATLAB, você apaga as variáveis do espaço de trabalho. Antes de terminar, pode-se salvar as variáveis digitando

```
>> save
```

Este comando salva todas as variáveis em um arquivo em disco chamado matlab.mat. Na próxima vez que MATLAB é chamado, pode-se executar o comando load para restaurar o espaço de trabalho com as variáveis de matlab.mat.

Podem ser utilizados os comandos save e load em arquivos com outros nomes, ou salvar somente algumas variáveis. O comando save temp salva as variáveis em um arquivo de nome temp.mat. O comando

```
>> save temp X
```

salva somente a variável X, enquanto que

```
>> save temp X Y Z
```

salva as variáveis X, Y e Z.

2.8 Números e Expressões Aritméticas

MATLAB utiliza a notação decimal convencional, com ponto decimal e sinal negativo opcionais, para números. Pode-se incluir um fator de escala em potência de dez ou uma indicação de número complexo como sufixo. Alguns exemplos de números válidos são

```
3   -99    0.0001  9.6397238   1.60210E-20   6.02252e23   2i   -3.141159i    3e5i
```

Podem ser construídas expressões com os operadores de aritmética usuais e as regras de precedência:

+	adição
-	subtração
*	multiplicação
/	divisão por número à direita

\ divisão por número à esquerda

^ potência

MATLAB possui também funções matemáticas elementares encontradas em calculadoras científicas. Estas funções incluem abs, sqrt, log e sin. Você pode facilmente adicionar mais funções através dos arquivos-M. Algumas funções simplesmente retornam valores especiais comumente utilizados, como a função pi que retorna o valor da constante . A função Inf indica resultado infinito; uma divisão por zero gera como resposta Inf. A variável NaN (Not a Number) possui diferentes propriedades com relação à variável Inf. Uma divisão 0/0 ou Inf/Inf produz NaN como resposta.

2.9 Números Complexos e Matrizes

MATLAB permite números complexos, indicados pela função especial i ou j, em todas as operações ou funções. Assim, temos:

```
>> z = 3 + 4*i
```

Uma outra forma é

```
>> w = r*exp(i*theta)
```

Quando forem entrados números complexos como elementos de matriz com os colchetes, deve ser evitado qualquer espaço em branco. Uma expressão como 1 + 5*i, com espaços em volta do sinal +, representa dois números. Isto também é válido para números reais; uma espaço antes do exponencial, como em 1.23 e-4, causa um erro.

O nome de uma função do MATLAB também pode ser usada como o nome de uma variável. Quando utilizado como uma variável, a função original se torna indisponível no espaço de trabalho até que a variável seja apagada.

2.10 Formatos de Saída

MATLAB mostra o resultado de uma linha de comando na tela, e atribui este resultado a uma variável específica, ou a ans se a variável não é dada. Você pode utilizar o comando format para controlar o formato numérico mostrado.

Se todos os elementos de uma matriz forem inteiros exatos, a matriz é mostrada em um formato sem pontos decimais. Se ao menos um elemento de uma matriz não é um inteiro exato, vários formatos de saída são possíveis. O formato padrão, chamado de *short format*, mostra aproximadamente cinco dígitos decimais significativos. Os outros formatos mostram mais dígitos significativos ou utilizam notação decimal. Como um exemplo, suponha

```
>> x = [4/3 1.2345e-6]
```

Os formatos e as saídas resultantes para este vetor são:

```
format short   1.3333 0.0000
format short e   1.3333e+00 1.2345e-06
format long   1.33333333333333 0.00000123456000
format long e   1.333333333333333e+00 1.234500000000000e-06
format bank   1.33 0.00
```

O comando help format mostra mais alguns tipos possíveis de formato.

2.11 Funções

Grande parte do poder do MATLAB á atribuído ao extensivo conjunto de funções. Algumas das funções são intrínsecas ao MATLAB. Outras funções estão disponíveis na biblioteca de arquivos-M distribuídos com MATLAB (a toolbox do MATLAB). Ainda existem outras que são adicionadas por usuários individuais ou grupos de usuários, para aplicações mais especializadas. Esta é uma importante característica do MATLAB; ***todo usuário pode criar funções que atuam da mesma forma que uma função intrínseca do MATLAB.***

Capítulo 3

Operações com Matrizes

As funções elementares de matrizes e a sintaxe das operações entre matrizes são o assunto deste capítulo.

A linha de comando B = A' indica que a variável B recebe a transposta da matriz A.

Para matrizes complexas, teremos que a linha de comando acima fornece a transposta complexa conjugada. Assim, para obtermos somente a transposta de uma matriz complexa devemos fazer B = conj (A').

A adição e subtração de matrizes pode ser feita entre matrizes de mesma dimensão ou entre um escalar e uma matriz. Na primeira, cada elemento de uma matriz é somado ou subtraído do correspondente na outra matriz. Na segunda forma, o escalar é adicionado ou subtraído de todos os elementos do outro operando.

A operação de multiplicação é definida quando o número de colunas da primeira matriz for igual ao número de linhas da segunda matriz. Dependendo das dimensões envolvidas, o resultado pode ser uma matriz, um vetor ou, até mesmo, um escalar. O produto de um escalar por uma matriz resulta em todos os elementos da matriz multiplicados pelo escalar.

Se A é uma matriz quadrada não-singular, então A\B e B/A corresponde formalmente a multiplicação à esquerda e à direita de B pela inversa de A; ou seja, inv(A)*B e B*inv(A), mas o resultado é obtido diretamente sem o cálculo da inversa. Em geral

X = A\B é uma solução para A*X = B

X = B/A é uma solução para X*A = B

A divisão à esquerda, A\B, é definida sempre que B tiver o mesmo número de linhas que A. Dependendo se A for quadrada ou não, deve-se usar algum método de fatoração.

A divisão normal de um escalar pelo outro é feita fazendo-se x/y.

A expressão A^p eleva A a p-ésima potência e é definida se A é uma matriz quadrada e p é um escalar.

MATLAB trata expressões como exp(A) e sqrt(A) como operações de vetores, definidas em cada elemento de A. As funções elementares de matrizes incluem

- poly - polinômio característico
- det - determinante
- trace - traço da matriz
- exp - exponencial de cada elemento da matriz
- log - logaritmo de cada elemento da matriz
- sqrt - raiz quadrada de cada elemento da matriz

MATLAB também calcula funções transcendentais, tais como exponencial e logaritmo da matriz. Estas funções especiais são definidas somente para matrizes quadradas. Uma função matemática transcendental é interpretada como uma função da matriz se um m é acrescentado ao nome da função. As três funções abaixo são distribuídas juntamente com o MATLAB

- expm - matriz exponencial
- logm - matriz logarítmica
- sqrtm - matriz raiz quadrada

Capítulo 4

Operações com Vetores

Neste capítulo são mostradas as operações lógicas e relacionais e as funções matemáticas para vetores.

O termo operações vetoriais refere-se a operações aritméticas de elemento-por-elemento, ao invés das operações algébricas lineares usuais de matrizes denotadas pelos símbolos * \ / ^ e ‘. Um ponto (.) precedendo um operador indica uma operação de elemento-por-elemento.

Para adição e subtração, operações vetoriais e operações matriciais são idênticas.

O símbolo .* significa multiplicação de elemento-por-elemento. Se A e B possuem a mesma dimensão, então A.*B resulta em uma matriz cujo os elementos são simplesmente os produtos dos elementos individuais de A e B.

A potência de elemento-por-elemento é simbolizada por .^, e faz com que cada elemento de uma matriz seja elevado ao elemento correspondente da outra matriz.

4.1 Operações Relacionais

Seis operadores relacionais estão disponíveis para a comparação de duas matrizes de dimensões idênticas.

<	menor que
<=	menor ou igual que
>	maior que
>=	maior ou igual que
==	igual
~=	diferente

MATLAB compara os elementos correspondentes de cada matriz.; o resultado é uma matriz de 1s e 0s, com 1 representando verdadeiro e 0 representando falso.

4.2 Operações Lógicas

Os operadores &, | e ~ correspondem aos operadores lógicos “e”, “ou” e “não”.

C = A & B é uma matriz cujo os elementos são 1s onde ambas as matrizes A e B são elementos não-nulos, e 0s onde uma das matrizes ou ambas são elementos nulos.

C = A | B é uma matriz cujo os elementos são 1s onde tanto A ou B possuem elementos não-nulos, e 0s onde ambas posssuem elementos nulos.

B = ~A é uma matriz cujo os elementos são 1s onde a matriz A é um elemento nulo, e 0s quando A é um elemento não-nulo.

Todas as operações acima são válidas apenas para quando A e B possuem mesma dimensão, ou quando uma das duas matrizes é um escalar.

As funções lógicas e relacionais do MATLAB são:

- any - condições lógicas
- all - condições lógicas
- find - encontra os índices da matriz de valores lógicos
- exist - verifica a existência de variáveis
- isnan - detecta se algum elemento da matriz é NaN
- isinf - detecta se algum elemento da matriz é infinito
- finite - verifica os valores finitos da matriz
- isempty - detecta matrizes vazias
- isstr - detecta variáveis string
- isglobal - detecta variáveis globais
- issparse - detecta matrizes esparsas

4.3 Funções Matemáticas

A funções trigonométricas incluídas no MATLAB são

- sin - seno
- cos - cosseno
- tan - tangente
- asin - arco-seno
- acos - arco-cosseno
- atan - arco-tangente
- atan2 - arco-tangente para os quatro quadrantes
- sinh - seno hiperbólico
- cosh - cosseno hiperbólico
- tanh - tangente hiperbólica
- asinh - arco-seno hiperbólico
- acosh - arco-cosseno hiperbólico
- atanh - arco-tangente hiperbólico

MATLAB inclui como funções elementares

- abs - valor absoluto ou módulo de um número complexo
- angle - ângulo de fase
- sqrt - raiz quadrada
- real - parte real
- imag - parte imaginária
- conj - complexo conjugado
- round - arredondamento para o inteiro mais próximo
- fix - arredondamento para o inteiro mais próximo de zero
- floor - arredondamento para o inteiro mais próximo de -
- ceil - arredondamento para o inteiro mais próximo de +
- sign - função sinal
- rem - remanescente ou módulo
- gcd - máximo divisor comum
- lcm - mínimo múltiplo comum
- exp - exponencial de base e
- log - logaritmo natural
- log10 - logaritmo de base 10

Capítulo 5

Manipulação de Vetores e Matrizes

Neste capítulo são mostrados comandos que geram vetores e funções que geram matrizes.

5.1 Gerando Vetores

Os dois pontos (:) representam um caractere importante no MATLAB. A linha de comando

```
>> x = 1:5
```

gera um vetor linha contendo os números de 1 a 5 com incremento de uma unidade. Desta forma é produzido o vetor

```
x =
1 2 3 4 5
```

Pode-se utilizar incrementos diferentes da unidade como em

```
>> y = 0: pi/4: pi
```

que resulta em um vetor linha começando em zero e terminando em (3.1416) com incremento de /4 (0.7854). Esta notação do dois pontos possibilita a fácil geração de tabelas. Para se obter um vetor coluna basta transpor o vetor linha gerado.

Outras funções para geração de vetor incluem o logspace, o qual gera vetores espaçados logaritmicamente, e o linspace, que permite que você especifique o número de pontos ao invés do incremento.

```
>> k = linspace (-pi,pi,4)
k =
-3,1416 -1.0472 1.0472 3.1416
```

5.2 Matrizes Especiais

MATLAB apresenta algumas funções úteis para gerar matrizes:

- zeros - zeros
- ones - constante
- rand - elementos randômicos uniformemente distribuídos
- randn - elementos randômicos normalmente distribuídos
- eye - identidade
- linspace - vetores espaçados linearmente
- logspace - vetores espaçados logaritmicamente
- meshgrid - utilizada com funções de duas variáveis

5.3 Manipulação de Matrizes

- rot90 - rotação
- fliplr - inverte a matriz da esquerda para a direita
- flipud - inverte a matriz de cima para baixo
- diag - extrai ou cria diagonal
- tril - triângulo inferior
- triu - triângulo superior
- reshape - altera o formato
- ‘ - transposição
- : - rearranjamento geral

Capítulo 6

Introdução aos Gráficos

Este capítulo mostra como visualizar dados em gráficos 2-D. As funções elementares de plotagem comentadas no início são descritas no capítulo 16.

6.1 Funções Elementares de Plotagem

A lista a seguir relaciona as funções que produzem gráficos simples. Estas funções se diferem apenas na maneira como apresentam as escalas dos eixos dos gráficos. Cada uma aceita a entrada na forma de vetores ou matrizes e, automaticamente, definem as escalas dos eixos de modo que os dados de entrada fiquem bem acomodados.

- plot - cria um gráfico de vetores ou de colunas de matrizes
- loglog - cria um gráfico utilizando escalas logarítmicas para ambos os eixos
- semilogx - cria um gráfico utilizando escala logarítmica no eixo x e escala linear no eixo y
- semilogy - cria um gráfico utilizando escala logarítmica no eixo y e escala linear no eixo x

Pode-se adicionar título, nome aos eixos, linhas de grade e textos a qualquer gráfico utilizando

- title - adiciona um título ao gráfico
- xlabel - define um nome para a variável do eixo x
- ylabel - define um nome para a variável do eixo y
- text - adiciona um texto em lugar específico
- gtext - adiciona um texto ao gráfico utilizando o mouse
- grid - ativa as linhas de grade

6.2 Criando um Gráfico

Se y é um vetor, o comando plot(y) produz um gráfico linear dos elementos de y versus o índice dos elementos de y. Se são especificados dois vetores como argumentos, o comando plot(x,y) produz um gráfico de y versus x.

Pode-se, também, especificar vários grupos de dados e definir o estilo de linha e a cor que serão usados em cada grupo em uma única chamada ao comando plot:

```
>> t = 0:pi/100:2*pi;
>> x = sin(t);
>> y1 = sin(t+.25);
>> y2 = sin(t+.5);
>> plot(x,y1,'r-',x,y2,'g-')
```

plot produz um gráfico de y1 versus x e y2 versus x no mesmo eixo. O primeiro grupo de dados (y1) é plotado com uma linha vermelha sólida e o segundo grupo (y2) é plotado com uma linha tracejada verde.

A linha de comando abaixo adiciona um título ao gráfico e nome aos eixos:

```
>> title('Phase Shift')

>> xlabel('x=sin(t)')

>> ylabel('y=sin(t+)')
```

6.3 Estilos de Linha, Marcadores e Cor

Como foi citado, pode ser passada uma string de caracteres como um argumento para a função plot de modo a especificar vários estilos de linhas, símbolos de traçado e cores. Na linha de comando,

```
>> plot(X,Y,S)
```

S é uma string envolvida por aspas e construída com os caracteres mostrados na tabela abaixo:

Símbolo	Cor	Símbolo	Estilo de Linha
y	amarelo	.	ponto
m	magenta	o	círculo
c	ciano	x	x
r	vermelho	+	sinal positivo
g	verde	*	asterisco
b	azul	-	sólida
w	branco	:	pontilhada
k	preto	-.	traço e ponto
		--	tracejada

Por exemplo, plot(X,Y,'b*') plota um asterisco azul em cada ponto de dado.

Se não for especificada uma cor, a função plot automaticamente utiliza as cores na ordem em que se apresentam na tabela acima. Desta forma, para uma linha somente, o gráfico é traçado em amarelo.

6.4 Adicionando Linhas a um Gráfico Existente

Podem ser adicionadas linhas a um gráfico já existente utilizando o comando hold. Quando hold é ativado, MATLAB não remove as linhas já existentes; ao invés disto, adiciona novas linhas aos eixos existentes. Pode ser que os eixos sejam redefinidos se os novos dados se encontrarem fora da escala do antigo eixo. Por exemplo, a linha de comando abaixo traça as três curvas em uma mesma figura:

```
>> plot(x)
>> hold on
>> plot(y1,'-')
>> plot(y2,'-.')
>> hold off
```

6.5 Dados Imaginários e Complexos

Quando os argumentos a serem plotados são complexos, a parte imaginária é ignorada exceto quando o comando plot é dado simplesmente com um argumento complexo. Para este caso especial, o comando é um atalho para um gráfico da parte real em função da parte imaginária. Portanto

```
>> plot(Z)
```

onde Z é um vetor ou uma matriz complexa, é equivalente a

```
>> plot(real(Z),imag(Z))
```

6.6 Plotando Matrizes

Um arquivo-M que possui uma função para gerar uma matriz de dados é o peaks. Os dados são baseados em uma função de

duas variáveis, tendo três pontos de máximo e mínimo:

$$f(x, y) = 3(1-x)^2 e^{-x^2-(y+1)^2} - 10(\frac{x}{5} - x^3 - y^5)e^{-x^2-y^2} - \frac{1}{3}e^{-(x+1)^2-y^2}$$

O arquivo-M cria uma matriz que possui os valores da função para x e y na faixa de -3 a 3. Os valores de x variam ao longo das colunas e os de y ao longo das linhas. Você pode especificar o tamanho da matriz quadrada que é retornada passando como argumento à função peaks. Por exemplo,

```
>> M = peaks(20)
```

cria uma matriz de dados de dimensão 20 por 20. Se você omitir o argumento de entrada, o valor utilizado é o padrão, que é 49.

A função plot pode ter um único argumento que pode ser uma matriz:

```
>> plot(Y)
```

Isto faz com que seja traçada uma linha no gráfico para cada coluna de Y. O eixo x é assumido como sendo o índice do vetor linha, 1:m, onde m é o número de linhas de Y. Por exemplo, a linha de comando

```
>> plot(peaks)
```

produz um gráfico com 49 linhas.

A função plot também aceita dois vetores ou matrizes como argumentos. Por exemplo,

```
>> plot(peaks,rot90(peaks'))
```

produz um gráfico da função peaks em função da transposta deslocada de 90° da mesma função peaks.

Em geral, se plot é utilizada com dois argumentos, e se X ou Y possuem mais de uma linha ou coluna, então:

- Se Y é uma matriz, e x é um vetor, plot(x,Y) plota sucessivamente as linhas ou colunas de Y, utilizando diferentes cores ou tipos de linha para cada, em função do vetor x. A orientação de linhas ou colunas é selecionada de forma a possuir o mesmo número de elementos de x. Se Y é quadrada, suas colunas é que são usadas.

- Se X é uma matriz e y é um vetor, plot(X,y) plota cada linha ou coluna de X em função do vetor y.

- Se ambos X e Y são matrizes de mesma dimensão, plot(X,Y) plota as colunas de X em função das colunas de Y. Também pode ser utilizada a função plot com vários pares de matrizes no argumento.

```
>> plot(X1,Y1,X2,Y2,...)
```

Cada par X-Y é plotado, gerando múltiplas linhas. Os diferentes pares podem ser de dimensões diferentes.

6.7 Copiando Gráficos

Os gráficos criados no MATLAB não podem ser salvos como é feito com o espaço de trabalho. Pode-se, entretanto, copiá-los como arquivo bitmap (BMP) para a área de transferência e depois colá-lo no Word ou no Paintbrush, por exemplo. Para tanto, escolha, na figura, o menu *Edit* e clique em *Copy*; a figura foi copiada para a área de transferência. Basta, agora, entrar no Word ou no Paintbrush e escolher a opção *Colar* do menu *Editar*.

PARTE II

Informações Avançadas

A segunda parte traz os capítulos de 7 a 15, e contém tópicos mais avançados.

Para avançar nesta parte o leitor deve ter compreendido a primeira parte.

Os tópicos mais importantes e que o usuário deve ter o controle ao final desta parte dizem respeito a representação polinomial, a utilização das opções de fluxo de controle e a criação de arquivos-M, sejam nas formas script ou função.

Capítulo 7

Análise de Dados

Este capítulo apresenta um grupo de funções que possuem a capacidade de análise de dados.

Um grupo de funções que possui a capacidade de análise de dados é mostrado abaixo:

- max - valor máximo
- min - valor mínimo
- mean - valor médio
- std - desvio médio
- sort - ordenar
- sum - soma dos elementos
- prod - produto dos elementos
- cumsum - soma acumulativa dos elementos
- cumprod - produto acumulativo dos elementos
- hist - histograma

Quando os argumentos forem vetores, não irá importar se o vetor é orientado por linha ou por coluna. Se o argumento for uma matriz, as funções operarão orientadas por coluna. Desta forma, se você aplicar a função max a uma matriz, o resultado é um vetor linha contendo o valor máximo de cada coluna.

Capítulo 8

Funções de Matriz

Neste capítulo são descritas algumas funções especiais para aplicações em matrizes, tais como fatoração e decomposição.

8.1 Fatoração Triangular

O método de fatoração mais básico expressa uma matriz qualquer como o produto de duas matrizes essencialmente triangulares, sendo uma delas a permutação de uma matriz triangular inferior e a outra uma matriz triangular superior. Esta fatoração é frequentemente chamada de LU, ou, em algumas vezes, de LR.

A linha de comando é descrita como

```
>> [L,U] = lu(A)
```

onde L é a permutação da matriz triangular inferior e U é a matriz triangular superior.

8.2 Fatoração Ortogonal

A fatoração QR é utilizada tanto para matrizes quadradas como retangulares. Esta fatoração expressa a matriz como o produto de uma matriz ortonormal e uma matriz triangular superior.

A linha de comando para esta fatoração é descrita como

```
>> [Q,R] = qr(A)
```

onde Q é a matriz ortonormal e R é a matriz triangular superior.

8.3 Decomposição dos Autovalores

Se A é uma matriz n-por-n, os n números que satisfazem $Ax = x$ são os autovalores de A. Eles são encontrados utilizando

```
>> eig(A)
```

que retorna os autovalores em uma vetor coluna. Se A é real e simétrica, os autovalores são reais. Mas se A não é simétrica, os autovalores frequentemente são números complexos. Para se obter os autovalores e os autovetores de uma vez basta utilizar a seguinte linha de comando:

```
>> [X,D] = eig(A)
```

onde os elementos da diagonal de D são os autovalores e as colunas de X são os autovetores correspondentes tais que AX = XD.

Capítulo 9

Polinômios e Processamento de Sinais

A representação polinomial e o tratamento dois polinômios são tratados neste capítulo.

9.1 Representação Polinomial

MATLAB representa polinômios como vetores linha contendo os coeficientes na ordem decrescente de potência. Por exemplo, a equação característica da matriz

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

é computada através de

```
>> p = poly(A)

p =
1 -6 -72 -27
```

Esta é a representação do MATLAB para o polinômio $\mathbf{s^3 - 6s^2 - 72s - 27}$.

As raízes desta equação são dadas por

```
>> r = roots(p)

r =
12.1229
-5.7345
-0.3884
```

As raízes da equação característica são os autovalores da matriz A. Pode-se obter o polinômio original através das raízes encontradas acima

```
>> p2 = poly®
p2 =
1 -6 -72 -27
```

Considere os polinômios $\mathbf{a(s) = s^2 + 2s + 3}$ e $\mathbf{b(s) = 4s^2 + 5s + 6}$. O produto dos polinômios é feito através da convolução dos coeficientes

```
>> a = [1 2 3]; b = [4 5 6];
>> c = conv(a,b)
c =
4 13 28 27 18
```

Uma lista completa das funções polinomiais inclui

- poly - polinômio característico
- roots - raízes do polinômio
- polyval - avalia o polinômio com o argumento substituindo a variável
- polyvalm - avalia o polinômio com o argumento sendo uma matriz
- conv - multiplicação
- deconv - divisão

- residue - expansão em frações parciais
- polyder - derivada do polinômio
- polyfit - ajuste do polinômio

9.2 Processamento de Sinais

Vetores são utilizados para guardar sinais amostrados ou sequências em processamento de sinais. Para sistemas de várias entradas, cada linha de uma matriz corresponde a um ponto de amostra onde as colunas representam os diversos canais. MATLAB possui algumas funções de processamento de sinais:

- abs - módulo de um número complexo
- angle - ângulo de fase de um número complexo
- conv - convolução
- deconv - desconvolução
- fft - transformada de Fourier
- ifft - transformada de Fourier inversa
- fftshift - troca os quadrantes da matriz Para argumentos bidimensionais, como matrizes, usam-se as funções modificadas:
  - fft2 - fft bidimensional
  - ifft2 - ifft bidimensional
  - conv2 - convolução bidimensional

9.3 Filtro de Dados

A função

```
>> y = filter(b, a, x)
```

filtra os dados contidos no vetor x com o filtro descrito pelos vetores a e b, criando os dados filtrados y. A estrutura do filtro pode ser descrita pela equação diferencial

$$\mathbf{y(n) = b(1)x(n) + b(2)x(n-1) + ... + b(nb)x(n-nb+1) - a(2)y(n-1) - ... - a(na)y(n-na+1)}$$

ou de modo equivalente, pela transformada Z

$$\mathbf{H(z) = \frac{Y(z)}{X(z)} = \frac{b(1) + b(2)z^{-1} + ... + b(nb)z^{-(na-1)}}{1 + a(2)z^{-1} + ... + a(na)z^{-(na-1)}}}$$

A função freqz retorna a resposta complexa em frequência de filtros digitais.

Capítulo 10

Funções de Função

Neste capítulo são tratadas as funções do MATLAB que não trabalham com matrizes numéricas, mas com funções matemáticas. As *funções de função* são descritas como sendo:

- Integração numérica
- - Equações não-lineares
- - Solução de equações diferenciais

MATLAB representa funções matemáticas através de arquivos de função-M. Por exemplo, a função

$$f(x) = \frac{1}{(x-0.3)^2 + 0.01)} + \frac{1}{(x-0.9)^2 + 0.04} - 6$$

pode ser encontrada no MATLAB, pois foi criada através de um arquivo-M chamado humps.m.

10.1 Integração Numérica

A área sob uma função *f(x)* pode ser determinada fazendo-se a integração numérica de *f(x).* Para integrar a função definida por humps.m de 0 a 1

```
>> q = quad('humps',0,1)
q =
29.8583
```

Pode-se notar que o primeiro argumento da função quad é o nome de uma função entre aspas. Isto é porque quad é chamada uma função de função - uma função que opera em outra função.

Uma gráfico da função é obtido através de

```
>> x = -1:.01:2;
>> plot(x,humps(x))
```

10.2 Equações Não-Lineares e Funções de Otimização

As funções de função para equações não-lineares e otimização incluem

- fmin - mínimo de uma função de uma variável
- fmins - mínimo de uma função de várias variáveis
- fzero - zero de uma função de uma variável

Continuando o exemplo definido por humps.m, a localização do mínimo na região de 0.5 a 1 é computado com fmin:

```
>> xm = fmin('humps',.5,1)
xm =
0.6370
```

O valor deste mínimo é

```
>> y = humps(xm)
y =
11.2528
```

Pelo gráfico, pode-se observar que a função humps apresenta dois zeros. A localização do zero próximo a x = 0 é

```
>> xz1 = fzero('humps',0)
xz1 =
-0.1316
```

O zero perto de x = 1 está em

```
>> xz2 = fzero('humps',1)
xz2 =
1.2995
```

10.3 Funções de Equações Diferenciais

As funções do MATLAB para solução de equações diferenciais ordinárias são

- ode23 - método de Runge-Kutta para 2ª e 3ª ordens
- ode45 - método de Runge-Kutta-Fehlberg para 4ª e 5ª ordens

Capítulo 11

Fluxo de Controle

Os fluxos de controle encontrados na maioria das linguagens de programação são tratados neste capítulo.

As linhas de comando de fluxo de controle permitem que MATLAB seja utilizado como uma linguagem de programação completa de alto-nível.

11.1 Loop FOR

MATLAB possui sua própria versão dos loops FOR e DO encontrados em linguagens de programação. Isto permite que um grupo de linhas de comando seja repetido por um número fixo e predeterminado de vezes. Por exemplo,

```
>> for i = 1:n, x(i) = 0, end
```

atribui 0 aos n primeiros elementos de x. Se n é menor do que 1, a construção ainda é válida, mas MATLAB não executa a linha de comando interna. Se x não existe ou possui menos de n elementos, então o espaço adicional é alocado automaticamente.

Uma importante observação é que se deve sempre finalizar um loop for com um end. Se for digitado

```
>> for i = 1:n, x(i) = 0
```

o sistema espera que sejam entradas o restante das linhas de comando no corpo do loop. Nada acontece até que seja digitado end.

A forma geral do loop for é

```
for v = expressão
linhas de comando
end
```

Pode-se utilizar mais de um loop dentro de um loop, mas não se pode esquecer que cada for deve ter seu próprio end. Pode-se, também, fazer com que o loop decresça, de acordo com a expressão descrita. Por exemplo,

```
>> for j = n-1:-1:1
A(:, j) = t.*A(:, j+1);
end
```

11.2 Loop WHILE

MATLAB possui sua própria versão do loop while, o qual permite que uma linha de comando, ou um grupo de linhas de comando, seja repetida um número indefinido de vezes, através do controle de uma condição lógica.

A forma geral de um loop while é:

```
>> while expressão
linhas de comando
end
```

As linhas de comando são executadas repetidamente enquanto os elementos da expressão não assumam valor zero. A matriz da expressão é quase sempre uma expressão relacional 1 por 1, então números não-nulos correspondem a uma expressão verdadeira. Quando a matriz da expressão não é um escalar, pode-se reduzi-la através das funções any e all.

11.3 Comandos IF *e* BREAK

O exemplo abaixo ilustra a utilização dos comandos while e if. Ele também mostra a função input e o comando break. Este exemplo trabalha com qualquer número inteiro positivo. Se ele é par, faz-se a divisão por 2; se ele é ímpar, faz-se a multiplicação por 3 e soma com 1. Este processo é repetido até que o resultado seja 1.

```
n = input('Enter n, negative quits. ');
if n 45= 0, break, end
while n > 1
if rem(n,2) == 0
n = n/2
else
n = 3*n+1
end
end
end
```

Capítulo 12

Arquivos-M: Scripts e Funções

Um dos assuntos mais importantes, a criação de arquivos-M é descrita neste capítulo. MATLAB é usualmente acionado por um comando; quando se entra com uma simples linha de comando, MATLAB a processa imediatamente e mostra o resultado. MATLAB também pode executar uma sequência de comandos que está armazenada em um arquivo.

Arquivos de disco que possuem linhas de comando para MATLAB são chamados arquivos-M em virtude de sua extensão ser do tipo .m. Por exemplo, o arquivo bessel.m contém linhas de comando do MATLAB para avaliar funções Bessel. Um arquivo-M consiste de uma sequência normal de linhas de comando do MATLAB, a qual pode fazer uma chamada a outros arquivos-M. Um arquivo-M pode chamar a si mesmo de modo recursivo. Pode-se criar estes arquivos com um editor de textos como o Notepad ou o Word.

Dois tipos de arquivos-M podem ser usados: scripts e funções. Scripts, ou arquivos script, realizam longas sequências de comandos. Funções, ou arquivos função, permitem adicionar novas funções à funções já existentes. A maior parte do poder do MATLAB se deve ao fato de se poder criar novas funções que resolvam problemas específicos.

12.1 Arquivos Script

Quando um script é chamado, MATLAB simplesmente executa os comandos encontrados no arquivo. As linhas de comando de um arquivo script operam globalmente com os dados que estão no espaço de trabalho. Scripts são úteis na realização de análise, solução de problemas, ou no projeto de longas sequências de comando, o que se torna cansativo para ser feito interativamente. Como um exemplo, suponha um arquivo chamado fibno.m que possui os comandos:

```
f = [1 1]; i = 1;
while f(i) + f(i+1) < 1000
f(i+2) = f(i) + f(i+1);
i = i + 1;
end
plot(f)
```

Digitando a linha de comando fibno faz com que MATLAB execute os comandos, calculando os 16 primeiros números da série de Fibonacci, e crie um gráfico. Após a execução do arquivo estar completa, as variáveis f e i ficam mantidas no espaço de trabalho.

Os demos fornecidos pelo MATLAB são bons exemplos de como se utilizar scripts para realizar tarefas mais complicadas. Para usá-los, basta digitar demos no prompt do MATLAB.

12.2 Arquivos Função

Um arquivo-M que contém a palavra function no ínicio da primeira linha é um arquivo função. Uma função difere de um script pelos argumentos que devem ser passados e pelas variáveis que são definidas e manipuladas, que são locais à função e não podem ser operadas globalmente no espaço de trabalho. O arquivo mean.m é um exemplo de um arquivo função que possui as linhas de comando:

```
function y = mean(x)
% MEAN Average or mean value
% For vectors, Mean(x) returns the mean value
% For matrices, MEAN(x) is a row vector
% containing the mean value of each column.
[m,n] = size(x);
if m == 1
m = n;
end
y = sum(x)/m;
```

A existência deste arquivo define uma nova função chamada mean. A nova função mean é usada como qualquer outra função do MATLAB. Por exemplo, se z é um vetor de inteiros de 1 a 99,

```
>> z = 1:99;
```

o valor médio deste é encontrado através do comando

```
>> mean(z)
```

que resulta em

```
ans =
50
```

As informações abaixo são para o arquivo mean.m, mas o princípio é válido para todos os arquivos função:

- A primeira linha declara o nome da função e os argumentos de entrada e saída. Sem esta linha, o arquivo é um arquivo script, e não um arquivo função.
- O símbolo % indica que o restante da linha é um comentário e deve ser ignorado.
- As primeiras linhas descrevem o arquivo-M e são mostradas quando você digita help mean.
- As variáveis m, n e y são locais a mean e não aparecem no espaço de trabalho após mean ter terminado. (Ou, se elas existem, permanecem inalteradas.)
- Não é necessário definir os inteiros de 1 a 99 em uma variável de nome x. No exemplo, a função mean foi usada com uma variável z. O vetor z que contém os inteiros de 1 a 99 foi passado ou copiado para mean onde ele se tornou uma variável local de nome x.

Pode-se criar uma ajuda online para os arquivos-M entrando com um texto de uma ou mais linhas de comentários, começando pela segunda linha do arquivo. Por exemplo, o arquivo-M angle.m contém

```
function p = angle(h)
%ANGLE Phase angle.
ANGLE(H) returns the phase angles, in radians, of a  matrix with complex elements.

See also ABS, UNWRAP.
```

Quando se entra com help angle, as linhas 2, 3 e 4 são mostradas. Baseado nisto, a primeira linha de comentários em qualquer arquivo-M deve conter o maior número de informações possíveis.

12.3 Variáveis Globais

Cada função do MATLAB definida por um arquivo-M possui suas próprias variáveis locais, as quais não tem relação com as de outras funções e com as do espaço de trabalho. Entretanto, se várias funções e também o plano de trabalho declararem uma variável particular como global, então todos eles dividem a mesma variável. Qualquer atribuição a esta variável, em qualquer função, fica disponível a todas as outras funções que a declaram como global.

Para fazer com que uma variável seja global, basta escrever

```
global X Y Z
```

onde as variáveis X, Y e Z irão trabalhar como sendo globais.

12.4 Strings de Texto

Strings de texto são entradas no MATLAB entre aspas simples (‘). Por exemplo,

```
>> s = 'Hello'
```

resulta em

```
s =
Hello
```

O texto é armazenado em um vetor, sendo um caractere por elemento. Neste caso,

```
>> size(s)

ans =
1 5
```

indica que s possui cinco elementos. Os caracteres são armazenados com seus valores ASCII, e a função abs mostra estes valores:

```
>> abs(s)

ans =
72 101 108 108 111
```

Utiliza-se colchetes para juntar variáveis de texto em strings maiores:

```
>> s = [s, ` World `]

s =
Hello World
```

A Função eval

A função eval trabalha com variáveis do tipo texto e implementa uma poderosa facilidade de macro. eval(t) faz com que o texto contido em t seja avaliado. Por exemplo,

```
t = ‘1/(i+j-1)’;
for i = 1:n
for j = 1:n
a(i,j) = eval(t);
end
end
```

cria uma matriz a de dimensão n por n, onde cada elemento é avaliado pela função que está descrita por t.

Capítulo 13

Arquivos de Disco

Os comandos relacionados com arquivos que permitem a importação e a exportação de dados são os assuntos deste capítulo. Os comandos do MATLAB save e load, gravam e executam o conteúdo do espaço de trabalho, respectivamente. Outros comandos relacionados com arquivos de disco permitem executar programas externos e torna possível a importação e a exportação de dados. Alguns desses comandos são dir, type, delete e cd. Eles atuam de forma idêntica como atuam no MS-DOS.

13.1 Importando e Exportando Dados

Pode-se introduzir dados de outros programas no MATLAB de várias maneiras. Da mesma forma, também pode-se exportar dados para outros programas. Pode acontecer, ainda, do programa manipular dados diretamente de arquivos-MAT, o formato de arquivo que MATLAB utiliza.

13.1.1 Importando Dados

O melhor método de importar dados depende da quantidade de dados existente, se eles já estão em uma forma de leitura conveniente, qual é esta forma, etc. Abaixo estão listadas formas; deve-se selecionar a mais apropriada.

- Entrar os dados como uma lista explícita de elementos. Se é pequena a quantidade de dados, digamos menos de 10 elementos, é fácil de digitá-los explicitamente utilizando colchetes. Este método se torna inconveniente no caso de ter que se digitar uma quantidade grande de dados, pois não é possível editá-los caso se cometa um erro.

- Criar os dados através de um arquivo-M. Neste caso, utiliza-se um editor de texto para criar um arquivo-M script que entra com os dados como uma lista explícita de elementos. Este método é útil quando os dados não estão em um formato apropriado de leitura e têm que ser digitados de qualquer maneira. Este método apresenta vantagem com relação ao primeiro, pois permite que os erros que possam ter acontecido durante a entrada dos dados, sejam corrigidos.

- Ler dados de um arquivo ASCII. Este arquivo armazena dados no formato ASCII, com linhas de comprimento fixo terminadas com ENTER e com espaços entre os números (arquivos ASCII podem ser editados fazendo-se uso de um editor de texto normal). Os arquivos podem ser lidos diretamente no MATLAB através do comando load. O resultado é colocado em uma variável cujo nome é o nome do arquivo.

- Ler dados utilizando fopen, fread e outras funções de E/S de baixo-nível. Este método é útil quando se quer acessar arquivos de dados de outras aplicações que possuem seus próprios formatos.

- Desenvolver um programa em C para traduzir seus dados para o formato do arquivo-MAT e depois ler este arquivo no MATLAB através do comando load.

13.1.2 Exportando Dados do MATLAB

Alguns métodos para exportação de dados estão descritos a seguir:

- Para pequenas matrizes, utiliza-se o comando diary para criar um arquivo diary, para onde as variáveis são passadas. Pode-se utilizar um editor de texto para manipular o arquivo diary após ter sido criado. A saída de diary inclui os comandos do MATLAB utilizados durante a sessão.

- Salvar os dados no formato ASCII utilizando o comando save com a opção -ascii. Por exemplo,

```
>> A = rand(4,3);
>> save temp.dat A -ascii
```

cria um arquivo ASCII chamado temp.dat.

- Gravar os dados em um formato especial utilizando fopen, fwrite e outras funções de E/S de baixo-nível. Este método é útil quando se quer gravar os dados em um formato de arquivo requerido por uma aplicação.

- Gravar os dados em um arquivo-MAT utilizando o comando save, e escrever um programa em C para traduzir este arquivo-MAT para o formato que se desejar.

Capítulo 14

O Debugger do MATLAB

Este capítulo descreve comando que auxiliam na descoberta de erros que ocorrem durante o tempo de execução. Apesar da linguagem que MATLAB utiliza ser menos complexa que as outras linguagens de programação, ele possui sua própria sintaxe, e talvez seja necessário reparar alguns erros que venham a acontecer. MATLAB encontra erros de sintaxe durante a compilação. Estes erros usualmente são fáceis de consertar. MATLAB também pode encontrar erros durante o tempo de execução; estes erros tendem a ser mais difíceis de serem repararados, pois o espaço de trabalho local à função é perdido quando um erro força o retorno ao prompt do MATLAB e ao espaço de trabalho principal. Se utiliza-se o ponto-e-vírgula para que os resultados imediatos das linhas de comando não sejam mostrados, não se saberá onde ocorreu o erro. Para se mostrar os resultados intermediários, pode-se utilizar qualquer um dos métodos abaixo:

- Remover os ponto-e-vírgulas de forma que se possa visualizar os resultados imediatos.
- Adicionar o comando keyboard para que se possa examinar a situação do espaço de trabalho no ponto onde keyboard foi inserido.
- Retirar a declaração de function para que os arquivos-M possam ser executados como script, tornando os resultados intermediários disponíveis no espaço de trabalho principal.
- Utilizar o debugger do MATLAB.

Os três primeiros métodos requerem a edição do arquivo-M. O último método foi introduzido a partir da versão 4.0 do MATLAB, e os seus comandos são

dbstop	Introduz uma interrupção
dbclear	Remove a interrupção
dbcont	Continua a execução
dbdown	Muda o contexto do espaço de trabalho local
dbstatus	Lista todas as interrupções
dbstep	Executa uma ou mais linhas
dbtype	Lista os arquivos-M com o número das linhas
dbup	Muda o contexto do espaço de trabalho local
dbquit	Finaliza o modo debug

Capítulo 15

Arquivos E/S

As funções de E/S de arquivo do MATLAB que permitem a leitura e a escrita em formato diferente ao gerados pelo mesmo são descritas neste capítulo.

As funções de E/S (Entrada/Saída) de arquivo do MATLAB permitem a leitura de dados coletados em outro formato diretamente pelo MATLAB, ou a escrita de dados gerados pelo MATLAB no formato requerido por outro programa ou dispositivo. As funções lêem e gravam arquivos de texto formatados e arquivos binários de dados.

15.1 Abrindo e Fechando Arquivos

Antes de se ler ou escrever em um arquivo, deve-se abri-lo com o comando fopen, especificando o arquivo a ser aberto e a string de permissão. Por exemplo,

```
>> fid = fopen('pen.dat','r')
```

abre para leitura o arquivo pen.dat.

As strings de permissão disponíveis são:

'r' para leitura
'w' para gravação
'a' para atribuição
'r+' tanto para leitura como para gravação

Outras strings de permissão podem ser obtidas com o comando help fopen.

A função fopen retorna um identificador de arquivo, que é um inteiro positivo atribuído ao arquivo pelo sistema operacional. Este identificador de arquivo é basicamente um atalho para se referenciar o arquivo. As funções de E/S de arquivo do MATLAB utilizam o identificador como argumento para identificar o arquivo aberto para leitura, escrita ou encerramento. Se o arquivo não pode ser aberto, fopen retorna -1 como identificador. É aconselhável testar o identificador cada vez que um arquivo é aberto. Um segundo valor que é retornado pode fornecer informação adicional sobre erros. Por exemplo, se MATLAB não encontra o arquivo pen.dat, o comando

```
>> [fid, message] = fopen('pen.dat','r')
```

atribui -1 para fid, e message recebe uma string com a forma abaixo

```
No such file or directory.
```

Uma vez aberto, o arquivo fica disponível para leitura e gravação. Quando se termina a leitura ou a gravação, usa-se fclose para fechar o arquivo. Por exemplo,

```
>> status = fclose(fid)
```

fecha o arquivo associado com o identificador fid, e

```
>> status = fclose('all')
```

fecha todos os arquivos abertos. Ambas as formas retornam 0 se esta operação for realizada com sucesso, ou -1 se algo de errado acontecer.

15.2 Leitura de Arquivos com Dados em Binário

A função fread lê arquivos de dados binários. Na sua forma mais simples, ele lê um arquivo inteiro em uma matriz. Por exemplo,

```
>> fid = fopen('pen.dat','r');
>> A = fread(fid);
>> status = fclose(fid);
```

lê todos os dados do demo penny como caractere, e os escreve em uma matriz A. Dois argumentos opcionais a fread fazem o controle do número de valores lidos e a precisão de cada valor.

```
>> fid = fopen('pen.dat','r');
>> A = fread(fid,100);
>> status = fclose(fid);
```

lê os 100 primeiros valores de dados em um vetor coluna A. Substituindo o número 100 pelas dimensões de uma matriz [10,10], faz com que sejam lidos os mesmos 100 elementos, armazenando-os em uma matriz 10x10. E

```
>> A = fread(fid,Inf)
```

lê até o final do arquivo, preenchendo a matriz A como um vetor coluna. Omitir o tamanho do argumento produz o mesmo efeito. O argumento de precisão numérica controla o número de bits lidos em cada valor e a interpretação destes bits como valores caractere, inteiro ou ponto flutuante. Algumas precisões comuns incluem

- 'char' e 'uchar' para caracteres com sinal e sem sinal (tipicamente 8 bits)
- 'short' e 'long' para inteiros curtos e longos (tipicamente 16 e 32 bits, respectivamente)
- 'float' e 'double' para valores em ponto flutuante de precisão simples e dupla (tipicamente 32 e 64 bits, respectivamente)

Se fid se referir a um arquivo aberto contendo valores em ponto flutuante, então

```
>> A = fread(fid,10,'float')
```

lê 10 valores em ponto flutuante preenchendo um vetor coluna A.

15.3 Escrevendo em Arquivos com Dados em Binário

A função fwrite escreve os elementos de uma matriz em um arquivo com uma precisão numérica específica, retornando o número de valores escritos. Por exemplo,

```
>> fwriteid = fopen('magic5.bin','w');
>> count = fwrite(fwriteid,magic(5),'integer*4');
>> status = fclose(fwriteid);
```

cria um arquivo binário de 100 bytes contendo os 25 elementos da matriz quadrada 5x5, armazenados em inteiros de 4 bytes. Isto atribui 25 à variável count.

15.4 Escrevendo Arquivos de Texto Formatados e Strings

A função fprintf converte dados em strings de caractere e os mostra na tela ou em um arquivo. O formato de saída é definido por um especificador de conversão e por um texto. Os especificadores de conversão controlam a saída dos elementos de uma matriz. Os textos são copiados diretamente. Os especificadores são precedidos pelo caractere %; conversões comuns incluem

- %e para notação exponencial
- %f para notação de ponto fixo
- %g que seleciona automaticamente o menor entre %e e %f

Campos opcionais no especificador de formato controlam o tamanho e a precisão do campo. Por exemplo,

```
>> x = 0: .1: 1;
>> y = [x; exp(x)];
>> fid = fopen('exptable.txt','w');
>> fprintf(fid,'Exponential Function \n\n');
>> fprintf(fid,'%6.2f %12.8f\n',y);
>> status = fclose(fid);
```

cria um arquivo de texto contendo uma pequena tabela para a função exponencial. A primeira chamada a fprintf escreve o título, seguido por dois comandos ENTER, o qual é definido por \n. A segunda chamada escreve a tabela propriamente dita. As strings de controle de formato definem o formato de cada linha da tabela como:

- - um valor de ponto fixo de seis caracteres com duas casas decimais
- - dois espaços
- - um valor de ponto fixo de doze caracteres com oito casas decimais

15.5 Lendo Arquivos de Texto Formatados e Strings

A função de entrada de texto do MATLAB, fscanf, é similar à função fprintf. fscanf possui como argumento o identificador para o arquivo de texto aberto, e uma string de controle do formato contendo caracteres e especificadores de conversão, nesta ordem. Os especificadores para fscanf são precedidos pelo caractere %; conversões comuns incluem

- %s para converter uma string
- %d para converter um número decimal
- %f para converter um valor em ponto flutuante

O exemplo a seguir faz a leitura do arquivo com os dados exponenciais escrito anteriormente:

```
>> fid = fopen('exptable.txt','r');
>> title = fscanf(fid,'%s');
>> [table,count] = fscanf(fid,' %d %f ');
>> status = fclose(fid);
```

A linha do título combina com o especificador %s na primeira chamada à fscanf. A segunda chamada entra com a tabela de valores, combinando alternadamente um valor decimal e um valor em ponto flutuante, como é lido em cada linha da tabela, até que o fim do arquivo seja atingido. count retorna o número de valores combinados.

Um argumento opcional controla o número de elementos lidos da matriz. Por exemplo, se fid faz referência a um arquivo aberto contendo os strings de decimal então

```
>> A = fscanf(fid,'%5d',100);
```

faz a leitura de 100 valores decimais para um vetor coluna A, e

```
>> A = fscanf(fid,'%5d',[10,10]);
```

faz a leitura de 100 valores decimais para uma matriz A 10x10.

PARTE III

Gráficos Avançados

A última parte traz o capítulo 16 que contém a parte avançada de gráficos.

São descritas as funções especiais para gráficos 2-D e a introdução e as funções especiais para gráficos 3-D. Os recursos de alteração de cores e plotagem de mais de um gráfico em uma mesma figura também são mostrados. Por fim, são explicados os objetos de nível inferior, essenciais ao entendimento da programação no MATLAB.

Capítulo 16

Gráficos Avançados

O sistema gráfico do MATLAB incorpora uma variedade de técnicas sofisticadas para apresentação e visualização de dados. Este sistema é construído através de um conjunto de objetos gráficos, tais como linhas e superfícies, cuja presença pode ser controlada pelo ajuste dos valores das propriedades de cada objeto. Entretanto, em virtude do MATLAB possibilitar um rico ajuste nas funções gráficas 2-D e 3-D, na maior parte do tempo não é necessário acessar estes objetos gráficos, os quais são considerados objetos de baixo nível.

16.1 Gráficos 2-D

16.1.1 Importando Dados

MATLAB permite que sejam importados e plotados dados gerados fora dele. Suponhando um arquivo, weather.dat, contendo a média de temperatura e precipitação dos meses, armazenadas em doze linhas de um texto em ASCII:

```
30 4.0
31 3.7
38 4.1
49 3.7
59 3.5
68 2.9
74 2.7
72 3.7
65 3.4
55 3.4
45 4.2
34 4.9
```

O comando,

```
>> load weather.dat
```

produz uma matriz 12 por 2. A linha de comando

```
>> temp = weather(:,1);
>> precip = weather(:,2);
```

armazena as colunas de temperatura e precipitação em vetores individuais.

Pode-se plotar tanto a temperatura como a precipitação em função do mês correspondente, em uma mesma figura, fazendo-se uso dos comandos plot e subplot:

```
>> subplot(2,11)
>> plot(temp)
>> subplot(2,1,2)
>> plot(precip)
```

Um gráfico scatter também é um modo útil de se apresentar os dados. As linhas de comando a seguir produzem um gráfico scatter mostrando a relação entre a temperatura e a precipitação em cada mês:

```
>> month = ['Jan'; 'Feb'; 'Mar'; 'Apr'; ...
'May'; 'Jun'; 'Jul'; 'Aug'; ...
'Sep'; 'Oct'; 'Nov'; 'Dec'];
>> plot(temp,precip,'o')
>> axis([28 80 2.5 5.2])
>> xlabel('temp')
>> ylabel('precip')
>> title('Boston')
```

A linha de comando axis, do exemplo, acima introduz espaço extra ao gráfico através do ajuste da escala dos eixos a valores maiores que a faixa de dados. Isto permite que o texto fique dentro do gráfico.

16.1.2 Funções Especiais para Gráficos 2-D

MATLAB inclui uma variedade de funções especiais para gráficos, necessárias em muitas aplicações. A lista a seguir descreve algumas delas.

- bar - cria um gráfico do tipo barra
- compass - cria um gráfico de ângulos e módulos de números complexos onde a representação dos pontos é feita por setas com início na origem
- errorbar - cria um gráfico com barras de erro
- feather - cria um gráfico de ângulos e módulos de números complexos onde a representação dos pontos é feita por setas com começo em pontos igualmente espaçados ao longo do eixo horizontal
- fplot - avalia uma função e plota os resultados
- hist - cria um histograma
- polar - cria um gráfico em coordenadas polares dos ângulos em função dos raios
- quiver - cria um gráfico do gradiente ou de outro campo do vetor
- rose - cria um histograma de ângulo
- stairs - cria um gráfico similar a um gráfico de barra, mas sem as linhas internas
- fill - desenha um polígono e o preenche com cores sólidas ou interpoladas

16.1.3 Polígonos Preenchidos

A função fill define uma área polinomial cujo interior é preenchido com uma cor especificada. Se é especificada uma cor para cada ponto, MATLAB utiliza interpolação bilinear para determinar a cor da parte interna. Por exemplo,

```
>> t = 0:.05:2*pi;
>> x = sin(t);
>> fill(x,t,'b')
```

produz um polígono preenchido pela cor azul.

Para criar uma área preenchida com uma interpolação de cores, utiliza-se o comando colormap da forma descrita abaixo:

```
>> colormap(hot)
>> fill(x,t,x)
```

A sombra varia do escuro para o claro à medida que o valor de sin(t) varia de um mínimo até um máximo (o que corresponde da esquerda para a direita no gráfico).

16.1.4 Plotando Funções Matemáticas

Existem várias maneiras diferentes de se plotar os gráficos de funções, $y = f(x)$. Uma aproximação que simplesmente avalia a função em alguns pontos no intervalo de interesse é descrita pelas linhas de comando abaixo. No caso, a função oscila infinitamente

no intervalo .

```
>> x = (0:1/2000:1)';
>> plot(x,cos(tan(pi*x)))
```

Essas linhas de comandos produzem o gráfico da função na faixa de 0 a 1.

A função fplot é uma aproximação mais elegante, visto que ela amostra a função com um número suficiente de pontos que forneça uma representação gráfica decente. Enquanto que a representação anterior avaliava a função em intervalos espaçados constantemente, fplot concentra a sua avaliação sobre regiões onde a taxa de variação da função é maior. Para avaliar uma função, deve-se criar um arquivo função e passar o nome do arquivo para fplot. Por exemplo, o arquivo função a seguir define uma função, chamada fofx, do exemplo anterior:

```
function y = fofx(x)
y = cos(tan(pi*x));
```

Passa-se o nome desta função para fplot para que seja avaliada e plotada. Em virtude da função oscilar infinitamente na faixa de avaliação, o número de interações usadas para preencher a região onde ocorrem as rápidas variações é reduzido com relação ao ajuste padrão para prevenir de um tempo de computação excessivamente longo.

```
>> fplot('fofx', [0 1], 25, 20, 10)
```

Neste exemplo, fplot utiliza menos pontos para avaliar a mesma função, mas agora amostra a função em intervalos menores em uma região onde a taxa de variação é a maior, gerando uma figura mais precisa nas proximidades de x = 0.5.

16.2 Gráficos 3-D

MATLAB possui uma variedade de funções para exibir dados em 3-D. Algumas plotam linhas em três dimensões, enquanto outras desenham superfícies e figuras utilizando "pseudocores" para representar uma quarta dimensão. A lista a seguir mostra essas funções:

- plot3 - plota linhas e pontos em 3-D
- contour, contour3 - cria gráficos de contorno
- pcolor - desenha uma matriz retangular de células cujas cores são determinadas pelos elementos da matriz
- image - mostra uma matriz como uma imagem através do mapeamento dos elementos da matriz pelo mapa de cores ativo
- mesh, meshc, meshz - cria gráficos 3-D em perspectiva dos elementos da matriz, mostrados como alturas acima de um plano delimitado
- surf, surfc, surfl - cria gráficos 3-D em perspectiva dos elementos da matriz, mostrados como alturas acima de um plano delimitado. O gráfico é formado pela geração de uma superfície colorida utilizando os pontos como vértices de um quadrilátero.

- fill3 - cria um polígono 3-D a o preenche com cores sólidas ou interpoladas

Obs.: Será utilizado o termo "superfíci e mesh" para indicar os gráficos criados através do comando mesh.

Além das funções de inserção de comentários e notas mostradas para os gráficos 2-D, MATLAB possui as seguintes funções para os gráficos 3-D:

- zlabel - define um nome para a variável do eixo z
- clabel - define identificações para os contornos em gráficos de contorno

MATLAB permite a especificação do ponto de visualização do gráfico. Em geral, as vistas são definidas por uma matriz de transformação 4 por 4 que o MATLAB utiliza para transformar um gráfico 3-D em uma tela 2-D. Entretanto, a função abaixo permite a especificação do ponto de visão de uma maneira simplificada:

- view - ajusta o ponto de visão atual onde os parâmetros de entrada podem ser o azimute (rotação horizontal) e a elevação vertical, ambos em grau, ou as coordenadas cartesianas.

16.2.1 Gráficos de Linhas

O análogo tridimensional da função plot é a função plot3. Se x, y e z são vetores de mesmo tamanho,

```
>> plot3(x,y,z)
```

gera uma linha no espaço tridimensional que passa pelos pontos de coordenadas dadas pelos elementos dos vetores x, y e z, e depois produz uma projeção bidimensional desta linha na tela.. Por exemplo

```
>> t = 0:pi/50:10*pi;
>> plot3(sin(t),cos(t),t);
```

produz um helicóide.

Se X, Y e Z são três matrizes de mesma dimensão,

```
>> plot3(X,Y,Z)
```

plota as linhas obtidas pelas colunas de X, Y e Z. Para especificar vários tipos de linhas, símbolos gráficos e cores, usa-se a linha de comando plot3(X,Y,Z,s), onde s é uma string de 1, 2 ou 3 caracteres formada pelos caracteres listados com a função plot.

```
>> plot3(x1,y1,z1,s1,x2,y2,z2,s2,x3,y3,z3.s3,...)
```

combina os gráficos definidos por cada grupo de elementos (x, y, z, s), onde x, y e z são vetores ou matrizes e s são as strings.

16.2.2 Meshgrid

MATLAB define uma superfície mesh pelas coordenadas z dos pontos, situando-a acima de uma área retangular no plano x-y. Desta forma, um gráfico é formado pela ligação dos pontos adjacentes com linhas retas. Superfícies mesh são úteis para visualização de matrizes que são muito grandes para serem mostradas na forma numérica, e funções gráficas de duas variáveis.

O primeiro passo para que seja mostrada uma função de duas variáveis, *z = f(x,y)*, é gerar as matrizes X e Y consistindo de linhas e colunas repetidas, respectivamente, sobre o domínio da função. Depois utiliza-se estas matrizes para avaliar e plotar a função.

A função meshgrid transforma o domínio especificado por dois vetores, x e y, em matrizes, X e Y. Utiliza-se estas matrizes para avaliar funções de duas variáveis. As linhas de X são cópias do vetor x e as colunas de Y são cópias do vetor y. Para ilustrar o uso da função meshgrid, considere a função sin®/r, também chamada função sinc. Para avaliar esta função entre -8 e 8, tanto em x como em y, passa-se como argumento dois vetores para a função meshgrid criando as matrizes necessárias:

```
>> x = -8:.5:8;
>> y = x;
>> [X,Y] = meshgrid(x,y);
>> R = sqrt(X.^2 + Y.^2) +eps;
>> Z = sin®./R;
>> mesh(Z)
```

A matriz R contém a distância dos pontos ao centro da matriz, o qual é a origem. Adiconando eps evita a divisão por zero, o que produz NaNs no dados.

16.2.3 Gráficos de Contorno

MATLAB possui funções para gerar gráficos de contorno tanto para 2-D como para 3-D. As funções contour e contour3 geram gráficos compostos de linhas obtidas da matriz entrada como argumento. Existe a opção de se especificar o número de linhas de contorno, a escala dos eixos e o valor dos dados com os quais se desenhará as linhas de contorno. Por exemplo, a linha de comando abaixo cria um gráfico de contorno contendo 20 linhas de contorno e usando o arquivo-M peaks para gerar os dados de entrada.

```
>> contour(peaks,20)
```

Os círculos de contorno agem da mesma forma que a função plot com relação a estilos de linha, marcadores e cores. Para criar um gráfico de contorno 3-D com os mesmo dados, utiliza-se a função contour3:

```
>> contour3(peaks,20)
```

16.2.4 Gráficos de Pseudocores

As linhas de comando,

```
>> z = peaks;
>> pcolor (z)
>> colormap(hot)
```

criam uma figura com a matriz peaks definida por um mapa de cores. O nome pcolor significa pseudocolor. Para cada ponto, z(i,j), um valor em escala do elemento da matriz, z(i,j), é usado como índice no mapa de cores para determinar a cor a ser mostrada naquele ponto.

O mapa de cores é uma matriz com três colunas que especificam a intensidade das três componentes de vídeo: vermelho, verde e azul. Neste caso, o mapa de cores especificado mapeia o menor dado como preto e o maior dado como branco. Os valores entre estes dois valores são mapeados com sombras de vermelho, laranja e amarelo. Isto é útil pois a escala contínua de cores do preto ao branco faz representar o contorno da função peaks. Os possíveis mapas de cores podem ser obtidos através da linha de comando help color.

16.2.5 Gráficos de Superfícies e Mesh

As funções mesh e surf mostram superfícies em três dimensões. Se Z é uma matriz cujo os elementos Z(i,j) definem a altura da superfície com relação a uma área delimitada (i,j), então

```
>> mesh(Z)
```

gera uma figura da superfície, colorida e definida por linhas, e a mostra em uma projeção em perspectiva. Da mesma forma

```
>> surf(Z)
```

gera uma figura da superfície, colorida e definida por faces, e a mostra em uma projeção em perspectiva. Ordinariamente, as faces são quadriláteros com cores constantes, delineadas por linhas mesh pretas, mas a função shading permite que sejam eliminadas estas linhas.

Quando mesh(Z) e surf(Z) são usados com simples matrizes como argumentos, este argumento especifica tanto a altura quanto as cores da superfície. Para a matriz peaks utilizada anteriormente

```
>> mesh(peaks)
```

produz uma superfície mesh para a função peaks

Matrizes de Cores

Utilizando duas matrizes como argumentos, as linhas de comando,

```
>> mesh(Z,C)
```

ou

```
>> surf(Z,C)
```

especifica independentemente através do segundo argumento, as cores a serem mostradas na superfície. Com pcolor©, os valores de C são postos em escala e usados como índices no mapa de cores selecionado. No caso, C é chamada matriz de cores.

Quando a superfície de dados contém NaNs, estes elementos não são plotados. Isto cria um falha na superfície no lugar correspondente. Definindo alguns elementos na matriz de cores como NaNs, é uma maneira útil de tornar regiões da superfície invisíveis. Por exemplo, a linha de comando abaixo produz uma falha na superfície definida pela matriz peaks:

```
>> p = peaks;
```

```
>> p(30:40,20:30) = nan*p(30:40,20:30);
>> mesh(peaks,p)
```

Superfícies Paramétricas

As funções mesh, surf e pcolor podem ter dois vetores ou matrizes adicionais como argumento, com a função de descrever a superfície com os dados x e y. Se Z é uma matriz m por n, e x e y são vetores de dimensões n e m respectivamente, então

```
>> mesh(x, y, Z, C)
```

descreve uma superfície mesh cujo os vértices possuem cores C(i,j) e estão localizados nos pontos (x(j), y(i), Z(i, j)) Perceba que x corresponde à coluna e y à linha da matriz Z.. De maneira mais geral, se X, Y, Z e C são matrizes de mesmas dimensões, então

```
>> mesh(X, Y, Z, C)
```

descreve uma superfície mesh cujo os vértices possuem cores C(i,j) e estão localizados no pontos (X(i,j), Y(i,j), Z(i,j))

As mesmas considerações podem ser aplicadas para a função surf(X, Y, Z, C), etc.

Variações de surf *e* mesh

MATLAB possui funções que produzem combinações nos gráficos baseados nas funções surf e mesh. Por exemplo, surfc desenha uma superfície com contorno bidimensional no plano z = 0. Da mesma forma, existem as funções surfl, meshc e meshz.

16.3 Funções Gráficas de Propósito Geral

Serão descritas abaixo funções para controle de aparência dos gráficos criados com as funções descritas anteriormente.

16.3.1 Ponto de Vista

MATLAB permite que seja especificado o ângulo do qual visualiza-se um gráfico 3-D. A função view define o ângulo de visão em coordenadas esféricas através da especificação do azimute (rotação horizontal) e da elevação vertical do ponto de vista, com relação a origem dos eixos. O azimute é um ângulo polar no plano *x-y*, sendo positivo quando a rotação for no sentido horário com relação ao ponto de vista. A elevação vertical é o ângulo acima (ângulo positivo) ou abaixo (ângulo negativo) do plano *x-y*. Por exemplo, as quatro linhas de comando abaixo proporcionam quatro maneiras diferentes de se visualizar a função peaks.

```
>> view(-37.5,30)
>> view(-7,80)
>> view(-90,0)
>> view(-7,-10)
```

16.3.2 Controlando os Eixos com a função axis

A função axis possui várias opções que permitem que sejam personalizados escala e orientação. Ordinariamente, MATLAB encontra os valores de máximo e mínimo da função a ser plotada e escolhe uma área de plotagem apropriada. Pode-se redefinir os limites pelo ajuste dos limites do eixos:

```
>> axis([xmin xmax ymin ymax])
```

Ou para um gráfico 3-D:

```
>> axis([xmin xmax ymin ymax zmin zmax])
```

axis('auto') retorna a escala do eixo para seu valor padrão, o modo automático. v = axis salva a escala dos eixos do gráfico que está ativo no vetor v. Para que gráficos subsequentes possuam os mesmos limites, basta entrar com o comando axis(v).

axis(axis) congela a escala nos valores que estão sendo usados.

axis('ij') define MATLAB no seu modo de eixo matriz. A origem do sistema de coordenadas se encontra no canto superior esquerdo.

O eixo i é vertical e é numerado de cima para baixo. o eixo j é horizontal e é numerado da esquerda para a direita.

axis('xy') define MATLAB com os eixos cartesianos. A origem do sistema de coordenadas se encontra no canto inferior esquerdo. O eixo x é horizontal e é numerado da esquerda para a direita. O eixo y é vertical e é numerado de baixo para cima.

axis('on') e axis('off') fazem com que o nome do eixo e os marcadores apareçam ou não, respectivamente, junto com o gráfico.

16.3.3 Tornando Visível Linhas e Superfícies Escondidas

O comando hidden permite que se enxergue através de uma superfície de modo a se poder visualizar possíveis figuras que estejam atrás desta superfície. Este comando se faz útil quando estão sendo plotadas várias figuras em um único gráfico.

Por exemplo, as linhas de comando a seguir fazem com que os dados gerados por pcolor possam ser visualizados através do gráfico de peaks.

```
>> mesh(peaks(20)+7)
>> hold
>> pcolor(peaks(20))
>> hidden off
```

16.3.4 Subgráficos

Podem ser mostrados vários gráficos em uma mesma janela ou imprimi-los em uma mesma folha de papel com a função subplot.

subplot(m,n,p) divide a janela em uma matriz m por n de subregiões e seleciona a p-ésima subregião para o gráfico que está sendo plotado no momento.

Os gráficos são numerados começando pela primeira linha na parte superior da janela; em seguida, a segunda linha, e assim segue. Por exemplo,

```
>> t = 0:pi/10:2*pi;
>> [X,Y,Z] = cylynder(4*cos(t));
>> subplot(2,2,1)
>> mesh(X)
>> subplot(2,2,2)
>> mesh(Y)
>> subpoot(2,2,2)
>>mesh(Y)
>> subplot(2,2,3)
>> mesh(Z)
>> subplot(2,2,4)
>> mesh(X,Y,Z)
```

plota dados em quatro diferentes subregiões na janela de gráficos.

16.3.5 Figura

Chamando a função figure sem argumentos faz com que seja aberto uma nova janela gráfica.

figure(N) faz com que a N-ésima figura se torne a figura atual; o resultado dos comandos gráficos subsequentes são mostrados nesta janela. Se a figura N não existe, MATLAB cria uma usando o próximo número disponível (não necessariamente N).

A função ginput permite que sejam utilizados o mouse ou as teclas de seta para selecionar pontos no gráfico. Isto fornece as coordenadas da posição do ponteiro.

16.4 Mapas de Cores e Controle de Cores

MATLAB define o mapa de cores como sendo uma matriz de três colunas. Cada linha da matriz define uma cor particular através de três valores dados na escala de 0 a 1. Estes valores especificam a intensidade das componentes de vídeo azul, verde e vermelho.

Algumas cores estão listadas na tabela abaixo:

Vermelho	Verde	Azul	Cor

0	0	0	preto
1	1	1	branco
1	0	0	vermelho
0	1	0	verde
0	0	1	azul
1	1	0	amarelo
1	0	1	magenta
0	1	1	ciano
.5	.5	.5	cinza
.5	0	0	vermelho escuro

Os mapas de cores podem ser fornecidos por tabelas, ou gerados em operações de matrizes. O diretório color do toolbox do MATLAB possui várias funções que geram mapas úteis, incluindo hsv, hot, cool, pink, copper e flag. Cada função possui um parâmetro opcional, m, o qual especifica o número de linhas no mapa resultante. Por exemplo

```
>> hot(m)
```

é uma matriz m por 3, cuja as linhas especificam a intensidade das cores padrões de um mapa que varia de preto, passando por sombras de vermelho, laranja e amarelo, até o branco.

Se o tamanho do mapa de cores não é especificado, MATLAB utiliza m = 64 como valor padrão. Isto permite três ou quatro figuras, onde cada uma possui sua tabela de 256 cores.

A linha de comando,

```
>> colormap(M)
```

faz com que a matriz de mapa de cores M seja usada pela figura ativa. Por exemplo,

```
>> colormap(hot)
```

torna ativo o mapa de cores hot de tamanho 64. Desde que as funções mesh, surf, pcolor e image utilizam mapas de cores, todas as outras funções que são derivadas destas quatro também o fazem. As funções plot, plot3, contour e contour3 não fazem uso dos mapas de cores.

16.4.1 Mostrando Mapas de Cores

As funções pcolor são úteis para mostrar mapas de cores. Por exemplo, para visualizar um mapa de cores em tons de cinza tendo oito entradas, utiliza-se as seguintes linhas de comando:

```
>> colormap(gray(8))
>> pcolor([1:9;1:9]')
```

16.4.2 Alterando os Mapas de Cores

O fato de os mapas de cores serem matrizes, permite que sejam manipulados como qualquer outra matriz. A função brighten utiliza esta vantagem para ajustar um dado mapa de cores de forma que a intensidade das cores escuras aumente ou diminua.

Pode-se combinar mapas de cores aritmeticamente, embora os resultados sejam inesperados em algumas vezes.

16.5 Manuseamento de Gráficos

As características gráficas apresentadas até aqui fazem parte da interface de alto-nível do sistema gráfico do MATLAB. Entretanto, este sistema também fornece um ajuste das funções de baixo-nível que permitem que sejam criadas e manipuladas linhas, superfícies e outros objetos gráficos que MATLAB utiliza para produzir gráficos sofisticados. Este sistema é denominado *Handle Graphics*.

16.5.1 Objetos Gráficos

MATLAB define os objetos gráficos como sendo os desenhos mais primitivos de seu sistema gráfico, e os organiza em uma hierarquia estruturada em árvore. Este objetos incluem a tela raiz (root screen), figuras (figures), eixos (axes), linhas (lines), patches (patches), superfícies (surfaces), imagens (images), texto (text), e interface de controle (uicontrol) e menus (uimenu) para o usuário.

- O objeto *root* é a raiz da hierarquia. Ele corresponde a tela do computador. Existe somente um objeto raiz, sendo todos os outros objetos seus descendentes.
- Objetos *figure* correspondem às janelas individuais na tela raiz. Pode existir qualquer número de objetos figura, onde cada qual é descendente do objeto raiz. Os demais objetos gráficos são descendentes das janelas de figura. Todos as funções de criação de objetos e todas as funções de alto-nível criam uma figura se ela não existe. Pode-se também criar uma figura utilizando-se a função figure.
- Objetos *axes* definem uma região na janela de figura e orienta os seus descendentes nesta região. Eixos são descendentes de figuras e são superiores a linhas, superfícies, texto, imagens e patches. Todos as funções de criação de objetos e todas as funções de alto-nível criam objetos eixo se eles não existem. Pode-se criar eixos diretamente utilizando-se a função axes.
- Objetos *line* são usados para criar a maioria dos gráficos 2-D e alguns 3-D. Eles são descendentes dos eixos e suas posições são definidas pelo sistema de coordenadas estabelecido pelo seus superiores (eixos). Objetos linha são criados por plot, plot3, contour e contour3.
- Objetos *patch* são definidos pelos polígonos preenchidos. Eles são descendentes dos eixos e suas posições são definidas pelo sistema de coordenadas estabelecido pelo seus superiores (eixos). Este objetos podem ser preenchidos com cores sólidas ou interpoladas. fill e fill3 cria objetos patch.
- Objetos *surface* são representações 3-D dos dados de uma matriz. Eles são compostos de quadriláteros cujo os vértices são especificados pelos dados definidos. Superfícies podem ser preenchidas com cores sólidas ou interpoladas ou somente com uma rede de linhas ligando os pontos. Eles são descendentes dos eixos e suas posições são definidas pelo sistema de coordenadas estabelecido pelo seus superiores (eixos). pcolor e o grupo de funções de mesh e surf criam os objetos superfície.
- Objetos *image* é dado pelo resultado do mapeamento dos elementos de uma matriz com o mapa de cores se encontra ativo. Imagens, geralmente, possuem seu próprio mapa de cores que definem somente as cores usadas naquela imagem. Imagens são 2-D e, portanto, não podem ser vistas por nenhum ângulo diferente do padrão. Eles são descendentes dos eixos e suas posições são definidas pelo sistema de coordenadas estabelecido pelo seus superiores (eixos). Objetos imagem são criados através da função image.
- Objetos *text* são as strings de caracteres. Eles são descendentes dos eixos e suas posições são definidas pelo sistema de coordenadas estabelecido pelo seus superiores (eixos).
- Objetos *uicontrol* são interfaces de controle que permitem ao usuário executar funções quando é selecionado um objeto com o mouse. Eles são descendentes das figuras e, portanto, são independentes dos eixos.
- Objetos *uimenu* são interfaces de menu que permitem o usuário criar menus na parte superior da janela de figura. Eles são descendentes das figuras e, portanto, independentes dos eixos.

Handle de Objetos

Cada objeto gráfico individual possui um único identificador (chamado handle) que é atribuído ao objeto quando ele é criado. Alguns gráficos tais como contour plot, são compostos de múltiplos objetos, cada qual com seu próprio handle. O handle do objeto raiz é sempre zero enquanto que o de uma figura é um inteiro. O handle de outros objetos são números decimais que contém informações utilizadas pelo MATLAB. MATLAB define as seguintes funções para simplificar o acesso aos handles dos objetos:

- gcf - informa o handle da figura ativa
- gca - informa o handle do sistema de eixos ativo

Pode-se utilizar estas funções como argumentos de entrada de outras funções que requeiram os handles de gráficos e eixos. Pode-se, também, remover qualquer objeto utilizando a função delete, descrevendo como argumento o handle do objeto. Por exemplo, os eixos atuais podem ser apagados através da linha de comando:

```
>> delete(gca)
```

Todas as funções do MATLAB que criam objetos, criam handles para estes objetos. Isto inclui funções de alto-nível, tais como surfc, a qual cria tanto objeto linha como objeto superfície.

Funções de Criação de Objetos

Todos os objetos são criados por funções de mesmo nome (a função text cria um objeto texto, a função figure cria um objeto figura, etc.). As funções gráficas de alto-nível do MATLAB chamam a função de baixo-nível apropriada para desenhar seus respectivos gráficos.

Muitas funções de alto-nível ajustam as propriedades do objeto com a finalidade de produzirem um resultado particular. As funções de baixo-nível simplesmente criam um dos nove objetos gráficos definidos pelo MATLAB (não se pode criar um novo objeto raiz) e os coloca no objeto-mãe apropriado. Por exemplo, chamando a função line,

```
>> line
```

MATLAB desenha a linha no eixo que está ativo usando os valores de dado padrão. Se não houver eixo, MATLAB cria um. Se não houver uma janela de figura onde possam ser criados os eixos, MATLAB também cria uma janela. Se é chamada a função line pela segunda vez, uma linha é desenhada nos eixos existentes (diferentemente com relação a função plot, por exemplo, a qual substitui os eixos em cada chamada). Esta característica é útil quando se quer adicionar um objeto a uma figura existente. Pode-se obter o mesmo resultado utilizando o comando hold.

16.5.2 Propriedades dos Objetos

Todos os objetos possuem propriedades que definem como eles são mostrados. Estas propriedades incluem informações gerais tais como o tipo do objeto, seus descendentes (children) e superiores (parent), o que está ou não visível, bem como informações únicas de um determinado objeto tais como os dados utilizados para se determinar um objeto superfície.

Quando se cria um objeto, ele é inicializado com os valores padrões; destes, alguns podem ser alterados (outros são definidos pelo MATLAB como sendo somente para leitura).

Especificando e Alterando as Propriedades dos Objetos

MATLAB possui dois modos de ajuste dos valores das propriedades. Pode-se especificar as propriedades do objeto quando se chama a função de criação do objeto, ou depois que o objeto foi criado fazendo uso da função set. Por exemplo, a linha de comando,

```
>> figh = figure('Color','white')
>> axh = axes('View',[37.530],'XColor','k','YColor','k','ZColor','k')
>> surfh = surface(peaks,,'FaceColor','none','LineStyle','.')
```

cria três objetos e atribui valores às propriedades onde os valores padrões não são desejados. Para o caso acima, a janela de figura criada possui fundo branco e as linhas dos eixos são definidas como sendo pretas ('k'). O ângulo de visão é definido como tendo um azimute de -37.5° e uma elevação de 30°. Os dados para que a superfície seja gerada são fornecidos pela função peaks, e esta superfície é definida por pontos, não possuindo nenhuma cor dentro dos quadriláteros formados por estes pontos.

Utilizando as Funções set *e* get

Uma outra forma de se especificar as propriedades dos objetos é referenciando este objeto depois de sua criação. Pode-se fazer isto utilizando o handle que é retornando pela função de criação.

A função set permite que seja ajustada qualquer propriedade do objeto pela especificação do handle do mesmo e pelo par propriedade/valor. Por exemplo, o exemplo anterior define uma superfície e salva seu handle em surfh. Pode-se alterar a propriedade do objeto referente ao estilo de linha (LineStyle), de uma linha pontilhado para uma linha sólida, através da linha de comando:

```
>> set(surfh,'LineStyle','-')
```

Para se ter uma lista das propriedades que podem ser alteradas de um objeto particular, chama-se a função set com o handle do objeto:

```
>> set(surfh)
```

Para se fazer a alteração do valor de uma propriedade, utiliza-se a função get.

Chamando-se a função get com o handle do objeto, obtêm-se a lista de todas as propriedades para aquele objeto com o valores atuais:

```
>> get(surfh)
```