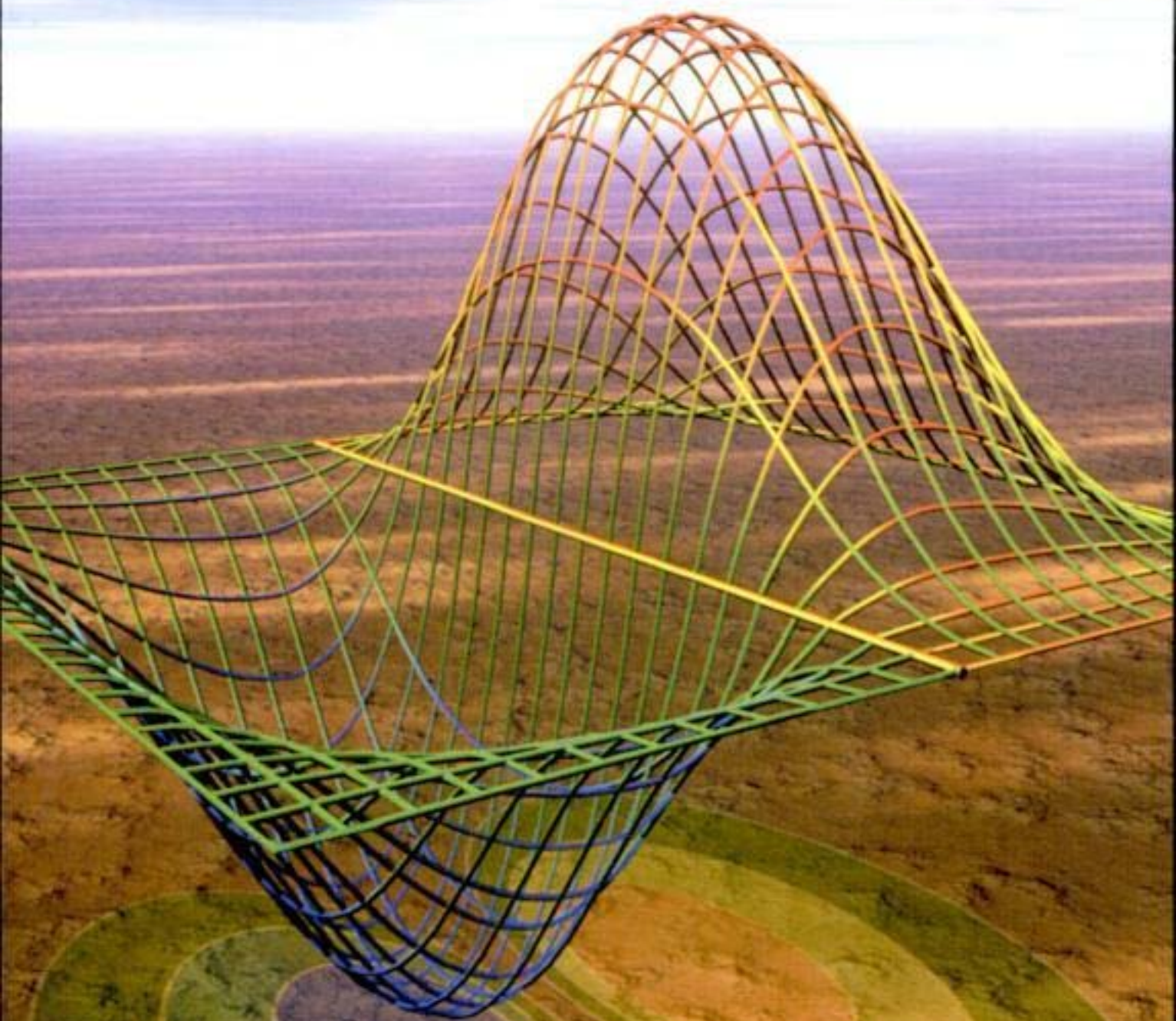


SEGUNDA EDIÇÃO

MATLAB®

COM APLICAÇÕES EM ENGENHARIA



Amos Gilat



Copyrighted material

**ENGENHARIA
ELÉTRICA/ELETRÔNICA**

ALEXANDER & SADIKU

Fundamentos de Circuitos Elétricos

CATHEY, J. J.

Dispositivos e Circuitos Eletrônicos, 2.ed.

(Coleção Schaum)

DINIZ, SILVA & NETTO

Processamento Digital de Sinais: Projeto e

Análise de Sistemas

EDMINISTER, J.

Eletromagnetismo, 2.ed. (Coleção Schaum)

ERCEGOVAC, LANG & MORENO

Introdução aos Sistemas Digitais

FITZGERALD, KINGSLEY & UMANS

Máquinas Elétricas com Introdução à Eletrônica

de Potência, 6.ed.

FLOYD, T.

Sistemas Digitais, 9.ed.

FUSCO, E.

Teoria e Técnica de Antenas: princípios e prática

GILAT, A.

MATLAB com Aplicações em Engenharia, 2.ed.

GUSSOW, M.

Elettricidade Básica, 2.ed. (Coleção Schaum)

HAYES, M.

Processamento Digital de Sinais (Coleção

Schaum)

HAYKIN, S.

Sistemas de Comunicação Analógicos e Digitais,

4.ed.

HAYKIN & MOHER

Comunicações Sem Fio

HAYKIN & MOHER

Introdução aos Sistemas de Comunicação, 2.ed.

HAYKIN & VAN VEEN

Sinais e Sistemas

HSU, H. P.

Comunicação Analógica e Digital, 2.ed. (Coleção

Schaum)

HSU, H. P.

Sinais e Sistemas (Coleção Schaum)

LATHI, B. P.

Sinais e Sistemas Lineares, 2.ed.

MUELLER & SOPER

PCs: Atualização e Manutenção, Guia Prático

NAHVI & EDMINISTER

Circuitos Elétricos, 4.ed. (Coleção Schaum)

PERTENCE JÚNIOR, A.

Eletrônica Analógica: Amplificadores

Operacionais e Filtros Ativos, 6.ed.

SADIKU, M.

Elementos de Eletromagnetismo, 3.ed.

ULABY, F.

Eletromagnetismo para Engenheiros

VAHID, F.

Sistemas Digitais: Projeto, Otimização e HDLs

***WENTWORTH, S.**

Eletromagnetismo Aplicado



Bookman Editora

Av. Jerônimo de Ornelas, 670

90040-340 Porto Alegre, RS, Brasil

Fone (51) 3027-7000 Fax (51) 3027-7070

e-mail: bookman@artmed.com.br

* Livro em produção no momento da impressão
desta obra, mas que muito em breve estará à
disposição dos leitores em língua portuguesa.

MATLAB®
COM APLICAÇÕES EM ENGENHARIA



G463m

Gilat, Amos

MATLAB com aplicações em engenharia / Amos Gilat ; tradução Glayson Eduardo de Figueiredo. – 2. ed. – Porto Alegre : Bookman, 2006.

360 p. : il. p&b ; 25 cm.

ISBN 978-85-363-0692-6

1. Computação – Programas – MATLAB. 2. MATLAB – Engenharia.

I. Título.

CDU 004.4:62MATLAB

Catálogo na publicação: Júlia Angst Coelho – CRB 10/1712

Amos Gilat

Department of Mechanical Engineering
The Ohio State University

SEGUNDA EDIÇÃO

MATLAB®

COM APLICAÇÕES EM ENGENHARIA

Tradução:

Glaysen Eduardo de Figueiredo
Bacharel em Física pela UFMG
Mestrando em Engenharia Elétrica (UFMG)

Consultoria, supervisão e revisão técnica desta edição:

Antonio Pertence Júnior
Engenheiro Eletrônico e de Telecomunicações
Especialista em Processamento de Sinais (Ryerson University - Canadá)
Professor de Telecomunicações da FUMEC (MG)
Professor Titular da Faculdade de Sabará/MG

Reimpressão 2008



2006

This One



DHC5-ZNB-UJDU

Unauthorized Material

Obra originalmente publicada sob o título
MATLAB® An Introduction with Applications

© 2005 John Wiley & Sons, Inc.

All rights reserved.

Tradução autorizada da edição em inglês publicada por John Wiley & Sons, Inc.

ISBN 0-471-69420-7

Capa: *Gustavo Demarchi*, arte sobre capa original

Leitura final: *Walson Pontes Carpes*

Supervisão editorial: *Denise Weber Nowaczyk*

Editoração eletrônica: *Laser House*

MATLAB é marca registrada de The Math Works, Inc.

Reservados todos os direitos de publicação, em língua portuguesa, à
ARTMED® EDITORA S. A.
(BOOKMAN® COMPANHIA EDITORA é uma divisão da ARTMED® EDITORA S.A.)
Av. Jerônimo de Ornelas, 670 - Santana
90040-340 Porto Alegre RS
Fone (51) 3027-7000 Fax (51) 3027-7070

É proibida a duplicação ou reprodução deste volume, no todo ou em parte,
sob quaisquer formas ou por quaisquer meios (eletrônico, mecânico, gravação,
fotocópia, distribuição na Web e outros), sem permissão expressa da Editora.

SÃO PAULO
Av. Angélica, 1091 - Higienópolis
01227-100 São Paulo SP
Fone (11) 3665-1100 Fax (11) 3667-1333

SAC 0800 703-3444

IMPRESSO NO BRASIL
PRINTED IN BRAZIL.

Aos meus pais Schoschana e Haim Gelbwacks

Prefácio*

O MATLAB® é um *software* bastante popular em computação técnica e científica; é usado no mundo inteiro por estudantes, engenheiros e cientistas em universidades, institutos de pesquisa e indústrias. A razão da popularidade deve-se ao poder e à facilidade de utilização desse programa. Os calouros podem pensar nele como a próxima ferramenta a ser utilizada após a calculadora científica, tão comum no Ensino Médio.

Este livro foi escrito seguindo um roteiro de muitos anos no ensino do *software* numa disciplina introdutória para calouros do curso de engenharia. O objetivo era escrever um livro que ensinasse o *software* de um modo amigável, sem intimidações. Assim, foi utilizada uma linguagem simples, informal e direta. Em muitas oportunidades foram utilizados vários recursos didáticos para explicar e detalhar tópicos específicos. O livro incorpora inúmeros exemplos da matemática, ciências e engenharia que os iniciantes do MATLAB® encontram nos cursos introdutórios.

Esta segunda edição foi atualizada para incorporar recursos do MATLAB 7. Traz também um capítulo novo (11) sobre operações envolvendo matemática simbólica e uma nova seção no Capítulo 4 que mostra como importar e exportar dados.

Gostaria de expressar meus agradecimentos a muitos dos meus colegas da The Ohio State University. Meu reconhecimento aos Professores Richard Freuler, Mark Walter, Brian Harper e ao Dr. Mike Parke pela leitura e sugestões abrangendo as várias seções que compõem este livro. Igualmente, sou grato pelo envolvimento e apoio dos Professores Robert Gustafson, John Demel e Dr. John Merrill, coordenadores do Programa do Primeiro Ano de Engenharia da The Ohio State University. Agradecimentos especiais ao

* N. de T. O *software* MATLAB é em inglês. Com o objetivo de facilitar a compreensão do leitor e a aplicação do conteúdo deste livro, optamos por manter o nome das funções matemáticas na língua inglesa, assim como o padrão inglês de separação de casas decimais (isto é, a utilização de ponto em vez de vírgula).

Professor Mike Lichtensteiger (OSU) e à minha filha Tal Gilat (Stanford University), revisores meticulosos desta obra, pelas valiosas críticas e comentários.

Minha gratidão dirige-se também a todos que leram os originais e ajudaram-me na revisão nos diversos estágios de desenvolvimento do livro, incluindo Betty Barr, University of Houston; Andrei G. Chakhovskoi, University of California, Davis; Roger King, University of Toledo; Richard Kwor, University of Colorado em Colorado Springs; Larry Lagerstrom, University of California, Davis; Yueh-Jaw Lin, University of Akron; H. David Sheets, Canisius College; Geb Thomas, University of Iowa; Brian Vick, Virginia Polytechnic Institute and State University; Jay Weitzen, University of Massachusetts, Lowell; e Jane Patterson Fife, The Ohio State University. Por fim, agradeço o apoio de Joe Hayton, Ken Santor, Caroline Sieg, Katherine Hepburn, Simon Durkin, John Stout e Jay Beck, todos da John Wiley & Sons.

Espero que este livro seja de grande utilidade e que os usuários do MATLAB® possam apreciá-lo.

Amos Gilat
Columbus, Ohio

Sumário

Introdução	01
Capítulo 1 Primeiros Passos com MATLAB	19
1.1 INICIALIZANDO O MATLAB: JANELAS DO MATLAB	19
1.2 TRABALHANDO COM A JANELA COMMAND WINDOW	22
1.3 OPERAÇÕES ARITMÉTICAS COM ESCALARES	24
1.3.1 Ordem de precedência	24
1.3.2 Utilizando o MATLAB como uma calculadora	25
1.4 FORMATANDO DADOS NUMÉRICOS	26
1.5 FUNÇÕES ELEMENTARES NATIVAS DO MATLAB	27
1.6 DECLARANDO VARIÁVEIS ESCALARES	29
1.6.1 O operador de atribuição	29
1.6.2 Regras quanto ao uso dos nomes das variáveis	32
1.6.3 Variáveis predefinidas	32
1.7 COMANDOS ÚTEIS NO MANUSEIO DE VARIÁVEIS	33
1.8 EXEMPLOS DE APLICAÇÕES DE MATLAB	33
1.9 PROBLEMAS	36
Capítulo 2 Criando Arranjos	40
2.1 CRIANDO UM ARRANJO UNIDIMENSIONAL (VETOR)	40
2.2 CRIANDO ARRANJOS BIDIMENSIONAIS (MATRIZES)	44
2.2.1 Os comandos <code>zeros</code> , <code>ones</code> e <code>eye</code>	45
2.3 OBSERVAÇÕES QUANTO AO USO DE VARIÁVEIS NO MATLAB	46

2.4	O OPERADOR DE TRANSPOSIÇÃO	47
2.5	REFERÊNCIA A UM ELEMENTO DO ARRANJO	47
2.5.1	Vetor	47
2.5.2	Matriz	48
2.6	DOIS PONTOS (:) REFERENCIANDO ELEMENTOS DE ARRANJOS	49
2.7	ADICIONANDO ELEMENTOS ÀS VARIÁVEIS DECLARADAS	51
2.8	ELIMINANDO ELEMENTOS DE UMA MATRIZ	54
2.9	FUNÇÕES NATIVAS PARA MANIPULAÇÃO DE ARRANJOS	55
2.10	CADEIA DE CARACTERES (STRINGS) E VARIÁVEIS TIPO STRINGS	59
2.11	PROBLEMAS	61
Capítulo 3	Operações Matemáticas com Arranjos	65
3.1	ADIÇÃO E SUBTRAÇÃO DE ARRANJOS	66
3.2	MULTIPLICAÇÃO DE ARRANJOS	67
3.3	DIVISÃO DE ARRANJOS	70
3.4	OPERAÇÕES ESCALARES ENVOLVENDO ELEMENTOS DE MATRIZES	74
3.5	USANDO ARRANJOS EM FUNÇÕES NATIVAS DO MATLAB	77
3.6	FUNÇÕES NATIVAS PARA CÁLCULOS DE ARRANJOS	78
3.7	GERAÇÃO DE NÚMEROS ALEATÓRIOS	79
3.8	EXEMPLOS DE APLICAÇÕES DO MATLAB	81
3.9	PROBLEMAS	87
Capítulo 4	Rotinas (M-Files)	91
4.1	OBSERVAÇÕES QUANTO AO USO DE ROTINAS	91
4.2	CRIANDO E SALVANDO UMA ROTINA	92
4.3	RODANDO UMA ROTINA	93
4.3.1	O campo Current Directory	93
4.3.2	Procurando um diretório (ou pasta)	95
4.4	VARIÁVEIS GLOBAIS	95
4.5	ENTRADAS EM ROTINAS	95
4.6	COMANDOS DE SAÍDA	98
4.6.1	O comando <code>disp</code>	98
4.6.2	O comando <code>fprintf</code>	101
4.7	IMPORTANDO E EXPORTANDO DADOS	109
4.7.1	Comandos para importação e exportação de dados	109
4.7.2	Usando o assistente de importação	111
4.8	EXEMPLOS DE APLICAÇÕES DO MATLAB	113
4.9	PROBLEMAS	118

Capítulo 5 Gráficos Bidimensionais	122
5.1 O COMANDO <code>plot</code>	123
5.1.1 Gráfico de uma tabela de dados	127
5.1.2 Gráfico de uma função	128
5.2 O COMANDO <code>fplot</code>	129
5.3 PLOTANDO GRÁFICOS MÚLTIPLOS EM UMA MESMA SAÍDA	130
5.3.1 Usando o comando <code>plot</code>	130
5.3.2 Usando os comandos <code>hold on</code> e <code>hold off</code>	132
5.3.3 Usando o comando <code>line</code>	132
5.4 FORMATANDO UM GRÁFICO	133
5.4.1 Formatando um gráfico por meio de comandos	133
5.4.2 Formatando um gráfico por meio da janela Figure Window	137
5.5 GRÁFICOS COM ESCALAS LOGARÍTMICAS	137
5.6 GRÁFICOS PARA FINS ESPECÍFICOS	139
5.7 HISTOGRAMAS	141
5.8 GRÁFICOS POLARES	143
5.9 MÚLTIPLOS GRÁFICOS NA MESMA JANELA DE SAÍDA	144
5.10 EXEMPLOS DE APLICAÇÕES DO MATLAB	145
5.11 PROBLEMAS	150
Capítulo 6 Funções	156
6.1 DECLARANDO UMA FUNÇÃO NO MATLAB	157
6.2 ESTRUTURA DE UMA FUNÇÃO	157
6.2.1 Linha de declaração da função	158
6.2.2 Argumentos de entrada e saída	159
6.2.3 A linha de descrição da função (linha H1) e as linhas de comentários	160
6.2.4 O corpo da função	161
6.3 VARIÁVEIS LOCAIS VERSUS VARIÁVEIS GLOBAIS	161
6.4 SALVANDO UMA FUNÇÃO	162
6.5 CHAMANDO UMA FUNÇÃO	163
6.6 EXEMPLOS DE FUNÇÕES SIMPLES	164
6.7 COMPARAÇÃO ENTRE ROTINAS E FUNÇÕES	166
6.8 FUNÇÕES <code>inline</code>	166
6.9 O COMANDO <code>feval</code>	169
6.10 EXEMPLOS DE APLICAÇÕES DO MATLAB	170
6.11 PROBLEMAS	174
Capítulo 7 Programando no MATLAB	180
7.1 OPERADORES LÓGICOS E RELACIONAIS	181
7.2 SENTENÇAS CONDICIONAIS	189
7.2.1 A estrutura <code>if-end</code>	190

7.2.2	A estrutura <code>if-else-end</code>	192
7.2.3	A estrutura <code>if-elseif-else-end</code>	193
7.3	A SENTENÇA <code>switch - case</code>	194
7.4	LAÇOS (<i>LOOPS</i>)	197
7.4.1	Laços <code>for-end</code>	198
7.4.2	Laços <code>while-end</code>	202
7.5	LAÇOS ANINHADOS E NINHOS DE SENTENÇAS CONDICIONAIS	205
7.6	Os comandos <code>break</code> e <code>continue</code>	207
7.7	EXEMPLOS DE APLICAÇÕES DO MATLAB	207
7.8	PROBLEMAS	216
Capítulo 8 Polinômios, Ajuste de Curvas e Interpolação		222
8.1	POLINÔMIOS	223
8.1.1	Valor numérico de um polinômio	223
8.1.2	Raízes de um polinômio	224
8.1.3	Adição, multiplicação e divisão de polinômios	226
8.1.4	Derivadas de polinômios	227
8.2	AJUSTE DE CURVAS	228
8.2.1	Ajuste polinomial – a função <code>polyfit</code>	229
8.2.2	Ajuste de curvas com outras funções	232
8.3	INTERPOLAÇÃO	236
8.4	A INTERFACE BASIC FITTING	239
8.5	EXEMPLOS DE APLICAÇÕES DO MATLAB	242
8.6	PROBLEMAS	248
Capítulo 9 Gráficos Tridimensionais		254
9.1	CURVAS NO ESPAÇO 3-D	254
9.2	MALHAS E SUPERFÍCIES	256
9.3	SUPERFÍCIES ESPECIAIS	261
9.4	O comando <code>view</code>	263
9.5	EXEMPLOS DE APLICAÇÕES DO MATLAB	266
9.6	PROBLEMAS	271
Capítulo 10 Aplicações em Cálculo Numérico		274
10.1	RESOLVENDO UMA EQUAÇÃO A UMA VARIÁVEL	274
10.2	ENCONTRANDO O MÁXIMO OU O MÍNIMO DE UMA FUNÇÃO	277
10.3	INTEGRAÇÃO NUMÉRICA	278
10.4	EQUAÇÕES DIFERENCIAIS ORDINÁRIAS (EDOS)	281
10.5	EXEMPLOS DE APLICAÇÕES DO MATLAB	285
10.6	PROBLEMAS	291

Capítulo 11 Matemática Simbólica	296
11.1 OBJETOS SIMBÓLICOS E EXPRESSÕES SIMBÓLICAS	297
11.1.1 Criando objetos simbólicos	297
11.1.2 Criando expressões simbólicas	299
11.1.3 O comando <code>findsym</code> e a variável simbólica padrão	302
11.2 MODIFICANDO A FORMA DE UMA EXPRESSÃO SIMBÓLICA	303
11.2.1 Os comandos <code>collect</code> , <code>expand</code> e <code>factor</code>	303
11.2.2 Os comandos <code>simplify</code> e <code>simple</code>	305
11.2.3 O comando <code>pretty</code>	307
11.3 RESOLVENDO EQUAÇÕES ALGÉBRICAS	308
11.4 DIFERENCIAÇÃO	313
11.5 INTEGRAÇÃO	315
11.6 RESOLVENDO UMA EQUAÇÃO DIFERENCIAL ORDINÁRIA	316
11.7 PLOTANDO EXPRESSÕES SIMBÓLICAS	320
11.8 CÁLCULOS NUMÉRICOS COM EXPRESSÕES SIMBÓLICAS	322
11.9 EXEMPLOS DE APLICAÇÕES DO MATLAB	326
11.10 PROBLEMAS	334
Apêndice: Sumário de Caracteres, Comandos e Funções	341
Respostas de Problemas Seleccionados	349
Índice	357

Introdução

O MATLAB[®] é uma linguagem poderosa em termos de computação técnica. O nome MATLAB[®] vem da elisão das palavras MATrix LABoratory. Isto se deve à base operacional do *software* que são as matrizes. O MATLAB[®] é bastante versátil em cálculos matemáticos, modelagens e simulações, análises numéricas e processamentos, visualização e gráficos, desenvolvimentos de algoritmos, etc.

Atualmente, o MATLAB[®] é largamente utilizado nas universidades e faculdades nos cursos introdutórios ou avançados de matemática, ciências e, especialmente, nas engenharias. Na indústria, o *software* alcançou o status de ferramenta de pesquisa, projeto e desenvolvimento. O pacote-padrão do MATLAB possui ferramentas (funções) comuns a diversas áreas do conhecimento. Além disso, o MATLAB disponibiliza uma série de ferramentas adicionais (os *toolboxes*) que formam uma coleção de programas especiais projetados (e dedicados!) para resolver problemas específicos. Dentre os *toolboxes* mais utilizados, pode-se citar: processamento de sinais, cálculos simbólicos (literais), sistemas de controle e lógica fuzzy.

Até recentemente, a maioria dos usuários do MATLAB era formada de pessoas que possuíam bastante conhecimento em linguagem de programação, tais como FORTRAN e C, e migravam naturalmente para o MATLAB, à medida que o *software* tornava-se popular. Conseqüentemente, a maior parte da literatura a respeito do MATLAB assumia que o leitor já conhecia certas nuances sobre programação de computadores. Os livros sobre o MATLAB traziam tópicos (ou aplicações) avançados, dedicados a certos campos de pesquisa. Entretanto, nos últimos anos, o MATLAB foi-se desvinculando desses pré-requisitos e sendo adotado nos cursos introdutórios das universidades como o primeiro (às vezes, o único!) programa de computador ensinado. Para esses estudantes, fez-se necessário introduzir um livro no universo acadêmico que ensinasse o MATLAB sem pressupor conhecimentos sobre programação de computadores.

O PROPÓSITO DESTES LIVROS

MATLAB com Aplicações em Engenharia é indicado aos estudantes que estão iniciando o uso do MATLAB e têm pouca, ou nenhuma, experiência em programação de computadores. Pode ser utilizado como livro-texto num curso introdutório para calouros dos cursos de engenharia ou em *workshops* sobre o MATLAB. Pode servir também como requisito básico nos cursos avançados nas ciências e engenharias, quando o MATLAB é tão simplesmente uma ferramenta para solução de problemas. Este livro foi escrito para facilitar os estudos autônomos sobre o MATLAB. Além disso, serve como referência complementar, ou como um segundo livro, nos cursos em que o MATLAB é adotado, mas o professor não dispõe de tempo para explicar certos detalhes do *software*.

TÓPICOS ABORDADOS

O MATLAB é um programa imenso; assim, é impossível abranger todos os aspectos num só livro. Aqui são enfocados, primeiramente, os fundamentos do MATLAB. Uma vez entendidos esses fundamentos, acredito que os estudantes sentir-se-ão capazes de assimilar tópicos avançados, muitas vezes recorrendo apenas ao menu Help.

A ordenação dos capítulos foi escolhida cuidadosamente, baseada nos muitos anos de experiência no ensino do MATLAB num curso introdutório de engenharia. Os tópicos vão sendo apresentados de modo a permitir ao estudante seguir o livro capítulo por capí-

culo Numérico. Nesse capítulo são resolvidas equações não-lineares, são determinados o máximo ou mínimo de funções, são feitas integrações numéricas e são encontradas as soluções de equações diferenciais ordinárias de primeira ordem para um problema de valor inicial (PVI). O Capítulo 11 é um novo capítulo introduzido na segunda edição do livro. Ele cobre em detalhes o uso do MATLAB em operações simbólicas.

A ESTRUTURA TÍPICA DE UM CAPÍTULO

Em cada capítulo os tópicos são introduzidos gradualmente, de modo a facilitar a compreensão por parte do leitor. O uso do MATLAB é demonstrado extensivamente no texto e nos exemplos. Alguns dos exemplos maiores nos Capítulos 1-3 foram intitulados tutoriais. Cada utilização do MATLAB foi impressa sobre um fundo cinza. Explicações adicionais aparecem dentro de caixas de texto sobre fundo branco. A idéia é que o leitor execute as demonstrações e os tutoriais de modo a adquirir experiência com o *software*. Além disso, cada capítulo traz uma lista formal de problemas-exemplo que, em síntese, são aplicações do MATLAB na matemática, nas ciências e na engenharia. Cada exemplo começa com o enunciado do problema e termina com uma solução detalhada. Às vezes, foram colocados problemas-exemplo no meio dos capítulos. Todos os capítulos (exceto o Capítulo 2) têm uma seção de problemas-exemplo. Deve ser mencionado que o modo de solucionar um problema no MATLAB não é único. As soluções dadas nos exemplos foram escritas de modo a facilitar o acompanhamento da solução. Isso significa que, muitas vezes, o problema poderia ser resolvido escrevendo-se um programa pequeno, às vezes arduo. Os estudantes são encorajados a escrever suas próprias soluções e tentar comparar os resultados finais. No final de cada capítulo há um conjunto de problemas propostos. Estão incluídos problemas genéricos de matemática e ciências e problemas das diferentes disciplinas da engenharia.

MATEMÁTICA SIMBÓLICA

O MATLAB é um *software* para cálculos numéricos. Entretanto, as operações matemáticas simbólicas podem ser executadas se o Symbolic Math *toolbox* estiver instalado. O Symbolic Math *toolbox* está disponível na versão do *software* para o estudante e pode ser instalado no pacote padrão.

HARDWARE E SOFTWARE

O programa MATLAB, como a maioria dos *software*, sofre atualizações contínuas e novas versões são disponibilizadas frequentemente. A versão do MATLAB seguida no livro é a 7.0, Release 14. Deve ser enfatizado, porém, que este livro trata dos fundamentos do MATLAB, os quais permanecem inalterados de uma versão para a outra. O livro aborda o MATLAB for Windows, mas quase tudo permanece o mesmo em máquinas rodando o MATLAB for Mac, Linux, Unix, etc. Por fim, pressupõe-se que o usuário já tem o MATLAB instalado no computador e que sabe manusear a máquina.

A ORDEM DOS TÓPICOS NO LIVRO

É praticamente impossível escrever um livro-texto em que todos os assuntos sejam apresentados numa sequência que agrade a todos. A ordem dos tópicos neste livro é tal que os fundamentos do MATLAB são desenvolvidos primeiramente (arranjos e suas operações), e, conforme mencionado anteriormente, cada tópico vai sendo abordado numa seção apropriada, possibilitando a utilização de referências. Algumas pessoas podem preferir essa sequência e seguir outra diferente (especialmente quando o MATLAB é usado em sala de aula). Por exemplo, alguns professores podem tratar as quatro primeiras seções do Capítulo 4 (Rotinas) antes de abordar os Capítulos 2 (Criando Arranjos) e 3 (Operações Matemáticas com Arranjos). Isso permite aos estudantes utilizarem rotinas em vez da linha do *prompt* da janela Command Window, quando estudarem os Capítulos 2 e 3. Da mesma forma, operadores lógicos e relacionais (Seção 7.1) e polinômios (Seção 8.1) podem ser consideradas operações matemáticas básicas e serem apresentadas juntamente com o restante das operações do Capítulo 3.

Iniciação ao Ambiente do MATLAB

Este capítulo descreve as principais características e propósitos das diferentes janelas do MATLAB. Primeiramente, discutiremos em detalhes a janela Command Window e a exploraremos até o final do capítulo. O Capítulo 1 mostra também como usar o MATLAB para realizar operações aritméticas com números de um modo bastante parecido com as operações realizadas em uma calculadora simples (isso inclui algumas funções matemáticas elementares). O capítulo termina ensinando a declarar as variáveis escalares (por meio do operador de atribuição) e a utilizá-las em cálculos numéricos.

1.1 INICIALIZANDO O MATLAB: JANELAS DO MATLAB

Considerando-se que o *software* encontra-se instalado e inicializado no computador do usuário, as janelas abertas são semelhantes às aquelas mostradas na Figura 1-1. A figura contém três pequenas janelas: Command Window, Current Directory Window e Command History Window. Esse é o modo de abertura padrão (*default*) do MATLAB. Ao todo, o MATLAB pode apresentar oito janelas diferentes. A lista completa das janelas do MATLAB com os respectivos propósitos está resumida na Tabela 1-1. O botão **Start**, no canto inferior da tela, é uma novidade das versões mais recentes. Pode-se acessar todas as ferramentas e propriedades do MATLAB por esse botão.

Quatro dessas janelas – Command Window, Figure, Editor e Help – serão usadas extensivamente ao longo deste livro e terão uma breve descrição nas páginas seguintes. Descrições mais detalhadas estão incluídas nos capítulos em que se faz uso específico de cada uma delas.

Command Window: A principal janela do MATLAB é a Command Window. É ativada sempre que o MATLAB é inicializado. Muitas vezes é conveniente mantê-la aberta sem as demais. Isso pode ser feito fechando-se todas as outras (clique no **x** no canto direito

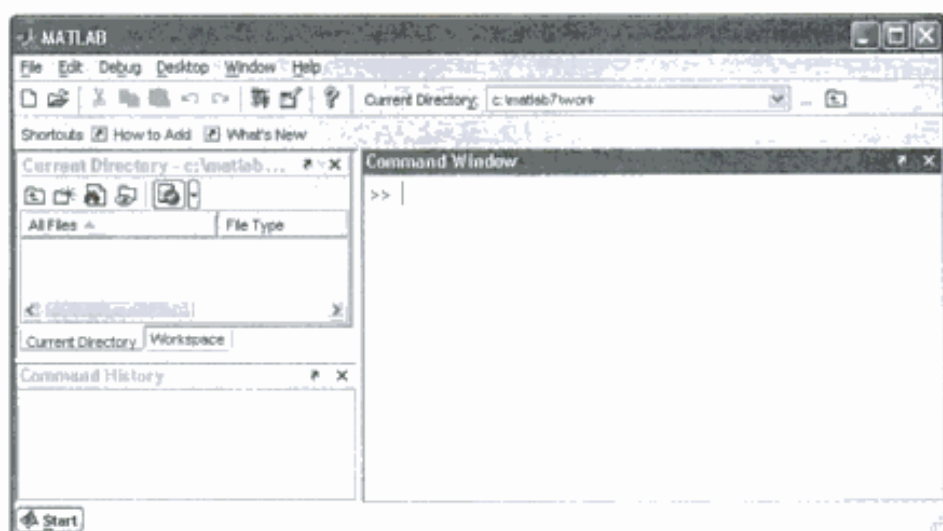


FIGURA 1-1 Desktop padrão do MATLAB.

TABELA 1-1 Janelas do MATLAB

Janela	Propósito
Command Window	Janela principal, inicializa variáveis e executa programas.
Figure	Apresenta o(s) resultado(s) dos comandos gráficos.
Editor	Permite a edição e a depuração de rotinas (<i>script files</i>) e funções.
Help	Ajuda na utilização do programa
Launch Pad	Fornece acesso às ferramentas, demos e documentação.
Command History	Apresenta o histórico dos comandos mais recentes digitados na janela Command Window.
Workspace	Disponibiliza informação sobre as variáveis que estão em uso.
Current Directory	Exibe os arquivos presentes no diretório ou na pasta atual.

superior da janela que você deseja fechar) ou, então, escolhendo a opção **Desktop Layout** no menu **Desktop** da barra de ferramentas e no submenu a opção **Command Window Only**. A Seção 1.2 descreve como trabalhar com a janela Command Window.

Janela Figure: A janela Figure é aberta toda vez que um comando gráfico é executado. Exibe o(s) gráfico(s) criado(s) por esse(s) comando(s). A Figura 1-2 mostra um exemplo gráfico na janela Figure. Uma descrição mais detalhada é apresentada no Capítulo 5.

Janela Editor: A janela Editor é usada para escrever e editar programas. É possível abri-la a partir do menu **File** na janela Command Window. Um exemplo de função na janela Editor é mostrada na Figura 1-3. Mais detalhes sobre essa janela serão apresentados no Capítulo 4 (criando rotinas e *scripts*) e no Capítulo 6 (criando funções).

Janela Help: Contém um descritivo detalhado do Help. É possível abri-la a partir do menu **Help** na barra de ferramentas de qualquer janela do MATLAB. Ela é interativa e pode ser utilizada para obtenção de informação sobre qualquer característica do MATLAB. A Figura 1-4 mostra uma janela típica do Help.

A primeira inicialização do MATLAB exibirá telas semelhantes à da Figura 1-1. Para a maioria dos iniciantes, pode ser conveniente fechar todas as janelas, exceto a Command Window. Não se preocupe, as janelas fechadas podem ser reabertas selecionando-as novamente no menu **Desktop**. Por fim, as janelas mostradas na Figura 1-1 po-

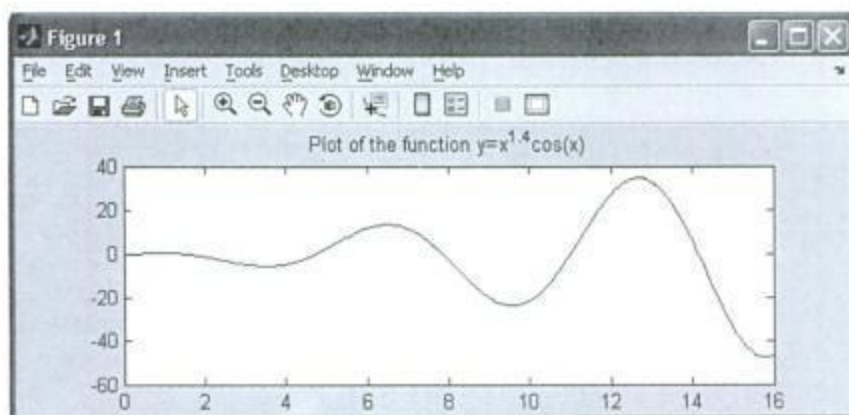


FIGURA 1-2 Exemplo de uma função na janela Figure.



FIGURA 1-3 Exemplo de aplicação na janela Editor.



FIGURA 1-4 O Help do MATLAB.

dem ser exibidas selecionando-as primeiramente no **Desktop Layout** no menu **Desktop** e, a seguir, na opção **Default** do submenu.

1.2 TRABALHANDO COM A JANELA COMMAND WINDOW

Conforme mencionado, a principal janela do MATLAB é a Command Window. Ela pode ser usada para executar comandos, abrir outras janelas, rodar programas escritos pelo usuário e/ou gerenciar o uso do MATLAB. Um exemplo simples de uso da janela Command Window é mostrado na Figura 1-5.

Observações quanto ao uso da janela Command Window:

- Para digitar um comando, o cursor deve ser posicionado junto ao *prompt* de comando (`>>`).
- Uma vez que o comando foi digitado e a tecla **Enter** foi pressionada, o comando é executado. Contudo, somente o último comando é executado. Qualquer comando digitado anteriormente torna-se inacessível (a menos que seja reescrito ou chamado novamente).
- Muitos comandos podem ser digitados na mesma linha. Isso pode ser feito separando-os por vírgulas. Quando a tecla **Enter** é pressionada, os comandos são executados na ordem em que foram digitados, sucessivamente da esquerda para a direita.
- Não é possível retornar à última linha exibida na janela Command Window para fazer uma correção (edição) e, então, executar o comando novamente, produzindo um resultado nessa mesma linha.

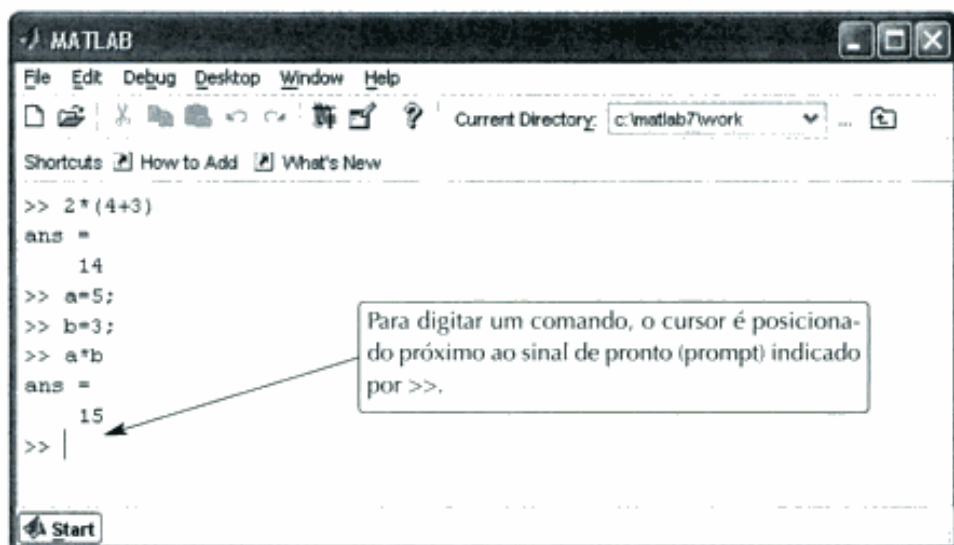


FIGURA 1-5 A janela Command Window.

- Um comando anteriormente digitado pode ser chamado outra vez através das teclas de navegação (↑) e (↓). Assim que o comando desejado for exibido no *prompt* de comando, é possível modificá-lo (se necessário) e executá-lo.
- Se um comando for extenso demais para caber em uma linha, basta digitar reticências (...) e pressionar a tecla **Enter** para continuar na próxima linha. A continuação de um comando no MATLAB pode ocupar 4.096 caracteres após a linha inicial.

O ponto-e-vírgula (;):

Um comando é executado quando é digitado na janela Command Window e a tecla **Enter** é pressionada. Qualquer resultado que o comando produza é mostrado no Command Window. O resultado de saída será ocultado se um ponto-e-vírgula (;) for digitado no final do comando. Digitar um ponto-e-vírgula é interessante quando o resultado da operação é óbvio ou conhecido, ou então quando a saída é grande demais.

Se múltiplos comandos forem digitados na mesma linha, havendo ponto-e-vírgula entre eles (em vez de uma vírgula), nenhum resultado será exibido.

Digitando %:

Quando o símbolo de percentagem (%) é digitado no início de uma linha de comando, toda a linha é designada como um comentário. Significa que, quando a tecla **Enter** for pressionada, a linha não será executada. O caractere %, seguido de um comentário, também pode ser digitado após um comando (na mesma linha). Isso não produzirá efeitos na execução desse comando.

A janela de Command Window não possui comentário padrão. Porém, comentários são freqüentemente usados em programas para adicionar descrições ou explicar algum ponto particular sobre programa (veja Capítulos 4 e 6).

O comando `clc`:

Nas expressões que possuem muitas operações, as operações com maior ordem de precedência serão executadas em primeiro lugar. Se duas ou mais operações tiverem a mesma ordem de precedência, a expressão mais à esquerda será executada primeiro. A próxima seção ilustra como os parênteses podem ser utilizados para mudar a ordem de precedência nos cálculos.

1.3.2 Utilizando o MATLAB como uma calculadora

O modo mais simples de utilizar o MATLAB é como uma calculadora. Isso pode ser feito digitando-se uma expressão matemática no *prompt* da janela Command Window e pressionando a tecla **Enter**. O MATLAB calcula a expressão, emite uma resposta do tipo `ans =` e exibe o resultado numérico da expressão nas linhas seguintes. Esses procedimentos são demonstrados passo a passo no Tutorial 1-1:

TUTORIAL 1-1 Utilizando o MATLAB como uma calculadora

The screenshot shows the MATLAB Command Window with the following commands and outputs:

```
>> 7 + 8/2
ans =
    11
>> (7 + 8)/2
ans =
    7.5000
>> 4 + 5/3 + 2
ans =
    7.6667
>> 5^3/2
ans =
    62.5000
>> 27^(1/3) + 32^0.2
ans =
    5
>> 27^1/3 + 32^0.2
ans =
    11
>> 0.7854 - (0.7854)^3/(1*2*3) + 0.785^5/(1*2*3*4*5)...
- (0.785)^7/(1*2*3*4*5*6*7)
ans =
    0.7071
>>
```

Annotations explaining the order of operations:

- For `7 + 8/2`: Digite e pressione **Enter**. $8/2$ é executado primeiro.
- For `(7 + 8)/2`: Digite e pressione **Enter**. $7 + 8$ é executado primeiro.
- For `5^3/2`: 5^3 é executado primeiro.
- For `5^3/2`: 5^3 é executado primeiro, $/2$ é executado em seguida.
- For `27^(1/3) + 32^0.2`: $1/3$ é executado primeiro, $27^{1/3}$ e $32^{0.2}$ são executados em seguida e $+$ é executado por último.
- For `27^1/3 + 32^0.2`: 27^1 e $32^{0.2}$ são executados inicialmente, $/3$ é executado em seguida e $+$ é executado por último.
- For the long expression: Digite reticências... (e pressione **Enter**) para continuar a expressão na linha seguinte.
- For the long expression: A última expressão contém os quatro primeiros termos da série de Taylor para $\sin(\pi/4)$.

1.4 FORMATANDO DADOS NUMÉRICOS

O usuário pode controlar o formato segundo o qual o MATLAB exibe os dados de saída na tela. No Tutorial 1-1, o formato de saída é o ponto fixo com quatro dígitos decimais (chamado `short`), que é o formato padrão para valores numéricos. O formato de saída pode ser modificado através do comando `format`. Uma vez digitado um comando `format`, todos os resultados exibidos na saída seguem esse formato específico. Muitos formatos disponíveis no MATLAB estão listados e descritos na Tabela 1-2.

O MATLAB possui diversos outros formatos para dados numéricos. Mais informações sobre esses formatos podem ser obtidas digitando-se `help format` na linha do *prompt* do Command Window. O formato escolhido para os dados numéricos não afeta a maneira como o MATLAB calcula e salva os números.

TABELA 1-2 Formatos padronizados dos números

Comando	Descrição	Exemplo
<code>format short</code>	Notação ponto fixo com 4 dígitos decimais para: $0.001 \leq \text{número} \leq 1000$ Caso contrário, formato do número <code>short</code> e.	<pre>>> 290/7 ans = 41.4286</pre>
<code>format long</code>	Notação ponto fixo com 14 dígitos decimais para: $0.001 \leq \text{número} \leq 1000$ Caso contrário, formato do número <code>long</code> e.	<pre>>> 290/7 ans = 41.42857142857143</pre>
<code>format short e</code>	Notação científica com 4 dígitos decimais.	<pre>>> 290/7 ans = 4.1429e+001</pre>
<code>format long e</code>	Notação científica com 15 dígitos decimais.	<pre>>> 290/7 ans = 4.142857142857143e+001</pre>
<code>format short g</code>	Melhor em 5 dígitos entre a notação de ponto fixo ou ponto flutuante.	<pre>> 290/7 ans = 41.429</pre>
<code>format long g</code>	Melhor em 15 dígitos entre a notação de ponto fixo ou ponto flutuante.	<pre>>> 290/7 ans = 41.4285714285714</pre>

TABELA 1-2 Formatos padronizados dos números

(continuação)

Comando	Descrição	Exemplo
<code>format bank</code>	Dois dígitos decimais.	<pre>>> 290/7 ans = 41.43</pre>
<code>format compact</code>	Elimina espaços para permitir que mais linhas de informação sejam mostradas na tela.	
<code>format loose</code>	Adiciona espaços entre linhas (oposto ao formato <code>compact</code>).	

1.5 FUNÇÕES ELEMENTARES NATIVAS DO MATLAB

Expressões no MATLAB podem incluir funções, além das operações aritméticas básicas. O MATLAB possui uma biblioteca extensa de funções nativas. Uma função é caracterizada por um nome e um argumento entre parênteses. Por exemplo, a função que calcula a raiz quadrada de um número é `sqrt(x)`. O nome da função é `sqrt` e o argumento é `x`. O argumento de uma função pode ser um número, uma variável (veja Seção 1.6) ou uma expressão composta de números e/ou variáveis. Funções também podem ser incluídas no argumento de outras funções, assim como em expressões. O Tutorial 1-2 ensina como usar a função `sqrt(x)` operando com escalares.

Listas de funções elementares nativas do MATLAB são mostradas nas Tabelas 1-3, 1-4 e 1-5. Uma lista completa de funções, organizadas pelo nome da categoria a que pertencem, pode ser encontrada em Help Window.

TUTORIAL 1-2 Usando a função nativa `sqrt`

```
>> sqrt(64)
```

```
ans =
    8
```

O argumento é um número.

```
>> sqrt(50 + 14*3)
```

```
ans =
    9.5917
```

O argumento é uma expressão.

```
>> sqrt(54 + 9*sqrt(100))
```

```
ans =
    12
```

O argumento inclui uma função.

```
>> (15 + 600/4)/sqrt(121)
```

```
ans =
    15
```

Uma função é incluída em uma expressão.

```
>>
```

TABELA 1-3 Funções matemáticas elementares

Função	Descrição	Exemplo
<code>sqrt(x)</code>	Raiz quadrada	<pre>>> sqrt(81) ans = 9</pre>
<code>exp(x)</code>	Exponencial (e^x)	<pre>>> exp(5) ans = 148.4132</pre>
<code>abs(x)</code>	Valor absoluto (módulo)	<pre>>> abs(-24) ans = 24</pre>
<code>log(x)</code>	Logaritmo neperiano ($\ln$)	<pre>>> log(1000) ans = 6.9078</pre>
<code>log10(x)</code>	Logaritmo base 10	<pre>>> log10(1000) ans = 3</pre>
<code>factorial(x)</code>	Fatorial de x ($x!$) (x deve ser um inteiro positivo)	<pre>>> factorial(5) ans = 120</pre>

TABELA 1-4 Funções trigonométricas

Função	Descrição	Exemplo
<code>sin(x)</code>	Seno do argumento x (x em radianos)	<pre>>> sin(pi/6) ans = 0.5000</pre>
<code>cos(x)</code>	Co-seno do argumento x (x em radianos)	<pre>>> cos(pi/6) ans = 0.8660</pre>
<code>tan(x)</code>	Tangente do argumento x (x em radianos)	<pre>>> tan(pi/6) ans = 0.5774</pre>
<code>cot(x)</code>	Cotangente do argumento x (x em radianos)	<pre>>> cot(pi/6) ans = 1.7321</pre>

As funções trigonométricas inversas são: `asin(x)`, `acos(x)`, `atan(x)` e `acot(x)`. Já as funções trigonométricas hiperbólicas são: `sinh(x)`, `cosh(x)`, `tanh(x)` e `coth(x)`. A tabela acima usa `pi` representando π^* (veja a Seção 1.6.3).

* N. de T.: Essa representação é apenas simbólica. Numericamente, π é um número irracional. Nenhum programa de computador é capaz de representá-lo em sua totalidade.

TABELA 1-5 Funções de arredondamento

Função	Descrição	Exemplo
<code>round(x)</code>	Arredonda para o inteiro mais próximo.	<pre>>> round(17/5) ans = 3</pre>
<code>fix(x)</code>	Arredonda para o inteiro positivo imediatamente menor.	<pre>>> fix(13/5) ans = 2</pre>
<code>ceil(x)</code>	Arredonda para o inteiro positivo imediatamente maior.	<pre>>> ceil(11/5) ans = 3</pre>
<code>floor(x)</code>	Arredonda para o inteiro negativo imediatamente menor.	<pre>>> floor(-9/4) ans = -3</pre>
<code>rem(x,y)</code>	Retorna o resto da divisão x por y .	<pre>>> rem(13,5) ans = 3</pre>
<code>sign(x)</code>	Função sinal. Retorna: 1 (se $x > 0$); -1 (se $x < 0$) e 0 (se $x = 0$).	<pre>>> sign(5) ans = 1</pre>

1.6 DECLARANDO VARIÁVEIS ESCALARES

Uma variável escalar no MATLAB é um nome formado por uma letra ou uma cadeia de letras (e números) ao qual é atribuído um valor. Uma vez atribuído um valor numérico à variável, podemos usá-la em expressões matemáticas, em funções, em sentenças e comandos do MATLAB. Tecnicamente, uma variável é um espaço de memória reservado para armazenar um certo tipo de dado, tendo um nome para referenciar o seu conteúdo. Quando uma variável é declarada, o MATLAB aloca um espaço de memória onde o conteúdo da variável é armazenado. Ao se utilizar a variável, automaticamente o conteúdo (dado armazenado) é passado ao comando ou à sentença que fará uso dela. Se for atribuído um novo valor à variável, o conteúdo da posição de memória será substituído. No Capítulo 1, atribuiremos somente valores numéricos (escalares) às variáveis. A atribuição e o endereçamento de arranjos – matrizes e vetores – serão discutidos no Capítulo 2.

1.6.1 O operador de atribuição

No MATLAB, o sinal de igualdade ($=$) é denominado operador de atribuição. O operador de atribuição inicializa ou modifica o valor de uma variável.

Nome_variável = Um valor ou uma expressão numérica

- O lado esquerdo do operador de atribuição pode conter apenas uma variável. O lado direito pode ser um número ou uma expressão numérica, possuindo números

e/ou variáveis previamente inicializadas. Teclando-se **Enter**, o resultado ou o número à direita do operador é atribuído à variável no lado esquerdo e o MATLAB exibe a variável com o seu respectivo valor nas duas próximas linhas.

Os exemplos a seguir mostram como o operador de atribuição funciona:

```
>> x = 15
x =
    15

>> x = 3*x - 12
x =
    33
>>
```

O número 15 é atribuído à variável x.

O MATLAB exibe a variável e o valor atribuído.

Um novo valor é atribuído à variável x. O novo valor é 3 vezes o valor anterior de x menos 12.

A última sentença ($x = 3x - 12$) ilustra a diferença entre o operador de atribuição e o sinal de igualdade. Se, nessa sentença, o sinal (=) significasse igualdade, o valor de x deveria ser 6 (resolvendo-se a equação para x).

A utilização de variáveis previamente declaradas para definir uma nova variável é demonstrada a seguir:

```
>> a = 12
a =
    12

>> B = 4
B =
     4

>> C = (a - B) + 40 - a/B*10
C =
    18
>>
```

Atribui 12 à variável a.

Atribui 4 à variável B.

Atribui o valor da expressão no lado direito do sinal (=) à variável C.

- Se um ponto-e-vírgula é digitado no fim de um comando, quando a tecla **Enter** for pressionada o MATLAB não exibirá o valor atribuído à variável (a variável é inicializada e encontra-se armazenada na memória).
- Qualquer variável declarada pode ter seu valor exibido a qualquer momento simplesmente digitando-se o nome da variável e pressionando-se **Enter**.

Por exemplo, se a última demonstração é repetida usando-se ponto-e-vírgula:

```
>> a = 12;
>> B = 4;
>> C = (a - B) + 40 - a/B*10;
>> C
C =
    18
>>
```

As variáveis *a*, *B* e *C* foram declaradas e inicializadas, mas não tiveram o conteúdo exibido em virtude do ponto-e-vírgula em cada linha.

O valor da variável *C* é exibido se o nome da variável é digitado e a tecla **ENTER** é pressionada.

- Muitas atribuições podem ser feitas numa mesma linha. Cada atribuição deve ser separada com uma vírgula ou ponto-e-vírgula, conforme conveniência (espaços também podem ser adicionados após uma vírgula ou ponto-e-vírgula). Pressionando a tecla **Enter**, as atribuições são efetuadas da esquerda para a direita e as variáveis com os respectivos valores são exibidas na tela (exceto aquelas terminadas em ponto-e-vírgula). Por exemplo, as atribuições das variáveis *a*, *B* e *C* acima podem ser feitas na mesma linha.

```
>> a = 12, B = 4; C = (a - B) + 40 - a/B*10
a =
    12
C =
    18
```

A variável *B* e o conteúdo dela não são exibidos porque um ponto-e-vírgula foi digitado após a inicialização de *B*.

- Uma variável previamente declarada pode ter seu conteúdo modificado. Por exemplo:

```
>> ABB = 72;
>> ABB = 9;
>> ABB
ABB =
     9
>>
```

O número 72 é atribuído à variável *ABB*.

Um novo número (9) é atribuído à variável *ABB*.

O valor real da variável *ABB* é exibido se o nome da variável é digitado na linha do *prompt* e a tecla **Enter** é pressionada.

- Uma vez declarada a variável, ela pode ser utilizada no argumento de funções. Por exemplo:

```
>> x = 0.75;
>> E = sin(x)^2 + cos(x)^2
E =
     1
>>
```

1.6.2 Regras quanto ao uso de nomes de variáveis

Os nomes das variáveis:

- Podem conter até 63 caracteres no MATLAB 7.0 (31 caracteres no MATLAB 6.0).
- Podem conter letras, números e o caractere sublinhar (*underscore*).
- Devem iniciar com uma letra.
- O MATLAB é *case sensitive*, ou seja, faz distinção entre nomes de variáveis com letras maiúsculas e minúsculas. Por exemplo: AA, Aa, aA e aa podem ser nomes de quatro variáveis diferentes.
- Evite usar nomes de funções nativas do MATLAB para nomear variáveis (por exemplo: cos, sin, exp, sqrt, etc). Uma vez que o nome de uma função é utilizado para definir uma variável, essa função não pode mais ser usada.

1.6.3 Variáveis predefinidas

Ao inicializar o uso do MATLAB, uma certa quantidade de variáveis é carregada automaticamente na memória do programa. Por exemplo:

<code>ans</code>	Variável que assume o valor da última expressão não atribuída a uma variável especificada (veja Tutorial 1-1). Se o usuário não atribui o valor de uma expressão a uma variável, o MATLAB armazena, automaticamente, o resultado em <code>ans</code> .
<code>pi</code>	O número π (aproximação).
<code>eps</code>	A menor diferença entre dois números. Equivale a $2^{(-52)}$ ou, aproximadamente, $2,2204e-16$.
<code>inf</code>	Infinito.
<code>i</code>	Definido como $\sqrt{-1}$, isto é: $0 + 1.0000i$.
<code>j</code>	O mesmo que <code>i</code> .
<code>NaN</code>	Abreviação de Not-a-Number. Usado quando o MATLAB não pode determinar um valor numérico válido. Por exemplo, $0/0$.

As variáveis predefinidas podem ser redefinidas a qualquer momento. É recomendado que as variáveis `pi`, `eps` e `inf` não sejam redefinidas, porque muitas aplicações fazem uso delas. Outras variáveis, como `i` e `j`, às vezes são redefinidas quando os números complexos não estão envolvidos numa aplicação particular (como acontece na linguagem de programação C/C++, dentre outras, `i` e `j` freqüentemente são associados aos índices de contagem em laços – *loops* – de programas).

1.7 COMANDOS ÚTEIS NO MANUSEIO DE VARIÁVEIS

A seguir são mostrados alguns comandos usados para eliminar variáveis ou obter informação a respeito das variáveis declaradas dentro de um sessão do MATLAB. Quando esses comandos são digitados na janela **Command Window** e a tecla **Enter** é pressionada, eles fornecem a informação desejada ou realizam uma tarefa, conforme discriminado a seguir:

Comando	Resultado
<code>clear</code>	Apaga da memória todas as variáveis.
<code>clear x y z</code>	Apaga da memória somente as variáveis <code>x</code> , <code>y</code> e <code>z</code> .
<code>who</code>	Exibe uma lista de variáveis declaradas/ativas na memória.
<code>whos</code>	Exibe uma lista de variáveis declaradas na memória, com o respectivo tamanho em <i>bytes</i> e a classe de armazenamento.

1.8 EXEMPLOS DE APLICAÇÕES DE MATLAB

Problema-exemplo 1-1: Identidade trigonométrica

Uma identidade trigonométrica é dada por:

$$\cos^2 \frac{x}{2} = \frac{\tan x + \sin x}{2 \tan x}$$

Substituindo $x = \frac{\pi}{5}$, verifique a identidade calculando cada lado da equação.

Solução

```
>> x = pi/5;
```

Declara e inicializa `x`.

```
>> LE = cos(x/2)^2
```

Calcula o lado esquerdo (LE) da identidade.

```
LE =
```

```
0.9045
```

```
>> LD = (tan(x) + sin(x))/(2*tan(x))
```

Calcula o lado direito (LD) da identidade.

```
LD =
```

```
0.9045
```

```
>>
```

Problema-exemplo 1-2: Geometria e trigonometria

Quatro círculos estão dispostos como mostra a figura. Os círculos tangenciam-se dois a dois num determinado ponto. A partir daí, determine a distância entre os centros C_2 e C_4 .

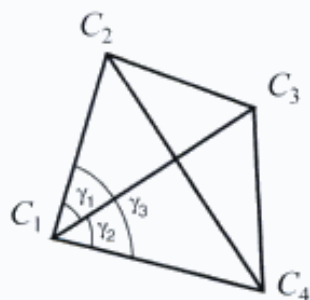
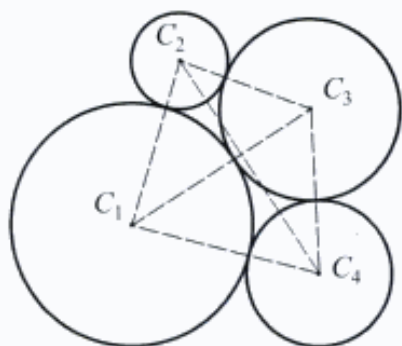
Os raios dos círculos são: $R_1 = 16\text{mm}$, $R_2 = 6.5\text{mm}$, $R_3 = 12\text{mm}$ e $R_4 = 9.5\text{mm}$.

Solução

As retas que ligam os centros dos círculos geram quatro triângulos. Em dois desses triângulos, $\Delta C_1 C_2 C_3$ e $\Delta C_1 C_3 C_4$, os comprimentos de todos os lados são conhecidos. Essa informação é usada para calcular os ângulos γ_1 e γ_2 , por meio da lei dos co-senos. Por exemplo, γ_1 é calculado a partir de:

$$(C_2 C_3)^2 = (C_1 C_2)^2 + (C_1 C_3)^2 - 2(C_1 C_2)(C_1 C_3) \cos \gamma_1$$

Em seguida, o comprimento do lado $C_2 C_4$ é calculado tomando-se como base o triângulo $\Delta C_1 C_2 C_4$. Para tanto, basta usar novamente a lei dos co-senos (os comprimentos dos lados $C_1 C_2$ e $C_1 C_4$ são conhecidos e o ângulo γ_3 é a soma dos ângulos γ_1 e γ_2).



```
>> R1 = 16; R2 = 6.5; R3 = 12; R4 = 9.5;
```

```
>> C1C2 = R1 + R2; C1C3 = R1 + R3;
```

```
>> C1C4 = R1 + R4; C2C3 = R2 + R3;
```

```
>> C3C4 = R3 + R4;
```

```
>> gama1 = acos((C1C2^2 + C1C3^2 - C2C3^2)/(2*C1C2*C1C3));
```

```
>> gama2 = acos((C1C3^2 + C1C4^2 - C3C4^2)/(2*C1C3*C1C4));
```

```
>> gama3 = gama1 + gama2;
```

```
>> C2C4 = sqrt(C1C2^2 + C1C4^2 - 2*C1C2*C1C4*cos(gama3))
```

```
C2C4 =
```

```
33.5051
```

Declara e inicializa os Rs.

Determina os comprimentos dos lados.

Determina γ_1 , γ_2 e γ_3 .

Determina o comprimento do lado $C_2 C_4$.

Problema-exemplo 1-3: Transferência de calor

Um objeto que está a uma temperatura inicial T_0 é colocado em $t = 0$ dentro de uma câmara à temperatura constante (T_s). A mudança na temperatura do objeto segue a lei:

$$T = T_s + (T_0 - T_s) e^{-kt}$$

onde T é a temperatura do objeto em um tempo t qualquer e k é uma constante. De posse dessas informações, considere uma lata de refrigerante exposta a uma temperatura 48.9°C. Em seguida, é colocada dentro de um refrigerador onde a temperatura é 3.3°C. Determine a temperatura da lata em graus Celsius, em valores inteiros, três horas após a lata ser colocada no refrigerador. Considere $k = 0.45$. Declare inicialmente todas as variáveis e então calcule a temperatura usando apenas um comando do MATLAB.

Solução

```
>> Ts = 3.3; T0=48.9; k=0.45; t=3;
>> T= round (Ts + (T0-Ts)*exp(-k*t))
T =
    15
```

Arredonda para um valor inteiro de temperatura.

Problema-exemplo 1-4: Juros compostos

O saldo B de uma conta de poupança após t anos é dado por:

$$B = P \left(1 + \frac{r}{n} \right)^{nt} \quad (1)$$

Investidas inicialmente P unidades monetárias, sendo a taxa de juros anual r e os juros compostos sendo calculados n vezes ao ano. Sendo os juros compostos calculados anualmente, o saldo é dado por:

$$B = P(1 + r)^t \quad (2)$$

Em uma conta de poupança são depositados \$ 5,000.00, investidos durante 17 anos em um banco, com juros compostos anualmente. Em outra conta são depositados \$ 5,000.00, com juros compostos mensalmente. Em ambas as contas, a taxa de juros anual é 8.5%. Use o MATLAB para determinar em quanto tempo (em anos e meses) o saldo da segunda conta será idêntico ao saldo da primeira conta, decorridos os 17 anos.

Solução

Passos:

- (a) Determine B , usando a Equação (2), para os \$5,000.00 investidos na conta onde os juros são compostos anualmente, durante 17 anos.

- (b) Determine t para o saldo B , calculado na letra (a), a partir dos juros compostos mensalmente, Equação (1).
- (c) Determine o número de anos e meses que correspondem ao intervalo de tempo t .

```
>> P = 5000; r = 0.085; ta = 17; n = 12;
```

```
>> B = P*(1+r)^ta
```

```
B =
```

```
2.0011e+004
```

```
>> t = log(B/P)/(n*log(1+r/n))
```

```
t =
```

```
16.3737
```

```
>> anos = fix(t)
```

```
anos =
```

```
16
```

```
>> meses = ceil((t - anos)*12)
```

```
meses =
```

```
5
```

Passo (a): determina B a partir da equação (2).

Passo (b): resolve a equação (1) para t e determina o valor de t .

Passo (c): determina o número de anos.

Determina o número de meses.

1.9 PROBLEMAS

Resolva os problemas a seguir utilizando a janela Command Window.

1. Calcule:

$$a) \frac{35 \cdot 7 \cdot 64 - 7^3}{45 + 5^2}$$

$$b) \frac{5}{4} \cdot 7 \cdot 6^2 + \frac{3^7}{(9^3 - 652)}$$

2. Calcule:

$$a) (2 + 7)^3 + \frac{273^{2/3}}{2} + \frac{55^2}{3}$$

$$b) 2^3 + 7^3 + \frac{273^3}{2} + 55^{3/2}$$

3. Calcule:

$$a) \frac{3^7 \log(76)}{7^3 + 546} + \sqrt[3]{910}$$

$$b) 43 \cdot \frac{(\sqrt[4]{250} + 23)^2}{e^{(45 \cdot y)^4}}$$

4. Calcule:

$$a) \cos^2\left(\frac{5\pi}{6}\right) \sin\left(\frac{7\pi}{8}\right)^2 + \frac{\tan\left(\frac{\pi \ln 8}{6}\right)}{\sqrt{7}}$$

$$b) \cos\left(\frac{5\pi}{6}\right)^2 \sin^2\left(\frac{7\pi}{8}\right) + \frac{\tan\left(\frac{\pi \ln 8}{6}\right)}{7 \cdot \frac{5}{2}}$$

5. Declare e inicialize a variável x como $x = 13.5$. Em seguida, determine:

$$a) x^3 + 5x^2 - 26.7x - 52$$

$$b) \frac{\sqrt{14x^3}}{e^{3x}}$$

$$c) \log|x^2 - x^3|$$

6. Declare e inicialize as variáveis x e z como $x = 9.6$ e $z = 8.1$. Em seguida, determine:

$$a) xz^2 - \left(\frac{2z}{3x}\right)^{\frac{2}{5}}$$

$$b) \frac{443z}{2x^3} + \frac{e^{-x}}{(x+z)}$$

7. Declare e inicialize as variáveis a , b , c e d como:

$$a = 15.62, b = -7.08, c = 62.5 \text{ e } d = 0.5(ab - c).$$

Em seguida, calcule:

$$a) a + \frac{ab(a+d)^2}{c \sqrt{|ab|}}$$

$$b) de^{\left(\frac{d}{z}\right)} + \frac{\frac{ad+cd}{\frac{20}{a} + \frac{30}{b}}}{(a+b+c+d)}$$

8. Calcule (usando apenas um comando) o raio r de uma esfera de volume $5.74e-3\text{m}^3$. Uma vez conhecido o valor de r , use-o para determinar a área da superfície da esfera.

9. Duas identidades trigonométricas são definidas como:

$$a) \sin 2x = 2 \sin x \cos x$$

$$b) \cos \frac{x}{2} = \sqrt{\frac{1 + \cos x}{2}}$$

Considerando $x = \frac{5}{24}\pi$, verifique cada uma das identidades calculando os dois lados da equação.

10. Duas identidades trigonométricas são definidas como:

$$a) \tan 2x = \frac{2 \tan x}{1 - \tan^2 x}$$

$$b) \tan \frac{x}{2} = \sqrt{\frac{1 - \cos x}{1 + \cos x}}$$

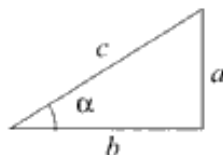
Considerando $x = \frac{3}{17}\pi$, verifique cada uma das identidades calculando os dois lados da equação.

11. Declare duas variáveis: $\alpha = 5\pi/9$ e $\beta = \pi/7$. Usando essas variáveis, mostre (calculando os dois lados da igualdade) que a identidade trigonométrica a seguir está correta.

$$\cos \alpha - \cos \beta = 2 \sin \frac{1}{2}(\alpha + \beta) \sin \frac{1}{2}(\beta - \alpha)$$

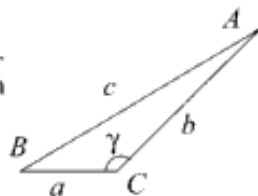
12. No triângulo retângulo mostrado na figura ao lado, $a = 11\text{cm}$ e $c = 21\text{cm}$. Declare as variáveis a e c e então:

- a) Usando o Teorema de Pitágoras, calcule b digitando apenas uma linha de comando.
b) Usando o valor de b encontrado em (a) e a função $\arccos(x)$, determine o ângulo α em graus, utilizando apenas uma linha de comando.



13. No triângulo mostrado, $a = 18\text{cm}$, $b = 35\text{cm}$ e $c = 50\text{cm}$. Declare as variáveis a , b e c para calcular o ângulo γ (em graus) substituindo as variáveis na lei dos co-senos.

$$(\text{Lei dos Co-senos: } c^2 = a^2 + b^2 - 2ab \cos \gamma)$$



14. A distância d de um ponto (x_0, y_0) à reta $Ax + By + C = 0$ é dada por:

$$d = \frac{|Ax_0 + By_0 + C|}{\sqrt{A^2 + B^2}}$$

Determine a distância do ponto $(2, -3)$ à reta $3x + 5y - 6 = 0$. Primeiramente, declare as variáveis A , B , C , x_0 e y_0 , então calcule d . (Use as funções `abs` e `sqrt`.)

15. Flores são embaladas em caixas com capacidade para acomodar uma dúzia delas. Usando a função `ceil`, determine quantas caixas serão necessárias para embalar 751 flores.

16. Declare as seguintes variáveis:

```
preco_mesa = $ 256.95
preco_cadeira = $ 89.99
```

Mude o formato da janela Command Window para *bank* e:

- Determine o preço total de duas mesas e oito cadeiras.
- Repita o item (a), mas adicione 5.5% de juros.
- Repita o item (b), mas arredonde o preço total em reais.

17. Um modo de adicionar frações é utilizando um denominador comum (no caso, o menor). Por exemplo, o menor denominador comum entre $1/4$ e $1/10$ é 20. Use a janela Help Window do MATLAB para determinar a função que determina o menor múltiplo comum de dois números. Em seguida, use essa função para mostrar que o menor múltiplo comum entre:

- 4 e 10 é 20.
- 6 e 38 é 114.

18. A intensidade M de um terremoto na escala Richter é dada por: $M = \frac{2}{3} \log \frac{E}{E_0}$,

onde E é a energia liberada pelo tremor e $E_0 = 10^{4.4}$ Joules é uma constante (energia liberada em um tremor de terra de referência). Determine quantas vezes a energia liberada em um terremoto de 7.2 na escala Richter é maior do que um terremoto 5.3 nessa mesma escala.

CAPÍTULO 2

Criando Arranjos

Definitivamente, os arranjos são o modo-padrão de o MATLAB armazenar e manipular dados. Um arranjo é uma lista de números organizados em linhas e/ou colunas. O arranjo mais simples (unidimensional) é formado por uma linha ou uma coluna de números; o arranjo mais complexo (p.ex., bidimensional) é uma coleção de números organizados em linhas e colunas. Frequentemente, um arranjo está ligado ao armazenamento de informação e dados, como em uma tabela. Na engenharia (e em outras ciências), em geral os arranjos unidimensionais representam os vetores e os arranjos bidimensionais representam as matrizes. Este capítulo mostra como criar e referenciar arranjos; o Capítulo 3 mostra como usar arranjos em operações matemáticas básicas. Além dos arranjos constituídos apenas de números, os arranjos do MATLAB também podem incluir uma cadeia de caracteres, denominada *string*. As *strings* são discutidas na Seção 2.10.

2.1 CRIANDO UM ARRANJO UNIDIMENSIONAL (VETOR)

Um arranjo unidimensional é uma lista de números dispostos em uma linha ou coluna. Um exemplo de arranjo é a representação da posição de um ponto no espaço tridimensional no sistema de coordenadas cartesianas. De acordo com a Fig. 2-1, a posição de um ponto *A* é definida por uma lista de três números, 2, 4 e 5, que são, essencialmente, as coordenadas do ponto no espaço em relação à origem do sistema de coordenadas.

A posição de um ponto genérico *A* pode ser expressa em termos de um vetor posição:

$$\mathbf{r}_A = 2\mathbf{i} + 4\mathbf{j} + 5\mathbf{k}$$

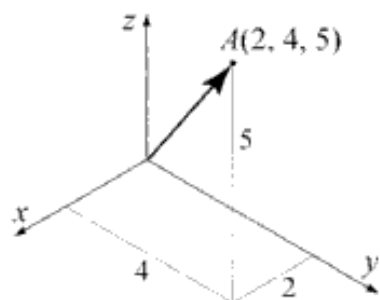


FIGURA 2-1 Posição de um ponto.

onde **i**, **j** e **k** são os vetores unitários nas direções *x*, *y* e *z*, respectivamente. Os números 2, 4 e 5 podem ser utilizados para definir um vetor linha ou coluna.

Qualquer lista de números pode ser tratada como um vetor. Por exemplo, a Tabela 2-1 representa dados sobre o crescimento populacional que podem ser utilizados para criar duas listas de números: uma referente a anos e outra da população em si. Cada lista pode ser interpretada como sendo constituída de entradas, elementos de um vetor, organizados em linha ou coluna.

TABELA 2-1 Dados populacionais

Ano	1984	1986	1988	1990	1992	1994	1996
População (milhões)	127	130	136	145	158	178	221

No MATLAB, a criação de um vetor acontece atribuindo-se os elementos desse vetor a uma variável. Isso pode ser feito de muitos modos diferentes, dependendo da fonte de informação utilizada na geração dos elementos do vetor. Quando um vetor possui apenas números conhecidos (como as coordenadas do ponto A), o valor de cada elemento é inicializado diretamente. É possível ainda fazer de cada elemento do vetor uma expressão matemática, incluindo variáveis previamente declaradas, números e funções. Frequentemente, os elementos de um vetor linha são uma série de números incrementados (ou decrementados) a partir de um fator constante. Nesses casos, o vetor pode ser criado a partir de comandos nativos do MATLAB. Um vetor pode ser criado como o resultado de uma operação matemática, mas isso será analisado no Capítulo 3.

Criando um vetor a partir de uma lista de números conhecidos:

O vetor é criado digitando-se os números da lista dentro de colchetes [].

```
Nome_variável = [ digite os elementos do vetor ]
```

Vetor linha: para criar um vetor linha, digite os elementos dentro dos colchetes, separando-os por um espaço ou uma vírgula.

Vetor coluna: para criar um vetor coluna, digite os elementos dentro dos colchetes a partir do colchete esquerdo [. Então, entre com os elementos separando-os por ponto-e-vírgula ou pressionando a tecla **Enter** após cada elemento. Por fim, digite o colchete] para terminar a criação do vetor coluna.

O Tutorial 2-1 mostra como os dados da Tabela 2-1 e as coordenadas do ponto A são usados para criar vetores linha e coluna.

TUTORIAL 2-1 Criando vetores a partir de um conjunto conhecido de dados

```
>> ano=[1984 1986 1988 1990 1992 1994 1996]
```

```
A lista de anos é atribuída ao vetor linha denominado ano.
```

TUTORIAL 2-1 Criando vetores a partir de um conjunto conhecido de dados

```

ano =
    1984    1986    1988    1990    1992    1994    1996
>> pop = [127; 130; 136; 145; 158; 178; 221]
pop =
    127
    130
    136
    145
    158
    178
    221
>> pontoAL = [2, 4, 5]
pontoAL =
     2     4     5
>> pontoAC = [2
4
5]
pontoAC =
     2
     4
     5
>>

```

Os dados populacionais são atribuídos ao vetor coluna `pop`.

As coordenadas do ponto A são atribuídas ao vetor linha denominado `pontoAL`.

As coordenadas do ponto A são atribuídas ao vetor linha denominado `pontoAC`. (A tecla **Enter** foi pressionada após a digitação de cada elemento.)

Criando um vetor com elementos espaçados de um fator constante:

Neste tipo de vetor, os elementos possuem a mesma separação entre si. Por exemplo, no vetor $v = 2\ 4\ 6\ 8\ 10$, o incremento entre os elementos é 2. Um vetor cujo primeiro elemento é m , o incremento é q e o último termo é n , pode ser criado digitando-se:

Nome_variável = [m:q:n]

ou

Nome_variável = m:q:n

(Os colchetes são opcionais)

Alguns exemplos:

```

>> x = [1:2:13]
x =
     1     3     5     7     9    11    13
>> y = [1.5:0.1:2.1]
y =
    1.5000    1.6000    1.7000    1.8000    1.9000    2.0000    2.1000

```

Primeiro elemento 1, espaçamento 2 e último elemento 13.

Primeiro elemento 1.5, espaçamento 0.1 e último elemento 2.1.

```
>> z = [-3: 7]
```

```
z =
```

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

Primeiro elemento -3, último elemento 7. Se o espaçamento é omitido, o padrão é 1.

```
>> xa = [21:-3:6]
```

```
xa =
```

```
21 18 15 12 9 6
```

Primeiro elemento 21, espaçamento -3 e último elemento 6.

```
>>
```

- Se os números m , q e n forem tais que o valor de n não possa ser obtido a partir de q incrementos do valor inicial de m , então (sendo n positivo) o último elemento do vetor será o último número que não exceder o valor de n .

Criando um vetor especificando o primeiro e último termos; e, em seguida, o número de termos:

Um vetor cujo primeiro elemento é xi , o último elemento é xf e o número de elementos é n , pode ser criado por meio do comando `linspace` (o MATLAB determina o incremento/decremento correto):

```
Nome_variável = linspace (xi, xf, n)
```

Alguns exemplos:

```
>> va = linspace(0,8,6)
```

```
va =
```

```
0 1.6000 3.2000 4.8000 6.4000 8.0000
```

6 elementos, primeiro elemento 0, último elemento 8.

```
>> vb = linspace(30,10,11)
```

```
vb =
```

```
30 28 26 24 22 20 18 16 14 12 10
```

11 elementos, primeiro elemento 30, último elemento 10.

```
>> u = linspace(49.5,0.5)
```

```
u =
```

```
Columns 1 through 10
```

```
49.5000 49.0051 48.5101 48.0152 47.5202 47.0253 46.5303 46.0354 45.5404  
45.0455
```

Primeiro elemento 49.5, último elemento 0.5.

Quando o número de elementos for omitido, o padrão será 100.

```
.....
```

```
Columns 91 through 100
```

```
4.9545 4.4596 3.9646 3.4697 2.9747 2.4798 1.9848 1.4899  
0.9949 0.5000
```

São mostrados ao todo 100 elementos.

```
>>
```


2.2 CRIANDO ARRANJOS BIDIMENSIONAIS (MATRIZES)

Um arranjo bidimensional (matriz) possui elementos dispostos em linhas e colunas. As matrizes muitas vezes são utilizadas para armazenar informação na forma de uma tabela de números ou *strings*. Além disso, exercem um papel importante na álgebra linear e são usadas na engenharia (e em outras ciências) para descrever muitas grandezas físicas.

Em uma matriz quadrada, o número de linhas e colunas é igual. Por exemplo, a matriz:

```
7  4  9
3  8  1    (matriz 3 × 3)
6  5  3
```

é quadrada, porque possui três linhas e três colunas. Em geral, o número de linhas e colunas é diferente. Por exemplo, a matriz:

```
31 26 14 18  5 30
 3 51 20 11 43 65    (matriz 4 × 6)
28  6 15 61 34 22
14 58  6 36 93  7
```

possui quatro linhas e seis colunas. Uma matriz $m \times n$ possui m linhas e n colunas, m por n é o tamanho (dimensão) da matriz.

Uma matriz é criada atribuindo-se os elementos do arranjo a uma variável. Isso é feito digitando-se os elementos, linha por linha, dentro de colchetes []. Primeiramente, digite o colchete esquerdo [; então, digite a primeira linha, separando os elementos com espaços ou vírgulas. Para digitar a próxima linha, digite um ponto-e-vírgula ou pressione **Enter**. Termine a matriz digitando o colchete] à direita do último elemento:

Nome_variável = [1ª linha de elementos; 2ª linha de elementos;
3ª linha de elementos;.....; última linha de elementos]

As entradas de uma matriz podem conter números ou expressões matemáticas (formadas de números, variáveis declaradas e funções). É sumariamente importante que todas as linhas possuam a mesma quantidade de elementos. Se um elemento na linha vale zero, ele deve ser digitado. O MATLAB exibe uma mensagem de erro, caso seja feita alguma tentativa de declarar uma matriz de modo incompleto. Exemplos de matrizes, declaradas de diferentes maneiras, são mostrados a seguir, no Tutorial 2-2.

TUTORIAL 2-2 Criando matrizes

```
>> a = [5 35 43; 4 76 81; 21 32 40]
```

```
a =
```

```
5 35 43
```

```
4 76 81
```

```
21 32 40
```

```
>> b = [7 2 76 33 8
```

```
1 98 6 25 6
```

Um ponto-e-vírgula deve ser digitado para marcar o início de uma nova linha.

A tecla **Enter** é pressionada após a digitação de cada uma das linhas desejadas.

TUTORIAL 2-2 Criando matrizes

(continuação)

```
5 54 68 9 0]
```

```
b =
```

```
7 2 76 33 8
```

```
1 98 6 25 6
```

```
5 54 68 9 0
```

```
>> cd = 6; e = 3; h = 4;
```

São declaradas e inicializadas três variáveis.

```
>> Mat = [e, cd*h, cos(pi/3); h^2, sqrt(h*h/cd), 14]
```

```
Mat =
```

```
3.0000 24.0000 0.5000
```

```
16.0000 1.6330 14.0000
```

Os elementos da matriz são definidos por expressões matemáticas.

```
>>
```

As linhas de uma matriz também podem ser geradas utilizando-se vetores, por meio do comando `linspace` ou da notação para criação de vetores com incremento (ou decremento) constante. Por exemplo:

```
>> A = [1:2:11; 0:5:25; linspace(10,60,6); 67 2 43 68 4 13]
```

```
A =
```

```
1 3 5 7 9 11
```

```
0 5 10 15 20 25
```

```
10 20 30 40 50 60
```

```
67 2 43 68 4 13
```

```
>>
```

No exemplo anterior, as duas primeiras linhas foram geradas como vetores usando a notação de incremento (ou decremento); a terceira linha foi gerada usando o comando `linspace`; e os elementos da última linha foram digitados um a um.

2.2.1 Os comandos zeros, ones e eye

Os comandos `zeros (m, n)`, `ones (m, n)` e `eye (n)` podem ser utilizados para criar matrizes cujos elementos possuem valores especiais. Os dois primeiros comandos, `zeros (m, n)` e `ones (m, n)`, criam matrizes com m linhas e n colunas, cujos elementos são os números 0 e 1, respectivamente. O comando `eye (n)` cria uma matriz quadrada com n linhas e n colunas, cujos elementos da diagonal principal são iguais a 1 e os demais elementos são 0. Essa matriz é denominada matriz identidade. Alguns exemplos:

```
>> zr=zeros(3,4)
zr =
     0     0     0     0
     0     0     0     0
     0     0     0     0
>> um = ones(4,3)
um =
     1     1     1
     1     1     1
     1     1     1
     1     1     1
>> idn = eye(5)
idn =
     1     0     0     0     0
     0     1     0     0     0
     0     0     1     0     0
     0     0     0     1     0
     0     0     0     0     1
>>
```

Matrizes também podem ser originadas como o resultado de operações matemáticas envolvendo outros vetores/matrizes. Esse tópico será tratado no Capítulo 3.

2.3 OBSERVAÇÕES QUANTO AO USO DE VARIÁVEIS NO MATLAB

- Todas as variáveis no MATLAB são de fato arranjos. Um escalar é um arranjo com um único elemento; um vetor é um arranjo de elementos numa única linha ou coluna, e uma matriz é um arranjo de elementos em linhas e colunas.
- Uma variável (escalar, vetor ou matriz) torna-se declarada quando lhe é atribuído algum valor. Não é necessário definir o tamanho do arranjo antes de os elementos serem atribuídos à variável (um elemento, no caso de escalar; uma linha ou coluna de elementos, no caso do vetor; e um arranjo de elementos bidimensional, no caso da matriz).
- Uma vez que uma variável tenha sido declarada (como escalar, vetor ou matriz), podemos modificá-la para qualquer outro tamanho ou tipo diferente do original. Por exemplo, um escalar pode ser modificado de modo a tornar-se um vetor ou uma matriz; um vetor pode ser colocado sob a forma de escalar, um vetor de tamanho diferente ou uma matriz; uma matriz pode ter seu tamanho (dimensão) reduzido ou aumentado, até tornar-se um vetor ou escalar. Essas mudanças são realizadas adicionando-se (ou retirando-se) elementos do arranjo. Esse tópico será discutido nas Seções 2.7 e 2.8.

2.4 O OPERADOR DE TRANSPOSIÇÃO

O operador de transposição, quando aplicado a um vetor, permuta um vetor linha para vetor coluna e vice-versa. Em matrizes, troca as linhas da matriz pelas colunas e vice-versa. O operador de transposição é aplicado ao vetor ou à matriz digitando-se uma aspa simples (') após a variável a ser transposta. Observe os seguintes exemplos:

```
>> aa = [3 8 1]
```

```
aa =
```

```
3 8 1
```

```
>> bb = aa'
```

```
bb =
```

```
3
```

```
8
```

```
1
```

```
>> C = [2 55 14 8; 21 5 32 11; 41 64 9 1]
```

```
C =
```

```
2 55 14 8
```

```
21 5 32 11
```

```
41 64 9 1
```

```
>> D = C'
```

```
D =
```

```
2 21 41
```

```
55 5 64
```

```
14 32 9
```

```
8 11 1
```

```
>>
```

Declara o vetor linha aa.

Declara o vetor coluna bb como o vetor transposto de aa.

Declara uma matriz C com 3 linhas e 4 colunas.

Declara a matriz D como matriz transposta da matriz C. (A matriz D possui 4 linhas e 3 colunas.

2.5 REFERÊNCIA A UM ELEMENTO DO ARRANJO

Elementos em um arranjo (vetor ou matriz) podem ser referenciados (endereçados) individualmente ou em subgrupos. Isso é muito útil quando é necessário redefinir apenas um dos elementos ou grupo de elementos de um arranjo ou usar o valor de um elemento específico em cálculos, ou ainda, quando elementos em um subgrupo são usados para redefinir uma nova variável.

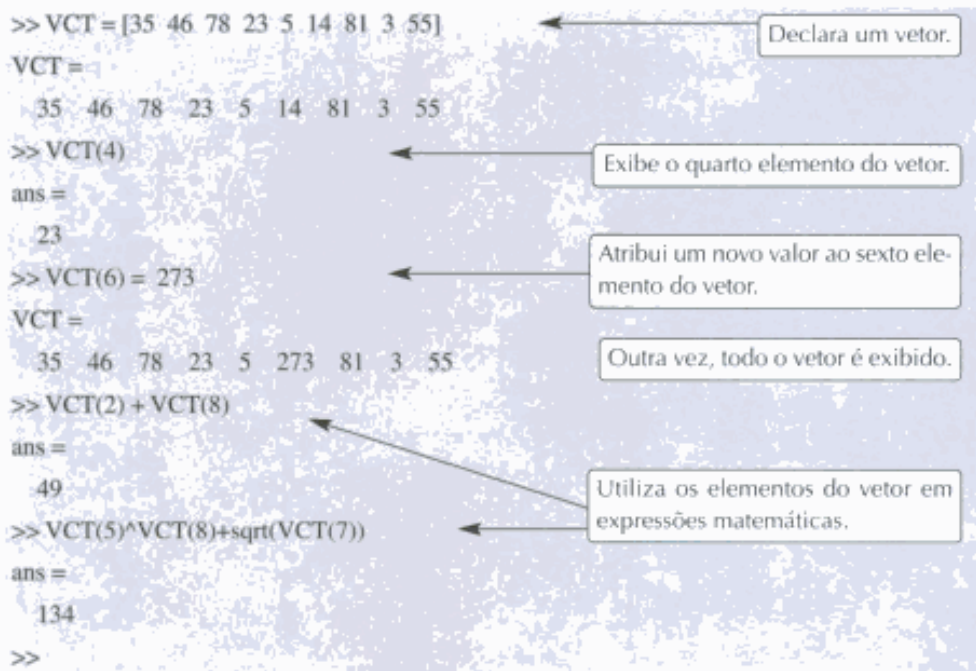
2.5.1 Vetor

Fazer referência a um elemento de um vetor é indicar a posição que ele ocupa na linha ou coluna desse vetor. Para um vetor *ve*, *ve(k)* referencia o elemento na posição *k*. A primeira posição, mais à esquerda do vetor, é a número 1. Por exemplo, se o vetor *ve* possui nove elementos:

```
ve = 35 46 78 23 5 14 81 3 55
```

então, $ve(4) = 23$, $ve(7) = 81$ e $ve(1) = 35$.

Um único elemento do vetor, $v(k)$, pode ser usado como uma variável. Por exemplo, é possível mudar o valor desse elemento atribuindo-lhe um novo valor na posição de referência. Isso pode ser feito digitando $v(k) = \text{ novo valor}$. Um único elemento também pode ser utilizado como uma variável numa expressão matemática. Alguns exemplos:



```

>> VCT = [35 46 78 23 5 14 81 3 55]
VCT =
    35    46    78    23     5    14    81     3    55
>> VCT(4)
ans =
    23
>> VCT(6) = 273
VCT =
    35    46    78    23     5   273    81     3    55
>> VCT(2) + VCT(8)
ans =
    49
>> VCT(5)^VCT(8)+sqrt(VCT(7))
ans =
   134
>>
  
```

Declara um vetor.

Exibe o quarto elemento do vetor.

Atribui um novo valor ao sexto elemento do vetor.

Outra vez, todo o vetor é exibido.

Utiliza os elementos do vetor em expressões matemáticas.

2.5.2 Matriz

A posição de referência de um elemento em uma matriz é definida especificando-se o número da linha e da coluna que ele ocupa. Atribuindo-se uma matriz a uma variável ma , o símbolo $ma(k,p)$ faz referência ao elemento na linha k e na coluna p .

Por exemplo, se a matriz é: $ma = \begin{bmatrix} 3 & 11 & 6 & 5 \\ 4 & 7 & 10 & 2 \\ 13 & 9 & 0 & 8 \end{bmatrix}$

então, $ma(1,1) = 3$ e $ma(2,3) = 10$.

Como aconteceu com os vetores, é possível modificar o valor de um único elemento da matriz atribuindo-lhe um novo valor. Igualmente, os elementos da matriz podem ser usados como variáveis em expressões e funções matemáticas. A seguir, apresentamos alguns exemplos:

```
>> MAT = [3 11 6 5; 4 7 10 2; 13 9 0 8]
```

Cria uma matriz 3 × 4.

```
MAT =
```

```
    3    11     6     5
    4     7    10     2
   13     9     0     8
```

```
>> MAT(3,1) = 20
```

Atribui um novo valor ao elemento na posição (3,1).

```
MAT =
```

```
    3    11     6     5
    4     7    10     2
   20     9     0     8
```

```
>> MAT(2,4) - MAT(1,2)
```

Utiliza os elementos (2,4) e (1,2) em expressões matemáticas.

```
ans =
```

```
    -9
```

```
>>
```

2.6 DOIS PONTOS (:) REFERENCIANDO ELEMENTOS DE ARRANJOS

Podemos usar dois pontos (:) para referenciar uma faixa de elementos dentro de um vetor ou de uma matriz.

Para um vetor:

$va(:)$ → referencia todos os elementos do vetor va (na linha ou coluna do vetor).

$va(m:n)$ → referencia os elementos entre as posições m e n do vetor va .

Exemplo:

```
>> v = [4 15 8 12 34 2 50 23 11]
```

Declara um vetor v.

```
v =
```

```
    4    15     8    12    34     2    50    23    11
```

```
>> u = v(3:7)
```

```
u =
```

```
    8    12    34     2    50
```

O vetor u é criado a partir dos elementos entre as posições 3 e 7 do vetor v.

```
>>
```

Para uma matriz:

$A(:,n)$ → referencia os elementos da matriz A em todas as linhas da coluna n .

$A(n,:)$ → referencia os elementos da matriz A em todas as colunas da linha n .

$A(:, m:n)$ → referencia os elementos da matriz A em todas as linhas entre as colunas m e n .

$A(m:n,:)$ → referencia os elementos da matriz A em todas as colunas entre as linhas m e n .

$A(m:n, p:q) \rightarrow$ referencia os elementos da matriz A entre as linhas m e n e as colunas p e q .

O Tutorial 2-3 ensina como usar o operador *dois pontos* para referenciar elementos de matrizes.

TUTORIAL 2-3 Usando dois pontos no endereçamento de arranjos

```
>> A=[1 3 5 7 9 11; 2 4 6 8 10 12; 3 6 9 12 15 18; 4 8 12 16 20 24; 5 10 15 20 25 30]
```

```
A =
```

```
1 3 5 7 9 11
2 4 6 8 10 12
3 6 9 12 15 18
4 8 12 16 20 24
5 10 15 20 25 30
```

Declara uma matriz A com 5 linhas e 6 colunas.

```
>> B=A(:,3)
```

```
B =
```

```
5
6
9
12
15
```

Define um vetor coluna B a partir de todos elementos da coluna 3 da matriz A.

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

```
C =
```

```
2 4 6 8 10 12
```

Declara um vetor linha C a partir de todos elementos da linha 2 da matriz A.

```
>> E=A(2:4,:)
```

```
E =
```

```
2 4 6 8 10 12
3 6 9 12 15 18
4 8 12 16 20 24
```

Define uma matriz E a partir de todos elementos entre as linhas 2 e 4 da matriz A.

```
>> F=A(1:3,2:4)
```

```
F =
```

```
3 5 7
4 6 8
6 9 12
```

Define uma matriz F a partir dos elementos situados entre as linhas 1-3 e colunas 2-4 da matriz A.

```
>>
```

No Tutorial 2-3, novos vetores e matrizes foram criados partindo-se de uma matriz previamente declarada, usando-se um conjunto de elementos ou um grupo de elementos dessa matriz. Entretanto, é possível selecionar elementos específicos (linhas e/ou colunas) de variáveis previamente declaradas para criar novas variáveis. Isto po-

de ser feito digitando os elementos das linhas (colunas) dentro de colchetes, como é mostrado a seguir:

```
>> v = 4:3:34
```

```
v =
```

```
4 7 10 13 16 19 22 25 28 31 34
```

Declarando um vetor *v* com 11 elementos.

```
>> u = v([3, 5, 7:10])
```

```
u =
```

```
10 16 22 25 28 31
```

Cria um vetor *u* a partir do terceiro, quinto e do sétimo ao décimo elementos de *v*.

```
>> A = [10:-1:4; ones(1,7); 2:2:14; zeros(1,7)]
```

```
A =
```

```
10 9 8 7 6 5 4
```

```
1 1 1 1 1 1 1
```

```
2 4 6 8 10 12 14
```

```
0 0 0 0 0 0 0
```

Cria uma matriz *A* de dimensões 4×7 .

```
>> B = A([1,3],[1,3,5:7])
```

```
B =
```

```
10 8 6 5 4
```

```
2 6 10 12 14
```

Cria uma matriz *B* a partir da primeira e terceira linhas e da primeira, terceira, quinta à sétima colunas de *A*.

2.7 ADICIONANDO ELEMENTOS ÀS VARIÁVEIS DECLARADAS

Uma variável tipo vetor ou matriz pode ser modificada pela adição de elementos na estrutura da mesma (lembre-se que um escalar é um vetor com um elemento). Um vetor (matriz de uma única linha ou coluna) pode ser modificado acrescentando novos elementos ou, então, pode ter a dimensão acrescida, ou seja, ser colocado na forma matricial (bidimensional). Além disso, linhas e/ou colunas podem ser adicionadas a uma matriz declarada para obter uma matriz de dimensão diferente. O acréscimo de elementos pode ser realizado simplesmente via atribuição de valores aos elementos que se deseja adicionar ou por anexação de elementos pertencentes a alguma variável previamente declarada.

Adicionando elementos ao vetor:

Conforme foi mencionado, elementos podem ser adicionados atribuindo-se valores aos novos elementos do arranjo. Por exemplo, se um vetor possui 4 elementos, podemos aumentá-lo atribuindo valores aos elementos 5, 6, e assim por diante. Se um vetor possui *n* elementos e um novo valor é atribuído ao elemento cuja referência é *n* + 2 ou maior, o MATLAB encarrega-se de atribuir zeros aos elementos posicionados entre o último elemento original e o novo elemento terminal do vetor. Exemplos:

```
>> DF= 1:4
```

```
DF =
```

```
1 2 3 4
```

Declara o vetor *DF* com 4 elementos.

```
>> DF(5:10) = 10:5:35
```

```
DF =
```

```
1 2 3 4 10 15 20 25 30 35
```

```
>> AD = [5 7 2]
```

```
AD =
```

```
5 7 2
```

```
>> AD(8) = 4
```

```
AD =
```

```
5 7 2 0 0 0 0 4
```

```
>> AR(5) = 24
```

```
AR =
```

```
0 0 0 0 24
```

```
>>
```

Adiciona 6 elementos a partir do quinto elemento de DF.

Declara o vetor AD com três elementos.

Atribui um valor ao oitavo elemento de AD.

O MATLAB adiciona zeros aos elementos nas posições 4 a 7.

Atribui um valor ao quinto elemento de um novo vetor AR.

O MATLAB adiciona zeros aos elementos nas posições 1 a 4.

Elementos também podem ser adicionados por meio da anexação de vetores já declarados. Dois exemplos:

```
>> RE = [3 8 1 24];
```

```
>> GT = 4:3:16;
```

```
>> KNH = [RE GT]
```

```
KNH =
```

```
3 8 1 24 4 7 10 13 16
```

```
>> KNV = [RE'; GT']
```

```
KNV =
```

```
3
```

```
8
```

```
1
```

```
24
```

```
4
```

```
7
```

```
10
```

```
13
```

```
16
```

Declara o vetor RE com 4 elementos.

Declara o vetor GT com 5 elementos.

Define um novo vetor KNH anexando o vetor GT no final do vetor RE.

Cria um novo vetor coluna KNV anexando GT' no final do vetor RE'.

Adicionando elementos a uma matriz:

Conforme mencionado, linhas e colunas podem ser inseridas em uma matriz previamente declarada. Isso é feito atribuindo-se valores aos novos elementos situados entre as linhas e colunas da matriz. É possível também aumentar o tamanho de uma matriz anexan-

do variáveis existentes. Entretanto, é necessário muita cautela, porque o tamanho das linhas ou colunas adicionadas deve estar de acordo com a definição da matriz original. Exemplos:

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

```
E =
```

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

```
>> E(3,:) = [10;4;22]
```

```
E =
```

```
1 2 3 4
5 6 7 8
10 14 18 22
```

```
>> K = eye(3)
```

```
K =
```

```
1 0 0
0 1 0
0 0 1
```

```
>> G = [E K]
```

```
G =
```

```
1 2 3 4 1 0 0
5 6 7 8 0 1 0
10 14 18 22 0 0 1
```

Declara uma matriz E de dimensões 2×4 .

Adiciona o vetor 10 14 18 22 para formar a terceira linha de E.

Declara uma matriz identidade K de dimensões 3×3 .

Anexa a matriz K à matriz E para formar a nova matriz G. O número de linhas em E e K deve ser o mesmo.

Se uma matriz possui a dimensão $m \times n$ e um novo valor é atribuído a um elemento cuja posição de referência extrapola o tamanho da matriz, o MATLAB aumenta o tamanho da matriz para incluir esse elemento novo. Outra vez, zeros são atribuídos aos demais elementos adicionados, mas não inicializados. Exemplos:

```
>> AW = [3 6 9; 8 5 11]
```

```
AW =
```

```
3 6 9
8 5 11
```

```
>> AW(4,5) = 17
```

```
AW =
```

```
3 6 9 0 0
8 5 11 0 0
0 0 0 0 0
```

Declara uma matriz 2×3 .

Atribui um valor ao elemento na posição (4,5).

O MATLAB modifica o tamanho da matriz para 4×5 e atribui zeros aos novos elementos.

```

0 0 0 0 17
>> BG(3,4) = 15
BG =
0 0 0 0
0 0 0 0
0 0 0 15
>>

```

Atribui um valor ao elemento na posição (3,4) da nova matriz.

O MATLAB cria uma matriz 4×5 e atribui zeros a todos elementos, exceto o BG(3,4).

2.8 ELIMINANDO ELEMENTOS DE UMA MATRIZ

Um elemento ou um grupo de elementos em uma variável declarada pode(m) ser apagado(s) atribuindo vazio a esse elemento. Isso é feito usando-se colchetes sem nenhum espaço ou caractere entre eles. Apagando elementos, um vetor pode ser colocado na forma escalar e uma matriz pode ser redefinida como um vetor. Exemplos:

```
>> kt = [2 8 40 65 3 55 23 15 75 80]
```

```
kt =
```

```
2 8 40 65 3 55 23 15 75 80
```

```
>> kt(6) = []
```

```
kt =
```

```
2 8 40 65 3 23 15 75 80
```

```
>> kt(3:6) = []
```

```
kt =
```

```
2 8 15 75 80
```

```
>> mtr = [5 78 4 24 9; 4 0 36 60 12; 56 13 5 89 3]
```

```
mtr =
```

```
5 78 4 24 9
```

```
4 0 36 60 12
```

```
56 13 5 89 3
```

```
>> mtr(:,2:4) = []
```

```
mtr =
```

```
5 9
```

```
4 12
```

```
56 3
```

```
>>
```

Declara um vetor com 10 elementos.

Elimina o sexto elemento do vetor kt.

Agora, o vetor possui 9 elementos

Elimina os elementos situados entre as posições 3 a 6 do vetor kt.

O novo vetor possui 5 elementos.

Declara uma matriz 3×5 .

Elimina as colunas 2, 3 e 4 da matriz mtr.

2.9 FUNÇÕES NATIVAS PARA MANIPULAÇÃO DE ARRANJOS

O MATLAB possui muitas funções nativas para gerenciamento e manipulação de arranjos. Algumas estão listadas abaixo:

TABELA 2-2 Funções nativas para manipulação de arranjos

Comando	Descrição	Exemplo
<code>length(A)</code>	Retorna o número de elementos no vetor A.	<pre>>> A = [5 9 2 4]; >> length(A) ans = 4</pre>
<code>size(A)</code>	Retorna um vetor linha $[m, n]$, onde m e n representam a dimensão ($m \times n$) da matriz A.	<pre>>> A = [6 1 4 0 12; 5 19 6 8 2] A = 6 1 4 0 12 5 19 6 8 2 >> size(A) ans = 2 5</pre>
<code>reshape(A, m, n)</code>	Rearranja uma matriz A com r linhas e s colunas para m linhas e n colunas. r vezes s deve ser numericamente igual a m vezes n .	<pre>>> A = [5 1 6; 8 0 2] A = 5 1 6 8 0 2 >> B = reshape(A, 3, 2) B = 5 0 8 6 1 2</pre>
<code>diag(v)</code>	Quando v é um vetor, cria uma matriz quadrada contendo os elementos de v na diagonal principal.	<pre>>> v = [7 4 2]; >> A = diag(v) A = 7 0 0 0 4 0 0 0 2</pre>
<code>diag(A)</code>	Quando A é uma matriz, cria um vetor coluna a partir dos elementos na diagonal principal.	<pre>>> A = [1 2 3; 4 5 6; 7 8 9] A = 1 2 3 4 5 6 7 8 9 >> ve = diag(A) ve = 1 5 9</pre>

No Help Window, são descritas funções nativas adicionais para manipulação de arranjos. Selecione “Functions by Category”, depois “Mathematics” e em seguida “Arrays and Matrices”.

No Capítulo 1 foi mencionado que o comando `who` exibe uma lista de variáveis declaradas pelo MATLAB (armazenadas na memória). O comando `whos`, além de listar as variáveis declaradas na memória, exibe o tamanho que elas ocupam (em *bytes*) e a respectiva classe de armazenamento. O exemplo a seguir ilustra o uso de `whos`:

```
>> a = 7;
>> E = 3;
>> d = [5, a+E, 4, E^2]
d =
```

```
    5    10     4     9
```

```
>> g = [a,a^2,13; a*E,1,a^E]
```

```
g =
```

```
    7    49    13
```

```
   21     1   343
```

```
>> who
```

```
Your variables are:
```

```
E a d g
```

```
>> whos
```

Name	Size	Bytes Class
E	1×1	8 double array
a	1×1	8 double array
d	1×4	32 double array
g	2×3	48 double array

```
Grand total is 12 elements using 96 bytes
```

```
>>
```

Criando as variáveis a, E, d e g.

O comando `who` exibe as variáveis ativas na memória do computador e gerenciadas pelo MATLAB.

O comando `whos` exibe as variáveis ativas na memória do computador juntamente com o espaço (em *bytes*) que elas ocupam na memória do computador.

Problema-exemplo 2-1: Criando uma matriz formada de zeros e uns – parte 1

Usando os comandos `ones` e `zeros`, crie uma matriz 4×5 cujas primeiras duas linhas sejam formadas de 0 e as duas linhas seguintes sejam formadas de 1.

Solução

```
>> A(1:2,:) = zeros(2,5)
```

```
A =
```

```
    0    0    0    0    0
```

Cria inicialmente uma matriz de zeros 2×5 .

```

0 0 0 0 0
>> A(3:4,:) = ones(2,5)

```

Adiciona as linhas 3 e 4 formadas de uns.

```

A =
0 0 0 0 0
0 0 0 0 0
1 1 1 1 1
1 1 1 1 1
>>

```

Problema-exemplo 2-2: Criando uma matriz formada de zeros e uns – parte 2

Crie uma matriz 6×6 cujos elementos nas linhas/colunas 3 e 4 são uns (1s), sendo o restante dos elementos da matriz zeros (0s).

Solução

```

>> AR = zeros(6,6)
AR =
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0

```

Cria inicialmente uma matriz de zeros 6×6 .

```

>> AR(3:4,:) = ones(2,6)
AR =
0 0 0 0 0 0
0 0 0 0 0 0
1 1 1 1 1 1
1 1 1 1 1 1
0 0 0 0 0 0
0 0 0 0 0 0

```

Modifica as linhas 3 e 4 para 1s.

```

>> AR(:,3:4) = ones(6,2)
AR =
0 0 1 1 0 0
0 0 1 1 0 0
1 1 1 1 1 1
1 1 1 1 1 1
0 0 1 1 0 0
0 0 1 1 0 0

```

Modifica as colunas 3 e 4 para 1s.

Problema-exemplo 2-3: Manipulação de matrizes

Dadas duas matrizes, A (5×6) e B (3×6), e um vetor v de 9 elementos:

$$A = \begin{bmatrix} 2 & 5 & 8 & 11 & 14 & 17 \\ 3 & 6 & 9 & 12 & 15 & 18 \\ 4 & 7 & 10 & 13 & 16 & 19 \\ 5 & 8 & 11 & 14 & 17 & 20 \\ 6 & 9 & 12 & 15 & 18 & 21 \end{bmatrix} \quad B = \begin{bmatrix} 5 & 10 & 15 & 20 & 25 & 30 \\ 30 & 35 & 40 & 45 & 50 & 55 \\ 55 & 60 & 65 & 70 & 75 & 80 \end{bmatrix}$$

$$v = [99 \ 98 \ 97 \ 96 \ 95 \ 94 \ 93 \ 92 \ 91]$$

Crie os três arranjos na janela Command Window e então, usando apenas um comando, substitua os quatro últimos elementos da primeira e terceira linhas de A pelos quatro primeiros elementos da primeira e segunda linhas de B . Em seguida, substitua os quatro últimos elementos da quarta linha de A pelos elementos nas posições de 5 a 8 de v . Por fim, substitua os quatro últimos elementos da quinta linha de A pelos elementos da última linha de B , compreendidos entre as colunas 2 e 5 da terceira linha de B .

Solução

```
>> A = [2:3:17; 3:3:18; 4:3:19; 5:3:20; 6:3:21]
```

```
A =
```

```
2   5   8  11  14  17
3   6   9  12  15  18
4   7  10  13  16  19
5   8  11  14  17  20
6   9  12  15  18  21
```

```
>> B = [5:5:30; 30:5:55; 55:5:80]
```

```
B =
```

```
5   10  15  20  25  30
30  35  40  45  50  55
55  60  65  70  75  80
```

```
>> v = [99:-1:91]
```

```
v =
```

```
99  98  97  96  95  94  93  92  91
```

```
>> A([1 3 4 5],3:6) = [B([1 2], 1:4); v(5:8); B(3,2:5)]
```

Matriz 4×4 construída a partir das colunas 3 a 6 e das linhas 1, 3, 4 e 5.

Matriz 4×4 . As duas primeiras linhas são os elementos das colunas 1 a 4, compreendidos entre as linhas 1 e 2 da matriz B . A terceira linha recebe os elementos 5 a 8 do vetor v . A quarta linha recebe os elementos das colunas 2 a 5 da linha 3 da matriz B .

A =

2	5	5	10	15	20
3	6	9	12	15	18
4	7	30	35	40	45
5	8	95	94	93	92
6	9	60	65	70	75

2.10 CADEIA DE CARACTERES (STRINGS) E VARIÁVEIS TIPO STRINGS

- Uma *string* é uma cadeia de caracteres organizada em um arranjo. Para criá-la basta digitar os caracteres entre aspas simples (' ').
- *Strings* podem conter letras, números, espaços e outros símbolos.
- Exemplos de *strings*: 'ad ef'; '3%fr2'; '{edcba:21!'; 'MATLAB'.
- Aspas simples também podem ser usadas dentro de *strings*. Para isso, são necessárias duas aspas dentro do corpo de uma *string* para que o MATLAB interprete como uma única aspa.
- Quando uma *string* estiver sendo digitada, os caracteres aparecem em roxo (púrpura) após a aspa de abertura da *string*. Fechando-se a segunda aspa, os caracteres mudam para marrom.

As *strings* podem ser utilizadas de maneiras e propósitos diferentes no MATLAB. Elas são usadas para imprimir mensagens de texto na tela (Capítulo 4), em comandos de formatação de gráficos (Capítulo 5) e como argumento de entrada de algumas funções (Capítulo 6).

- Ao se utilizar *strings* na formatação de gráficos (rótulos dos eixos, título e notas textuais), os caracteres dentro da *string* podem ser formatados em fontes específicas, tamanhos, formas (maiúscula e minúscula), cor, etc. Veja os detalhes no Capítulo 5.

Strings também podem ser atribuídas às variáveis. Para isso, basta digitar a *string* no lado direito do operador de atribuição, conforme os seguintes exemplos*:

```
>> a = 'FRty 8'
a =
FRty 8
>> B = 'My name is John Smith'
B =
My name is John Smith
>>
```

* N. de T.: As *strings* foram mantidas em inglês para preservar o link entre as explicações que aparecem no texto e os programas rodados no MATLAB. O leitor poderá escrevê-las em português.

Quando uma variável é declarada como *string*, os caracteres são armazenados formando um arranjo semelhante ao que foi descrito para números. Cada caractere, incluindo espaços, é um elemento do arranjo. Significa que uma *string* de uma linha é um vetor linha, sendo a quantidade de elementos do vetor numericamente igual à quantidade de caracteres contidas na *string*. Além disso, os elementos dos vetores são referenciados (endereçados) pela posição que ocupam no vetor. Por exemplo, no vetor B declarado anteriormente, o quarto elemento é a letra n, o décimo segundo elemento é J e assim por diante.

```
>> B(4)
ans =
n
>> B(12)
ans =
J
>>
```

Em um arranjo tipo *string* também é possível modificar elementos específicos referenciando-os diretamente e atribuindo-lhes os novos valores. Por exemplo, o vetor B anterior pode ter o nome John substituído por Bill:

```
>> B(12:15)='Bill'
B =
My name is Bill Smith
>>
```

Usando dois pontos para modificar os caracteres contidos nas posições 12 a 15 do vetor B.

Matrizes também podem conter *strings*. Como foi feito para números, digite ponto-e-vírgula (;) ou pressione a tecla **Enter** no final de cada linha. Cada linha deve ser digitada como *string*, ou seja, é necessário colocar aspas no início e no fim de cada uma delas. Além disso, o número de elementos em cada linha deve ser o mesmo. Isto pode causar problemas quando a intenção é criar linhas com palavras específicas. A solução nesses casos é adicionar espaços entre os caracteres de modo a ajustar a quantidade de caracteres nas linhas para igualá-la ao número de elementos.

O MATLAB possui uma função nativa, chamada *char*, que cria um arranjo de linhas tendo o mesmo número de caracteres, a partir de caracteres de entrada que possuem tamanhos diferentes. O MATLAB adiciona espaços para igualar as linhas menores àquela que possuir mais elementos (ou seja, a maior linha da matriz). Na função *char*, as linhas devem ser separadas por vírgulas de acordo com o formato:

```
nome_variável = char ('string_1', 'string_2',
                     'string_3', ..., 'string_N');
```

Por exemplo:

```
>> Info = char('Nome do estudante:', 'John Smith', 'Nota:', 'A+')
```

```
Info =
```

```
Nome do estudante:
```

```
John Smith
```

```
Nota:
```

```
A+
```

```
>>
```

À variável declarada como `Info` são atribuídos quatro *strings*, todas de tamanhos diferentes.

A função `char` cria um arranjo de quatro linhas, todas com o mesmo tamanho da maior *string* escrita no argumento da função `char`.

Uma variável pode ser declarada um número ou uma *string*, possuindo a mesma sequência numérica. Por exemplo, a variável `x` é declarada numérica, sendo-lhe atribuído o número 536. A variável `y` é declarada *string* e lhe são atribuídos os caracteres 536, nessa ordem.

```
>> x = 536
```

```
x =
```

```
536
```

```
>> y = '536'
```

```
y =
```

```
536
```

```
>>
```

Exibindo o conteúdo de `x` e `y` – e observando a semelhança entre eles –, supõe-se que `x` e `y` sejam iguais. Entretanto, as duas variáveis não são idênticas. A variável `x` pode ser empregada em expressões matemáticas, enquanto a variável `y` não*.

2.11 PROBLEMAS

1. Crie um vetor linha contendo os seguintes elementos: 32, 4, 81, $e^{2.5}$, 63, $\cos(\pi/3)$ e 14.12.
2. Crie um vetor coluna que possua os elementos: 55, 14, $\ln(51)$, 987, 0 e $5\sin(2.5\pi)$.
3. Crie um vetor linha, onde o primeiro elemento é 1 e o último 33. Produza números entre os dois, incremento-os de 2 unidades (1, 3, 5, ..., 33).

* N. de T.: Faça um teste com a soma `x + y`. O resultado é o que se esperava?

4. Crie um vetor coluna, onde o primeiro elemento é 15, o último -25 e o decremento vale -5 (o vetor coluna pode ser criado como o vetor transposto do vetor linha).
5. Crie um vetor linha contendo 15 elementos espaçados igualmente, onde o primeiro elemento é 7 e o último 40.
6. Crie um vetor coluna contendo 12 elementos espaçados igualmente, onde o primeiro elemento é -1 e o último -15.
7. Crie um vetor (`Aprim`) contendo 16 elementos, onde o primeiro número é 4 e o último 49. O espaçamento vale 3 unidades. Em seguida, usando o operador dois pontos, crie um segundo vetor (`Asegun`) que contenha 8 elementos. Os quatro primeiros elementos de `Aprim` devem ser os primeiros de `Asegun`. Os demais elementos de `Asegun` são os quatro últimos de `Aprim`.
8. Construa a matriz indicada abaixo usando a notação vetorial para declarar vetores cujos elementos sejam espaçados igualmente e/ou o comando `linspace` para entrar com as linhas.

$$B = \begin{bmatrix} 1 & 4 & 7 & 10 & 13 & 16 & 19 & 22 & 25 \\ 72 & 66 & 60 & 54 & 48 & 42 & 36 & 30 & 24 \\ 0 & 0.125 & 0.250 & 0.375 & 0.500 & 0.625 & 0.750 & 0.875 & 1.000 \end{bmatrix}$$

9. Declare a seguinte matriz A :

$$A = \begin{bmatrix} 6 & 43 & 2 & 11 & 87 \\ 12 & 6 & 34 & 0 & 5 \\ 34 & 18 & 7 & 41 & 9 \end{bmatrix}$$

Use-a para:

- a) Criar um vetor linha `va` de cinco elementos contendo os elementos da segunda linha de A .
- b) Criar um vetor linha `vb` de três elementos que contenha os elementos da quarta coluna de A .
- c) Criar um vetor linha `vc` de dez elementos que contenha os elementos da primeira e segunda linhas de A .
- d) Criar um vetor linha `vd` de seis elementos que contenha os elementos da segunda e quinta colunas de A .

10. Declare a seguinte matriz C :

$$C = \begin{bmatrix} 2 & 4 & 6 & 8 & 10 \\ 3 & 6 & 9 & 12 & 15 \\ 7 & 14 & 21 & 28 & 35 \end{bmatrix}$$

Use-a para:

- Criar um vetor coluna ua de três elementos que contenha os elementos da terceira coluna de C .
- Criar um vetor coluna ub de cinco elementos que contenha os elementos da segunda linha de C .
- Criar um vetor coluna uc de nove elementos que contenha os elementos da primeira, terceira e quinta colunas de C .
- Criar um vetor ud de dez elementos que contenha os elementos da primeira e segunda linhas de C .

11. Declare a seguinte matriz A :

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 2 & 4 & 6 & 8 & 10 & 12 & 14 \\ 21 & 18 & 15 & 12 & 9 & 6 & 3 \\ 5 & 10 & 15 & 20 & 25 & 30 & 35 \end{bmatrix}$$

- Crie uma matriz B (3×4) a partir dos seguintes elementos de A : primeira, terceira e quarta linhas; entre a primeira, terceira, quinta e sétima colunas.
- Crie um vetor linha u de 15 elementos a partir dos seguintes elementos de A : terceira linha; entre a quinta e sétima colunas.

12. Usando os comandos `zeros`, `ones` e `eye`, crie os seguintes arranjos:

$$a) \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \quad b) \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad c) \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 1 \end{bmatrix}$$

13. Utilizando o comando `eye`, crie o arranjo A indicado abaixo (à esquerda). Então, usando o operador dois pontos para referenciar elementos no arranjo, reorganize a matriz A tal que seus elementos reassumam os valores mostrados na matriz A (à direita).

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad A = \begin{bmatrix} 2 & 2 & 2 & 0 & 5 & 5 & 5 \\ 2 & 2 & 2 & 0 & 5 & 5 & 5 \\ 3 & 3 & 3 & 0 & 5 & 5 & 5 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 4 & 4 & 7 & 0 & 9 & 9 & 9 \\ 4 & 4 & 7 & 0 & 9 & 9 & 9 \\ 4 & 4 & 7 & 0 & 9 & 9 & 9 \end{bmatrix}$$

14. Usando os comandos `zeros` e `ones`, crie uma matriz 3×5 onde a primeira, segunda e quinta colunas sejam formadas de 0s e as demais colunas sejam formadas de 1s.

15. Crie uma matriz 5×7 onde a primeira linha seja formada pelos números 1 2 3 4 5 6 7; a segunda linha sejam os números 8 9 10 11 12 13 14; a terceira linha os números de 15 a 21, e assim por diante, até completar a matriz. Partindo dessa matriz, crie uma nova matriz 3×4 a partir das linhas 2 a 4 e entre as colunas 3 a 6 da matriz inicial.
16. Crie uma matriz A (3×3) onde todos os elementos sejam 1 e uma matriz B onde todos os elementos sejam 5. Em seguida, adicione elementos à matriz A por anexação da matriz B , tal que a matriz A transforme-se em:

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

Operações Matemáticas com Arranjos

Uma vez que as variáveis do MATLAB tenham sido declaradas, podemos usá-las em uma ampla variedade de operações matemáticas. No Capítulo 1, as variáveis escalares foram usadas em operações matemáticas triviais, isto é, onde arranjos 1×1 eram manipulados por meio de operações matemáticas de modo que era possível interpretá-las como operações envolvendo números ordinários. No Capítulo 2, vimos que os arranjos não precisam ser necessariamente escalares ou vetoriais (arranjos com um elemento, uma linha ou uma coluna), eles podem ser do tipo matricial ou bidimensionais (arranjos em linhas e colunas) e até mesmo de dimensões maiores. Nesses casos, as operações matemáticas tornam-se um pouco mais complexas. Como sugere o nome do *software*, o MATLAB foi desenvolvido para realizar operações matemáticas avançadas, e isso inclui arranjos bastante genéricos, que aparecem frequentemente em muitas aplicações nas engenharias e em outros campos das ciências exatas. Este capítulo apresenta as operações matemáticas básicas que o MATLAB realiza manipulando arranjos.

As operações de adição e subtração são relativamente simples e, por isso, são descritas na Seção 3.1. As demais operações básicas (multiplicação, divisão e exponenciação) podem ser realizadas no MATLAB de dois modos diferentes. O modo mais elementar utiliza os símbolos padronizados (*, / e ^) e segue as regras operacionais da álgebra linear. Essas operações são apresentadas nas Seções 3.2 e 3.3. O segundo modo, que realiza operações escalares entre os elementos de dois ou mais arranjos, está descrito na Seção 3.4. Essas operações usam os símbolos .*, ./ e .^ (um ponto deve ser digitado antes do símbolo padrão da operação). Para os dois modos de operação, o MATLAB possui ainda os operadores de divisão à esquerda (\ ou \), explicados nas Seções 3.3 e 3.4.

Observação para iniciantes do MATLAB:

Embora as operações envolvendo matrizes sejam apresentadas nas primeiras seções deste capítulo e as operações escalares envolvendo elementos de dois ou mais arranjos apareçam logo em seguida, essa ordem pode ser comutada, visto que as duas operações são completamente independentes. É esperado que os usuários tenham algum conhecimento de operações com matrizes e de álgebra linear, sendo capazes de assimilar, sem dificuldade, o conteúdo das Seções 3.2 e 3.3. Entretanto, alguns leitores podem desejar ler primeiramente a Seção 3.4. O MATLAB pode ser utilizado em operações escalares entre arranjos e, de fato, são numerosas as aplicações que não requerem as operações de multiplicação ou divisão da álgebra linear.

3.1 ADIÇÃO E SUBTRAÇÃO DE ARRANJOS

As operações de adição (+) e subtração (−) são efetuadas apenas com matrizes de mesmo tamanho ou mesma dimensão. A soma ou a diferença entre duas matrizes é obtida adicionando-se ou subtraindo-se, respectivamente, elementos em posições correspondentes nas matrizes.

Em geral, se A e B são duas matrizes, por exemplo, matrizes 2×3 ,

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \end{bmatrix} \quad \text{e} \quad B = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \end{bmatrix}$$

então a matriz resultante da adição $A + B$ é:

$$\begin{bmatrix} (A_{11} + B_{11}) & (A_{12} + B_{12}) & (A_{13} + B_{13}) \\ (A_{21} + B_{21}) & (A_{22} + B_{22}) & (A_{23} + B_{23}) \end{bmatrix}$$

Quando um escalar (número) é adicionado ou subtraído a uma matriz, cada elemento da matriz é adicionado ou subtraído desse número. Exemplos:

```

>> VetA = [8 5 4]; VetB = [10 2 7];
>> VetC = VetA + VetB
VetC =
    18    7    11
>> A = [5 -3 8; 9 2 10]
A =
     5     -3     8
     9      2    10
>> B = [10 7 4; -11 15 1]
B =
    10     7     4
   -11    15     1
>> A - B

```

Declara dois vetores.

Define o vetor VetC igual a soma VetA + VetB.

Declara duas matrizes 2×3 , A e B.

Subtrai a matriz B da matriz A.

```
ans =
```

```
    -5   -10    4
```

```
    20   -13    9
```

```
>> C = A + B
```

```
C =
```

```
    15    4   12
```

```
    -2   17   11
```

```
>> C - 8
```

```
ans =
```

```
     7    -4    4
```

```
   -10     9    3
```

```
>>
```

Define uma matriz C igual a A + B.

O escalar 8 é subtraído de todos os elementos da matriz C.

3.2 MULTIPLICAÇÃO DE ARRANJOS

A operação de multiplicação ($*$) é executada pelo MATLAB de acordo com as regras da álgebra linear. Significa que, se A e B são duas matrizes, a operação $A*B$ tem sentido se, e somente se, o número de colunas da matriz A for igual ao número de linhas da matriz B . O resultado é uma matriz que possui o mesmo número de linhas de A e o mesmo número de colunas de B . Por exemplo, se A for uma matriz 4×3 e B uma matriz 3×2 :

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \\ A_{41} & A_{42} & A_{43} \end{bmatrix} \quad \text{e} \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \\ B_{31} & B_{32} \end{bmatrix}$$

então, a matriz obtida da operação $A*B$ possui dimensão 4×2 e os elementos são:

$$\begin{bmatrix} (A_{11}B_{11} + A_{12}B_{21} + A_{13}B_{31}) & (A_{11}B_{12} + A_{12}B_{22} + A_{13}B_{32}) \\ (A_{21}B_{11} + A_{22}B_{21} + A_{23}B_{31}) & (A_{21}B_{12} + A_{22}B_{22} + A_{23}B_{32}) \\ (A_{31}B_{11} + A_{32}B_{21} + A_{33}B_{31}) & (A_{31}B_{12} + A_{32}B_{22} + A_{33}B_{32}) \\ (A_{41}B_{11} + A_{42}B_{21} + A_{43}B_{31}) & (A_{41}B_{12} + A_{42}B_{22} + A_{43}B_{32}) \end{bmatrix}$$

Um exemplo numérico:

$$\begin{bmatrix} 1 & 4 & 3 \\ 2 & 6 & 1 \\ 5 & 2 & 8 \end{bmatrix} \begin{bmatrix} 5 & 4 \\ 1 & 3 \\ 2 & 6 \end{bmatrix} = \begin{bmatrix} (1 \cdot 5 + 4 \cdot 1 + 3 \cdot 2) & (1 \cdot 4 + 4 \cdot 3 + 3 \cdot 6) \\ (2 \cdot 5 + 6 \cdot 1 + 1 \cdot 2) & (2 \cdot 4 + 6 \cdot 3 + 1 \cdot 6) \\ (5 \cdot 5 + 2 \cdot 1 + 8 \cdot 2) & (5 \cdot 4 + 2 \cdot 3 + 8 \cdot 6) \end{bmatrix} = \begin{bmatrix} 15 & 34 \\ 18 & 32 \\ 43 & 74 \end{bmatrix}$$

O resultado da multiplicação de duas matrizes quadradas de mesmo tamanho é outra matriz quadrada do mesmo tamanho (dimensão) das matrizes originais. Contudo, a multiplicação de matrizes não obedece à propriedade comutativa. Significa que se A e B são matrizes $n \times n$, em geral, $A*B \neq B*A$. Além disso, a operação de potenciação somente pode ser realizada com matrizes quadradas (porque a operação $A*A$ só pode ser efetuada se o número de colunas da primeira matriz for igual ao número de linhas da segunda).

Dois vetores podem ser multiplicados um pelo outro somente se possuírem o mesmo número de elementos e se um dos vetores for um arranjo linha e o outro vetor um arranjo coluna. O resultado da multiplicação de um vetor linha por um vetor coluna é um arranjo 1×1 , ou seja, um escalar. Esse é o produto escalar usual de dois vetores. (O MATLAB possui uma função nativa chamada `dot(a, b)` que calcula o produto escalar de dois vetores.) Quando a função `dot` é utilizada, os vetores `a` e `b` podem ser tanto vetores linha como vetores coluna (veja a Tabela 3-1). A multiplicação de um vetor coluna por um vetor linha (ambos com n elementos) resulta numa matriz $n \times n$.

TUTORIAL 3-1 Multiplicação de arranjos

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

```
A =
```

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

Declara uma matriz A de dimensão 4×3 .

```
>> B = [6 1; 2 5; 7 3]
```

```
B =
```

```
6 1
2 5
7 3
```

Declara uma matriz B de dimensão 3×2 .

```
>> C = A*B
```

```
C =
```

```
28 27
65 49
98 32
84 38
```

Define a matriz C como o resultado da multiplicação das matrizes A e B.

```
>> D = B*A
```

```
??? Error using ==> *
```

```
Inner matrix dimensions must agree.
```

A tentativa de multiplicar B por A ($B \cdot A$) conduz ao erro, visto que o número de colunas de B é 2 e o número de linhas de A é 4.

```
>> F = [1 3; 5 7]
```

```
F =
```

```
1 3
5 7
```

```
>> G = [4 2; 1 6]
```

```
G =
```

Declara duas matrizes de dimensão 2×2 .

TUTORIAL 3-1 Multiplicação de arranjos

(continuação)

```
4 2
```

```
1 6
```

```
>> F*G
```

Produto $F \cdot G$.

```
ans =
```

```
7 20
```

```
27 52
```

```
>> G*F
```

Produto $G \cdot F$.

```
ans =
```

```
14 26
```

```
31 45
```

Observe que o produto $G \cdot F$ não é igual ao produto $F \cdot G$.

```
>> AV = [2 5 1]
```

```
AV =
```

```
2 5 1
```

Declara um vetor linha AV com três elementos.

```
>> BV = [3; 1; 4]
```

```
BV =
```

```
3
```

```
1
```

```
4
```

Declara um vetor coluna BV com três elementos.

```
>> AV*BV
```

```
ans =
```

```
15
```

Produto AV por BV. A resposta é um escalar.
(Produto escalar de dois vetores).

```
>> BV*AV
```

```
ans =
```

```
6 15 3
```

```
2 5 1
```

```
8 20 4
```

Produto BV por AV. A resposta é uma matriz de dimensão 3×3 .

```
>>
```

Quando um arranjo é multiplicado por um número, cada elemento do arranjo é multiplicado por esse número (lembre-se que um número é um arranjo 1×1). Exemplo:

```
>> A = [2 5 7 0; 10 1 3 4; 6 2 11 5]
```

```
A =
```

```
2 5 7 0
```

```
10 1 3 4
```

```
6 2 11 5
```

Declara uma matriz A de dimensão 3×4 .

```
>> b = 3
```

```
b =
```

Atribui o número 3 à variável b.

```

3
>> b*A
ans =
    6    15    21    0
   30    3     9    12
   18    6    33    15
>> C=A*5
C =
   10    25    35    0
   50     5    15    20
   30    10    55    25
>>

```

Multiplica a matriz A por b. Isso pode ser feito digitando-se $b \cdot A$ ou $A \cdot b$.

Define a matriz C como o produto da matriz A pelo escalar 5.

As regras para a multiplicação de arranjos da álgebra linear fornecem um modo bastante conveniente de escrever um sistema de equações lineares. Por exemplo, o sistema de três equações a três incógnitas:

$$A_{11}x_1 + A_{12}x_2 + A_{13}x_3 = B_1$$

$$A_{21}x_1 + A_{22}x_2 + A_{23}x_3 = B_2$$

$$A_{31}x_1 + A_{32}x_2 + A_{33}x_3 = B_3$$

pode ser escrito na forma matricial:

$$\begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix}$$

e na notação matricial:

$$AX = B \quad \text{onde} \quad A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}, \quad X = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \text{e} \quad B = \begin{bmatrix} B_1 \\ B_2 \\ B_3 \end{bmatrix}$$

3.3 DIVISÃO DE ARRANJOS

A divisão de arranjos também está condicionada às regras da álgebra linear. Essa operação é ainda mais complexa e, por isso, será apresentada apenas uma breve explanação sobre o assunto, a seguir. Explicações mais completas podem ser encontradas em livros específicos sobre álgebra linear.

O único modo de explicar a operação de divisão de arranjos é por meio da matriz identidade e da matriz inversa.

Matriz identidade:

A matriz identidade é uma matriz quadrada cujos elementos da diagonal principal são unitários (1s) e os demais elementos são nulos (0s). Conforme foi visto na Seção 2.2.1, uma matriz identidade pode ser criada no MATLAB acionando-se o comando `eye`. Quando uma matriz ou um vetor é multiplicado pela matriz identidade, o resultado é a própria matriz ou vetor (a multiplicação deve ser feita de acordo com as regras da álgebra linear). Isto é o equivalente matricial à multiplicação de um escalar por 1. Exemplos:

$$\begin{bmatrix} 7 & 3 & 8 \\ 4 & 11 & 5 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 7 & 3 & 8 \\ 4 & 11 & 5 \end{bmatrix} \text{ ou } \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 8 \\ 2 \\ 15 \end{bmatrix} = \begin{bmatrix} 8 \\ 2 \\ 15 \end{bmatrix} \text{ ou } \begin{bmatrix} 6 & 2 & 9 \\ 1 & 8 & 3 \\ 7 & 4 & 5 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 6 & 2 & 9 \\ 1 & 8 & 3 \\ 7 & 4 & 5 \end{bmatrix}$$

Se a matriz A for quadrada, não importa a ordem de multiplicação pela matriz identidade I , isto é, à esquerda ou à direita de A :

$$AI = IA = A$$

Inversa de uma matriz:

Uma matriz B é a matriz inversa da matriz A se o produto dessas duas matrizes é a matriz identidade (supondo que podemos multiplicar as duas matrizes). Ambas as matrizes devem ser quadradas e a multiplicação deve comutar, isto é, a ordem BA ou AB não é importante.

$$BA = AB = I$$

Obviamente, se B é a inversa de A , naturalmente A será a inversa de B . Por exemplo:

$$\begin{bmatrix} 2 & 1 & 4 \\ 4 & 1 & 8 \\ 2 & -1 & 3 \end{bmatrix} \begin{bmatrix} 5.5 & -3.5 & 2 \\ 2 & -1 & 0 \\ -3 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 5.5 & -3.5 & 2 \\ 2 & -1 & 0 \\ -3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 & 4 \\ 4 & 1 & 8 \\ 2 & -1 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Tipicamente, a matriz inversa de A é simbolizada por A^{-1} . No MATLAB, a inversa de A pode ser obtida elevando A à potência -1 ou utilizando a função nativa `inv(A)`. Por exemplo, a multiplicação das matrizes acima no MATLAB pode ser observada no programa a seguir:

```
>> A = [2 1 4; 4 1 8; 2 -1 3]
```

```
A =
```

```
2 1 4
4 1 8
2 -1 3
```

```
>> B = inv(A)
```

```
B =
```

```
5.5000 -3.5000 2.0000
2.0000 -1.0000 0
```

Declarando a matriz A.

Usa a função `inv` para determinar a inversa de A e atribui o resultado à matriz B.


```
-3.0000  2.0000 -1.0000
```

```
>> A*B
```

```
ans =
```

```
1 0 0
```

```
0 1 0
```

```
0 0 1
```

```
>> A*A^-1
```

```
ans =
```

```
1 0 0
```

```
0 1 0
```

```
0 0 1
```

O produto das matrizes A e B resulta na matriz identidade.

Utiliza a potência -1 para determinar a inversa de A. Multiplicando-a pela matriz A também obtém-se a matriz identidade.

- Nem toda matriz possui uma inversa. Uma matriz tem inversa se ela é quadrada e possui determinante diferente de zero.

Determinantes:

Determinante é uma função associada às matrizes quadradas. Uma rápida revisão de determinantes é fornecida a seguir. Mais detalhes sobre os determinantes podem ser encontrados em livros específicos sobre álgebra linear.

O determinante é uma função que associa uma matriz quadrada A a um número, chamado determinante da matriz. Simbolicamente, o determinante de uma matriz A é escrito como $\det(A)$ ou $|A|$. O cálculo do determinante é feito segundo regras específicas. Para uma matriz 2×2 , a regra é:

$$A = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} = a_{11}a_{22} - a_{12}a_{21}, \text{ por exemplo, } \begin{vmatrix} 6 & 5 \\ 3 & 9 \end{vmatrix} = 6 \cdot 9 - 5 \cdot 3 = 39$$

O MATLAB possui uma função nativa, chamada `det`, que calcula o determinante de matrizes quadradas (veja a Tabela 3-1).

Divisão de arranjos:

O MATLAB possui dois tipos de operadores de divisão: o operador de divisão à direita (`/`) e o operador de divisão à esquerda (`\`).

Operador de divisão à esquerda `\`:

O operador de divisão à esquerda é usado, entre outras coisas, para resolver a equação matricial $AX = B$. Nessa equação, X e B são vetores coluna. Ela pode ser resolvida multiplicando-se (à esquerda) ambos os lados da equação pela inversa de A :

$$A^{-1}AX = A^{-1}B$$

O lado esquerdo da equação é X , pois:

$$A^{-1}AX = IX = X$$

Assim, a solução da equação $AX = B$ é:

$$X = A^{-1}B$$

No MATLAB, a última equação pode ser reescrita usando o operador de divisão à esquerda:

$$X = A \setminus B$$

Deve ser ressaltado, entretanto, que embora as duas últimas equações produzam o mesmo resultado, o método segundo o qual o MATLAB determina X é diferente do apresentado. Na primeira equação, o MATLAB encontra a inversa de A (isto é, A^{-1}) e a utiliza na multiplicação por B . Na segunda (divisão à esquerda), a solução para X é obtida numericamente por meio do método de eliminação de Gauss. O método de eliminação de Gauss é mais recomendado na resolução de um conjunto de equações lineares, porque o cálculo da inversa pode exibir uma precisão menor que o método de Gauss quando o sistema de equações é formado por matrizes de dimensão muito elevada.

Operador de divisão à direita /:

O operador de divisão à direita é usado para resolver a equação matricial $XC = D$. Nessa equação, X e D são vetores linha. A equação pode ser resolvida multiplicando-se os dois lados da igualdade pela inversa de C :

$$X \cdot CC^{-1} = D \cdot C^{-1}$$

resultando em:

$$X = D \cdot C^{-1}$$

No MATLAB, a última equação pode ser reescrita usando-se o operador de divisão à direita:

$$X = D / C$$

O exemplo a seguir ilustra o modo de utilização dos operadores de divisão à esquerda e à direita, além de mostrar como a função nativa `inv` pode ser utilizada para resolver um sistema de equações lineares.

Problema-exemplo 3-1: Resolvendo um sistema linear com três equações (divisão de arranjo)

Use operações matriciais para resolver o seguinte sistema de equações lineares.

$$\begin{aligned} 4x - 2y + 6z &= 8 \\ 2x + 8y + 2z &= 4 \\ 6x + 10y + 3z &= 0 \end{aligned}$$

Solução

Utilizando as regras da álgebra linear mostradas anteriormente, o sistema de equações acima pode ser reescrito na forma matricial $AX = B$ ou $XC = D$:

$$\begin{bmatrix} 4 & -2 & 6 \\ 2 & 8 & 2 \\ 6 & 10 & 3 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 8 \\ 4 \\ 0 \end{bmatrix} \quad \text{ou} \quad \begin{bmatrix} x & y & z \end{bmatrix} \begin{bmatrix} 4 & 2 & 6 \\ -2 & 8 & 10 \\ 6 & 2 & 3 \end{bmatrix} = \begin{bmatrix} 8 & 4 & 0 \end{bmatrix}$$

A seguir, exibimos ambas as soluções:

```
>> A = [4 -2 6; 2 8 2; 6 10 3];
>> B = [8; 4; 0];
>> X = A \ B
X =
    -1.8049
     0.2927
     2.6341
>> Xb = inv(A) * B
Xb =
    -1.8049
     0.2927
     2.6341
>> C = [4 2 6; -2 8 10; 6 2 3];
>> D = [8 4 0];
>> Xc = D / C
Xc =
    -1.8049    0.2927    2.6341
>> Xd = D * inv(C)
Xd =
    -1.8049    0.2927    2.6341
```

Resolvendo a forma $AX = B$.

Resolvendo por meio da divisão à esquerda $X = A \setminus B$.

Resolvendo por meio da inversa de A: $X = A^{-1}B$.

Resolvendo a forma $XC = D$.

Resolvendo por meio da divisão à direita $X = D / C$.

Resolvendo por meio da inversa de C: $X = D \cdot C^{-1}$.

3.4 OPERAÇÕES ESCALARES ENVOLVENDO ELEMENTOS DE MATRIZES (OPERAÇÕES ELEMENTO POR ELEMENTO)

Nas Seções 3.2 e 3.3 vimos que, quando os símbolos de multiplicação e divisão ordinários ($*$ e $/$) são utilizados em arranjos, as operações matemáticas seguem rigorosamente as regras da álgebra linear. Há, porém, muitas situações que requerem operações escalares envolvendo elementos correspondentes em dois ou mais arranjos (vetores ou matrizes). Essas operações são levadas a cabo em cada elemento dos arranjos. A adição e a subtração são, por definição, operações elemento por elemento, visto que, quando dois arranjos são somados ou subtraídos, a operação é executada com os elementos que ocupam a mesma posição nos arranjos. Essas operações só podem ser realizadas com arranjos de mesma dimensão.

As operações escalares de multiplicação, divisão e exponenciação de matrizes, envolvendo elemento por elemento dos arranjos, são sinalizadas no MATLAB digitando-se um ponto antes do operador aritmético.

Símbolo	Descrição	Símbolo	Descrição
.*	Multiplicação escalar	./	Divisão escalar à direita
.^	Exponenciação escalar	.\	Divisão escalar à esquerda

Se dois vetores a e b são da forma $a = [a_1 \ a_2 \ a_3 \ a_4]$ e $b = [b_1 \ b_2 \ b_3 \ b_4]$, então a multiplicação, divisão e exponenciação escalar, envolvendo elementos correspondentes dos dois vetores, resultam em:

$$\begin{aligned} a.*b &= [a_1 b_1 \ a_2 b_2 \ a_3 b_3 \ a_4 b_4] \\ a./b &= [a_1 / b_1 \ a_2 / b_2 \ a_3 / b_3 \ a_4 / b_4] \\ a.^b &= [(a_1)^{b_1} \ (a_2)^{b_2} \ (a_3)^{b_3} \ (a_4)^{b_4}] \end{aligned}$$

Se duas matrizes A e B são:

$$A = \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix} \text{ e } B = \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{21} & B_{22} & B_{23} \\ B_{31} & B_{32} & B_{33} \end{bmatrix}$$

então a multiplicação e divisão escalar das duas matrizes, elemento por elemento, resultam em:

$$A.*B = \begin{bmatrix} A_{11}B_{11} & A_{12}B_{12} & A_{13}B_{13} \\ A_{21}B_{21} & A_{22}B_{22} & A_{23}B_{23} \\ A_{31}B_{31} & A_{32}B_{32} & A_{33}B_{33} \end{bmatrix} \quad A./B = \begin{bmatrix} A_{11}/B_{11} & A_{12}/B_{12} & A_{13}/B_{13} \\ A_{21}/B_{21} & A_{22}/B_{22} & A_{23}/B_{23} \\ A_{31}/B_{31} & A_{32}/B_{32} & A_{33}/B_{33} \end{bmatrix}$$

A exponenciação de cada elemento da matriz A resulta em:

$$A.^n = \begin{bmatrix} (A_{11})^n & (A_{12})^n & (A_{13})^n \\ (A_{21})^n & (A_{22})^n & (A_{23})^n \\ (A_{31})^n & (A_{32})^n & (A_{33})^n \end{bmatrix}$$

O Tutorial 3-2 demonstra o uso das operações escalares envolvendo elementos correspondentes de matrizes.

TUTORIAL 3-2 Operações escalares com matrizes (elemento por elemento)

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

```
A =
```

```
     2     6     3
```

```
     5     8     4
```

```
>> B = [1 4 10; 3 2 7]
```

```
B =
```

```
     1     4    10
```

```
     3     2     7
```

```
>> A.*B
```

Declara uma matriz A de dimensão 2 × 3

Declara uma matriz B de dimensão 2 × 3

TUTORIAL 3-2 Operações escalares com matrizes (elemento por elemento) *(continuação)*

```
ans =
    2    24    30
   15    16    28
>> C = A ./ B
C =
    2.0000    1.5000    0.3000
    1.6667    4.0000    0.5714
>> B.^3
ans =
    1    64   1000
   27    8    343
>> A * B
??? Error using ==> *
Inner matrix dimensions must agree.
>>
```

Divisão escalar das matrizes A e B. O resultado é atribuído à matriz C.

Exponenciação escalar dos elementos da matriz B. O resultado é um arranjo onde cada elemento corresponde a um elemento de B elevado à potência três.

A tentativa de realizar o produto $A \cdot B$ seguindo as regras da álgebra linear falha, pois o número de colunas da matriz A não é igual ao número de linhas da matriz B.

Cálculos envolvendo elementos correspondentes em arranjos são bastante úteis quando se quer testar múltiplos valores no argumento de uma função ao mesmo tempo. Primeiramente é declarado um vetor contendo os possíveis valores da variável independente e, depois, esse vetor é usado para gerar um vetor contendo os valores correspondentes da função a ser calculada. Exemplo:

```
>> x = [1:8]
x =
    1    2    3    4    5    6    7    8
>> y = x.^2-4*x
y =
   -3   -4   -3    0    5   12   21   32
>>
```

Cria um vetor x com oito elementos.

Vetor x é utilizado em operações escalares para gerar os elementos do novo vetor y.

No exemplo acima, $y = x^2 - 4x$. A operação envolvendo elemento por elemento do vetor x é necessária quando é calculado o quadrado de x. Cada elemento no vetor y possui um valor tal que seria obtido substituindo-se o elemento correspondente de x na equação de y. Outro exemplo:

```
>> z = [1:2:15]
z =
    1    3    5    7    9   10   11   13   15
```

Cria um vetor z com oito elementos.

```
>> y = (z.^3+5*z)./(4*z.^2-10)
```

```
y =
```

```
-1.0000 1.6154 1.6667 2.0323 2.4650 2.9241 3.3964 3.8764
```

```
>>
```

Vetor z é utilizado em operações escalares para gerar os elementos do novo vetor y .

Nesse exemplo, $y = \frac{z^3 + 5z}{4z^2 - 10}$. As operações escalares (elemento por elemento) são usadas três vezes no exemplo: para encontrar z^3 e z^2 e para dividir o numerador pelo denominador.

3.5 USANDO ARRANJOS EM FUNÇÕES NATIVAS DO MATLAB

As funções nativas do MATLAB são escritas de tal forma que, sendo o argumento (entrada) um arranjo, a operação é executada em cada elemento desse arranjo. (Podemos pensar em operações em cada elemento do arranjo sob aplicação da função específica.) O resultado (saída) da aplicação da função no arranjo é um novo arranjo, cujas entradas são formadas pela função agindo em cada elemento do arranjo original. Por exemplo, se um vetor x com sete elementos é o argumento da função `cos(x)`, o resultado da operação é um vetor com sete elementos cujas entradas são os co-senos das entradas do vetor x . Exemplo:

```
>> x = [0:pi/6:pi]
```

```
x =
```

```
0 0.5236 1.0472 1.5708 2.0944 2.6180 3.1416
```

```
>> y = cos(x)
```

```
y =
```

```
1.0000 0.8660 0.5000 0.0000 -0.5000 -0.8660 -1.0000
```

```
>>
```

Um exemplo envolvendo argumento matricial:

```
>> d = [1 4 9; 16 25 36; 49 64 81]
```

```
d =
```

```
1 4 9
```

```
16 25 36
```

```
49 64 81
```

```
>> h = sqrt(d)
```

```
h =
```

Criando uma matriz de dimensão 3×3 .

```

1 2 3
4 5 6
7 8 9
>>

```

h é uma matriz de dimensão 3×3 na qual cada elemento é a raiz quadrada do elemento correspondente na matriz d , introduzida como argumento de `sqrt`.

Essa característica do MATLAB segundo a qual arranjos são utilizados como argumentos de funções é denominada *vetorização*.

3.6 FUNÇÕES NATIVAS PARA CÁLCULOS DE ARRANJOS

O MATLAB está repleto de funções nativas para cálculos envolvendo arranjos. A Tabela 3-1 mostra algumas delas.

TABELA 3-1 Funções nativas para cálculos com arranjos

Função	Descrição	Exemplo
<code>mean(A)</code>	Se A é um vetor, retorna o valor médio dos elementos do vetor.	<pre>>> A = [5 9 2 4]; >> mean(A) ans = 5</pre>
<code>C=max(A)</code>	Se A é um vetor, C receberá o maior elemento de A . Se A é uma matriz, C é um vetor linha contendo o maior elemento em cada coluna de A .	<pre>>> A = [5 9 2 4 11 6 7 11 0 1]; >> C = max(A) C = 11</pre>
<code>[d,n]=max(A)</code>	Se A é um vetor, d recebe o maior elemento de A e n indica a posição desse elemento no vetor (a primeira posição, caso haja mais de um elemento de valor igual ao máximo).	<pre>>> [d,n] = max(A) d = 11 n = 5</pre>
<code>min(A)</code>	Semelhante à função <code>max(A)</code> , mas retorna o menor elemento de A .	<pre>>> A = [5 9 2 4] >> min(A) ans = 2</pre>
<code>[d,n]=min(A)</code>	Semelhante a <code>[d,n]=max(A)</code> para o menor elemento de A .	
<code>sum(A)</code>	Se A é um vetor, retorna a soma dos elementos do vetor.	<pre>>> A = [5 9 2 4]; >> sum(A) ans = 20</pre>

TABELA 3-1 Funções nativas para cálculos com arranjos

(continuação)

Função	Descrição	Exemplo
sort (A)	Se A é um vetor, ordena os elementos de A na ordem crescente.	<pre>>> A = [5 9 2 4]; >> sort(A) ans = 2 4 5 9</pre>
median (A)	Se A é um vetor, retorna o valor mediano dos elementos do vetor.	<pre>>> A = [5 9 2 4]; >> median(A) ans = 4.5000</pre>
std (A)	Se A é um vetor, retorna o desvio-padrão dos elementos do vetor.	<pre>>> A = [5 9 2 4]; >> std(A) ans = 2.9439</pre>
det (A)	Retorna o determinante da matriz quadrada A.	<pre>>> A = [2 4; 3 5]; >> det(A) ans = -2</pre>
dot (a, b)	Determina o produto escalar de dois vetores a e b. Os vetores podem ser tanto linha quanto coluna.	<pre>>> a = [1 2 3]; >> b = [3 4 5]; >> dot(a,b) ans = 26</pre>
cross (a, b)	Determina o produto vetorial de dois vetores a e b ($a \times b$). Os dois vetores devem possuir três elementos.	<pre>>> a = [1 3 2]; >> b = [2 4 1]; >> cross(a,b) ans = -5 3 -2</pre>
inv (A)	Retorna a matriz inversa da matriz quadrada A.	<pre>>> A = [2 -2 1; 3 2 -1; 2 -3 2]; >> inv(A) ans = 0.2000 0.2000 0 -1.6000 0.4000 1.0000 -2.6000 0.4000 2.0000</pre>

3.7 GERAÇÃO DE NÚMEROS ALEATÓRIOS

Simulações de muitos processos físicos e aplicações das engenharias requerem um número ou um conjunto de números aleatórios. O MATLAB possui dois comandos `rand` e `randn` que podem ser utilizados para atribuir números aleatórios às variáveis.

O comando rand:

O comando `rand` gera números distribuídos uniformemente com valores entre 0 e 1. Como mostra a Tabela 3-2, o comando pode ser usado para atribuir números a uma variável escalar, vetorial ou matricial.

TABELA 3-2 O comando `rand`

Comando	Descrição	Exemplo
<code>rand</code>	Gera um único número aleatório entre 0 e 1.	<pre>>> rand ans = 0.2311</pre>
<code>rand(1,n)</code>	Gera vetor linha com n elementos aleatórios entre 0 e 1.	<pre>>> a = rand(1,4) a = 0.6068 0.4860 0.8913 0.7621</pre>
<code>rand(n)</code>	Gera uma matriz $n \times n$ com números aleatórios situados entre 0 e 1.	<pre>>> b = rand(3) b = 0.4565 0.4447 0.9218 0.0185 0.6154 0.7382 0.8214 0.7919 0.1763</pre>
<code>rand(m,n)</code>	Gera uma matriz $m \times n$ com números aleatórios situados entre 0 e 1.	<pre>>> c = rand(2,4) c = 0.4057 0.9169 0.8936 0.3529 0.9355 0.4103 0.0579 0.8132</pre>
<code>randperm(n)</code>	Gera um vetor linha contendo n elementos que são a permutação aleatória dos inteiros de 1 até n.	<pre>>> randperm(8) ans = 8 2 7 4 3 6 5 1</pre>

Às vezes são necessários números aleatórios que estão distribuídos em um intervalo diferente de (0,1) ou números aleatórios possuindo apenas valores inteiros. Isso pode ser realizado no MATLAB aplicando-se operações matemáticas na função `rand`. Números aleatórios distribuídos em um intervalo arbitrário (a,b) podem ser obtidos multiplicando-se a função `rand` pelo tamanho do intervalo ($b-a$) e adicionando-se o produto à extremidade a :

$$(b-a)*\text{rand} + a$$

Por exemplo, um vetor com 10 elementos possuindo valores entre -5 e 10 pode ser criado impondo-se ($a = -5$; $b = 10$):

```
>> v = 15*rand(1,10)-5
v =
    7.3508   -4.3005    3.9687    9.2373   -0.6680    8.3325   -3.4762   -4.0203   -1.4855
    8.9965
```


Um arranjo de números aleatórios inteiros pode ser gerado por meio de uma das funções de arredondamento. Por exemplo, uma matriz 2×15 contendo apenas números inteiros com valores entre 1 e 100 pode ser criada por:

```
>> A = round(99*rand(2,15)+1)
```

```
A =
```

```
7 100 50 68 77 14 2 82 3 62 25 27 66 61 66
27 22 30 96 67 10 30 99 82 56 82 76 22 61 19
```

O comando randn:

O comando `randn` gera uma distribuição normalizada de números com média 0 e desvio-padrão 1. O comando pode ser usado para gerar variáveis escalares, vetoriais ou matriciais de modo equivalente ao comando `rand`. Por exemplo, uma matriz 3×4 pode ser criada desta forma:

```
>> d = randn(3,4)
```

```
d =
```

```
-0.4326 0.2877 1.1892 0.1746
-1.6656 -1.1465 -0.0376 -0.1867
0.1253 1.1909 -0.3273 0.7258
```

O valor médio e o desvio-padrão podem ser modificados por meio de operações matemáticas. Isso pode ser feito multiplicando-se o número gerado por `randn` pelo desvio-padrão pretendido e adicionando-se o resultado à média desejada. Por exemplo, um vetor com 10 números inteiros, média 50 e desvio-padrão 5 é gerado por:

```
>> v = round(5*randn(1,15)+50)
```

```
v =
```

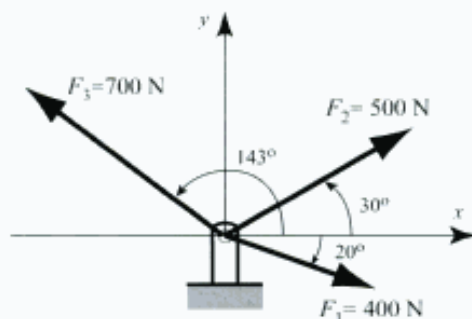
```
53 45 55 38 49 50 48 49 42 44 46 52 47 58 51
```

No exemplo acima, os números inteiros foram obtidos a partir da função `round`.

3.8 EXEMPLOS DE APLICAÇÕES DO MATLAB

Problema-exemplo 3-2: Adição de vetores (decomposição de forças)

Conforme mostra a figura, três forças estão aplicadas num suporte. Determine a força total (ou resultante) aplicada ao suporte.



Solução

Uma força é uma grandeza vetorial, ou seja, uma grandeza física que possui módulo, direção e sentido. No sistema de coordenadas cartesianas, um vetor bidimensional $\mathbf{F}$ pode ser escrito em suas componentes de acordo com:

$$\mathbf{F} = F_x \mathbf{i} + F_y \mathbf{j} = F \cos\theta \mathbf{i} + F \sin\theta \mathbf{j} = F (\cos\theta \mathbf{i} + \sin\theta \mathbf{j})$$

onde F é o módulo do vetor $\mathbf{F}$; θ é o ângulo medido relativamente ao eixo x ; F_x e F_y são as componentes de $\mathbf{F}$ nas direções x e y , respectivamente; e $\mathbf{i}$ e $\mathbf{j}$ são os vetores unitários nas direções x e y , respectivamente. Sendo conhecidas as componentes F_x e F_y , podemos determinar F e θ por:

$$F = \sqrt{F_x^2 + F_y^2} \text{ e } \tan\theta = \frac{F_y}{F_x}$$

A força total ou resultante aplicada ao suporte é obtida adicionando-se as forças individuais que agem sobre o suporte. A seguir, vejamos uma solução do problema em três passos no MATLAB:

- Escreva cada força como um vetor bidimensional, onde o primeiro elemento é a componente x e o segundo é a componente y .
- Adicione as respectivas componentes dos vetores para encontrar o vetor resultante.
- Determine o módulo e o sentido da força resultante sobre o suporte.

```
>> MF1=400; MF2=500; MF3=700;
```

Declara variáveis com o módulo de cada vetor.

```
>> teta1 = -20*pi/180; teta2 = 30*pi/180; teta3 = 143*pi/180;
```

Define as variáveis representando os ângulos de cada vetor.

```
>> F1 = MF1*[cos(teta1) sin(teta1)]
```

```
F1 =  
375.8770 -136.8081
```

```
>> F2=MF2*[cos(teta2) sin(teta2)]
```

```
F2 =  
433.0127 250.0000
```

Define os três vetores forças individuais.

```
>> F3=MF3*[cos(teta3) sin(teta3)]
```

```
F3 =  
-559.0449 421.2705
```

```
>> Ftot = F1 + F2 + F3
```

```
Ftot =  
249.8449 534.4625
```

Determina o vetor força total.

```
>> MFtot = sqrt(Ftot(1)^2 + Ftot(2)^2)
```

```
MFtot =
```

Calcula o módulo do vetor força total.

```

589.9768
>> teta = (180/pi)*atan(Ftot(2)/Ftot(1))
teta =
64.9453

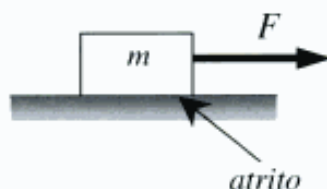
```

Determina o ângulo θ (em graus) do vetor força total.

A força resultante tem módulo 589.98N e ângulo 64.95° (sentido trigonométrico). Escrito na notação vetorial, o vetor força é: $\mathbf{F} = (249.84\mathbf{i} + 534.46\mathbf{j})$ N.

Problema-exemplo 3-3: Experimento para a determinação do coeficiente de atrito cinético

O coeficiente de atrito cinético (μ) pode ser determinado experimentalmente medindo-se o módulo da força necessário para mover uma massa m sobre uma superfície com atrito. Quando F é medida e sendo conhecidos os valores de m , o coeficiente de atrito cinético pode ser determinado por:



$$\mu = F / (mg) \quad (g = 9.81 \text{ m/s}^2)$$

Um conjunto de seis medidas é mostrado na tabela abaixo. Determine o coeficiente de atrito por medida e a respectiva média no experimento.

Medida	1	2	3	4	5	6
Massa m (kg)	2	4	5	10	20	50
Força F (N)	12.5	23.5	30	61	117	294

Solução

Uma solução no MATLAB (na janela Command Window) é mostrada a seguir:

```

>> m = [2 4 5 10 20 50];
>> F = [12.5 23.5 30 61 117 294];
>> mi = F./(m*9.81)
mi =
    0.6371    0.5989    0.6116    0.6218    0.5963    0.5994
>> mi_medio=mean(mi)
mi_medio =
0.6109

```

Entra com os valores de m em um vetor.

Entra com os valores de F em um vetor.

Usando operações elemento por elemento é determinado um valor de μ para cada medida separada.

O valor do coeficiente de atrito cinético é determinado a partir do cálculo da média dos elementos do vetor mi .

Problema-exemplo 3-4: Análise de malhas resistivas* (resolvendo um sistema de equações lineares)

A figura ao lado mostra um circuito elétrico composto de resistores e fontes de tensão. Determine a corrente em cada resistor usando a lei de Kirchhoff para tensões e o método das correntes de malha.

$$\begin{aligned} V_1 &= 20\text{V}, V_2 = 12\text{V}, V_3 = 40\text{V} \\ R_1 &= 18\Omega, R_2 = 10\Omega, R_3 = 16\Omega \\ R_4 &= 6\Omega, R_5 = 15\Omega, R_6 = 8\Omega \\ R_7 &= 12\Omega, R_8 = 14\Omega \end{aligned}$$

Solução

A lei de Kirchhoff para as tensões afirma que a soma das tensões (quedas e elevações) ao longo de um circuito ou malha fechada é zero. No método das correntes de malha, são arbitrados sentidos para as correntes de malha, na figura:

i_1, i_2, i_3 e i_4 no sentido horário. Em seguida, a lei de Kirchhoff para as tensões é aplicada em cada malha. Disso, resulta um sistema de quatro equações linearmente independentes, com as correntes de malhas sendo as incógnitas. A corrente em um resistor pertencente a duas malhas distintas é a soma algébrica com sinal das correntes em cada malha. É conveniente arbitrar todas as correntes no mesmo sentido. Na equação para cada malha, uma fonte de tensão entra com sinal positivo se a corrente de malha chegar primeiro ao pólo negativo; e a queda de tensão em cada resistor entra com sinal negativo na equação da malha se a corrente do resistor apontar na direção da corrente arbitrada.

As equações para as quatro malhas do problema são:

$$\begin{aligned} V_1 - R_1 i_1 - R_3 (i_1 - i_3) - R_2 (i_1 - i_2) &= 0 \\ -R_5 i_2 - R_2 (i_2 - i_1) - R_4 (i_2 - i_3) - R_7 (i_2 - i_4) &= 0 \\ -V_2 - R_6 (i_3 - i_4) - R_4 (i_3 - i_2) - R_3 (i_3 - i_1) &= 0 \\ V_3 - R_8 i_4 - R_7 (i_4 - i_2) - R_6 (i_4 - i_3) &= 0 \end{aligned}$$

As quatro equações pode ser reescritas na forma matricial padrão $[A][x] = [B]$:

$$\begin{bmatrix} -(R_1 + R_2 + R_3) & R_2 & R_3 & 0 \\ R_2 & -(R_2 + R_4 + R_5 + R_7) & R_4 & R_7 \\ R_3 & R_4 & -(R_3 + R_4 + R_6) & R_6 \\ 0 & R_7 & R_6 & -(R_6 + R_7 + R_8) \end{bmatrix} \begin{bmatrix} i_1 \\ i_2 \\ i_3 \\ i_4 \end{bmatrix} = \begin{bmatrix} -V_1 \\ 0 \\ V_2 \\ -V_3 \end{bmatrix}$$

* N. de T.: Resistências são elementos ôhmicos (lineares) que asseguram a linearidade do sistema de equações no circuito.

Uma solução deste sistema de equações no MATLAB é apresentada a seguir:

```
>> V1 = 20; V2 = 12; V3 = 40;
>> R1 = 18; R2 = 10; R3 = 16; R4 = 6;
>> R5 = 15; R6 = 8; R7 = 12; R8 = 14;
>> A = [-(R1 + R2 + R3) R2 R3 0
        R2 -(R2 + R4 + R5 + R7) R4 R7
        R3 R4 -(R3 + R4 + R6) R6
        0 R7 R6 -(R6 + R7 + R8)]
A =
    -44    10    16     0
     10   -43     6     12
     16     6   -30     8
     0    12     8   -34
>> B = [-V1; 0; V2; -V3]
B =
    -20
     0
     12
    -40
>> I = A \ B
```

A =

```

    -44    10    16     0
     10   -43     6     12
     16     6   -30     8
     0    12     8   -34
```

>> B = [-V1; 0; V2; -V3]

B =

```

    -20
     0
     12
    -40
```

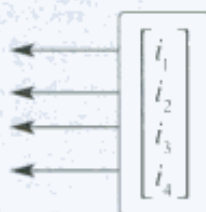
>> I = A \ B

I =

```

    0.8411
    0.7206
    0.6127
    1.5750
```

>>



Declara todas as variáveis para os Vs e os Rs do circuito.

Cria a matriz A.

Valor numérico da matriz A.

Cria o vetor B.

Resolve o sistema para as correntes usando divisão à esquerda.

O vetor coluna I possui como entradas as correntes em cada malha. As correntes nos resistores R_1 , R_5 e R_8 são $i_1 = 0.8411\text{ A}$, $i_2 = 0.7206\text{ A}$ e $i_4 = 1.5750\text{ A}$, respectivamente. Os demais resistores pertencem simultaneamente a duas malhas diferentes e a corrente neles é a soma algébrica com sinal das correntes em cada malha.

A corrente no resistor R_2 é $i_1 - i_2 = 0.1205\text{ A}$.

A corrente no resistor R_3 é $i_1 - i_3 = 0.2284\text{ A}$.

A corrente no resistor R_4 é $i_2 - i_3 = 0.1079\text{ A}$.

A corrente no resistor R_6 é $i_4 - i_3 = 0.9623\text{ A}$.

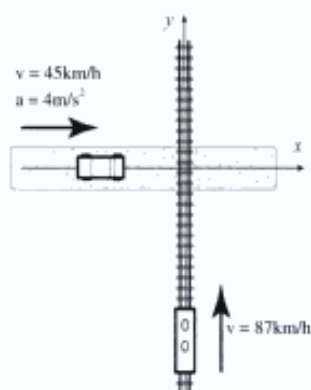
A corrente no resistor R_7 é $i_4 - i_2 = 0.8544\text{ A}$.

Problema-exemplo 3-5: Movimento de duas partículas

Um trem e um carro aproximam-se de uma passagem de nível. Em $t = 0$, o trem está 122m ao sul da passagem, viajando no sentido norte, com velocidade constante de

87km/h; o carro está 61m a leste da passagem, viajando no sentido oeste com velocidade de 45km/h e aceleração de 4m/s^2 . Determine as posições do trem e do carro; a distância e as velocidades relativas em cada segundo até o tempo $t = 10\text{s}$.

Os resultados devem ser exibidos em uma matriz 11×6 tal que a primeira coluna represente os intervalos de tempo de interesse. A posição do trem e do carro, a distância relativa carro-trem, a velocidade do carro e a velocidade do trem, em relação ao carro, são outras colunas da tabela, nessa ordem.



Solução

A posição de um objeto que se move ao longo de uma linha reta com aceleração constante é dada por $s = s_0 + v_0 t + \frac{1}{2} a t^2$, onde s_0 e v_0 são a posição e a velocidade no tempo $t = 0$ e a é a aceleração. Aplicando-se essa equação ao movimento do trem e do carro, obtemos:

$$y = -122 + v_{0\text{trem}} t \quad (\text{trem})$$

$$x = -61 + v_{0\text{carro}} t + \frac{1}{2} a_{\text{carro}} t^2 \quad (\text{carro})$$

A distância relativa entre o carro e o trem é: $d = \sqrt{x^2 + y^2}$.

Definindo convenientemente as coordenadas do sistema (veja a figura), a velocidade constante do trem pode ser escrita em notação vetorial como: $\mathbf{v}_{\text{trem}} = v_{0\text{trem}} \mathbf{j}$. O movimento do carro é uniformemente acelerado e a equação para a velocidade em um tempo t é: $\mathbf{v}_{\text{carro}} = (v_{0\text{carro}} + a_{\text{carro}} t) \mathbf{i}$. A velocidade do trem relativamente ao carro ($\mathbf{v}_{\text{t/c}}$) é dada por: $\mathbf{v}_{\text{t/c}} = \mathbf{v}_{\text{trem}} - \mathbf{v}_{\text{carro}} = -(v_{0\text{carro}} + a_{\text{carro}} t) \mathbf{i} + v_{0\text{trem}} \mathbf{j}$. A velocidade desejada é o módulo desse vetor.

Para resolver o problema, primeiro será criado o vetor $\mathbf{t}$ com 11 elementos, correspondentes aos intervalos de tempo de 0 a 10s. Em seguida serão calculadas as posições do trem e do carro, a distância entre eles e a velocidade do trem relativamente ao carro, para cada intervalo de tempo do vetor $\mathbf{t}$. Uma solução desse problema no MATLAB é mostrada a seguir:

```
>> v0trem = 87*1000/3600; v0carro = 45*1000/3600; acarro = 4;
```

Declara as velocidades iniciais (em m/s) e a aceleração.

```
>> t = 0:10;
```

Declara o vetor t.

```
>> y = -122 + v0trem*t;
```

```
>> x = -61 + v0carro*t + 0.5*acarro*t.^2;
```

Determina as posições instantâneas do trem e o do carro.

```
>> d = sqrt(x.^2 + y.^2);
```

Determina a distância relativa trem-carro.

```
>> vcarro = v0carro + acarro*t;
```

Determina a velocidade do carro.

```
>> vt_c = sqrt(vcarro.^2 + v0trem.^2);
```

Determina a velocidade do trem em relação ao carro.

```
>> Tabela=[t' y' x' d' vt_c']
```

Cria uma tabela com os dados solicitados.

```
Tabela =
```

0	-122.0000	-61.0000	136.4001	27.2080
1.0000	-97.8333	-46.5000	108.3218	29.2622
2.0000	-73.6667	-28.0000	78.8085	31.6903
3.0000	-49.5000	-5.5000	49.8046	34.4133
4.0000	-25.3333	21.0000	32.9056	37.3668
5.0000	-1.1667	51.5000	51.5132	40.5003
6.0000	23.0000	86.0000	89.0225	43.7753
7.0000	47.1667	124.5000	133.1351	47.1622
8.0000	71.3333	167.0000	181.5969	50.6387
9.0000	95.5000	213.5000	233.8857	54.1874
10.0000	119.6667	264.0000	289.8553	57.7951

```
>>
```

Tempo (s)	Posição trem (m)	Posição carro (m)	Distância carro_trem (m)	Velocidade trem_carro (m/s)
--------------	------------------------	-------------------------	--------------------------------	-----------------------------------

Observação: nos comandos acima, *tabela* é o nome da variável matricial que armazena os dados a serem exibidos.

Neste problema, os resultados numéricos foram exibidos pelo MATLAB sem qualquer texto de referência. Instruções sobre como utilizar caixas de textos de saídas geradas pelo MATLAB serão apresentadas no Capítulo 4.

3.9 PROBLEMAS

Nota: Os problemas propostos a seguir podem ser resolvidos escrevendo-se os comandos na janela Command Window. Entretanto, pode ser mais conveniente usar rotinas criadas pelo usuário no MATLAB (*script files*) para resolvê-los. As rotinas são os objetos de estudo do próximo capítulo. É aconselhável estudar o Capítulo 4 (ao menos as três primeiras seções) e, então, resolver os problemas deste capítulo usando esse recurso. Nos problemas adicionais do Capítulo 4, é possível praticar, efetivamente, as operações matemáticas envolvendo arranjos.

1. Dois vetores são definidos como:

$$\mathbf{u} = 4\mathbf{i} + 9\mathbf{j} - 5\mathbf{k} \quad \text{e} \quad \mathbf{v} = -3\mathbf{i} + 6\mathbf{j} - 7\mathbf{k}$$

Use o MATLAB para calcular o produto escalar $\mathbf{u} \cdot \mathbf{v}$ dos vetores de dois modos:

- a) Declare $\mathbf{u}$ como um vetor linha e $\mathbf{v}$ com um vetor coluna e, então, proceda à multiplicação dos arranjos.
- b) Use a função `dot`.

2. Considere a função $y = (x^2 + 1)^3 x^3$. Determine o valor de y para os seguintes valores de x : -2.5 -2 -1.5 -1 -0.5 0 0.5 1 1.5 2 2.5 3. Declare inicialmente o vetor x . Em seguida, resolva o problema para o vetor y , usando operações escalares (elemento por elemento).

3. A profundidade de um poço (d), em metros, pode ser determinada deixando-se cair uma pedra dentro dele (com velocidade inicial zero) e aguardando-se até que ela atinja o fundo do mesmo. Isso ocorre após uma distância:

$$d = \frac{1}{2} g t^2, \text{ onde } t \text{ é o tempo de queda livre da pedra em segundos e } g = 9,8 \text{ m/s}^2.$$

Determine d para $t = 1, 2, 3, 4, 5, 6, 7, 8, 9$ e 10 s.

(Sugestão: Crie o vetor t e determine d usando operações escalares – elemento por elemento.)

4. Declare x e y os vetores: $x = 2, 4, 6, 8, 10$ e $y = 3, 6, 9, 12, 15$. Em seguida, use-os na seguinte expressão para a variável z .

$$z = \frac{xy + \frac{y}{x}}{(x+y)^{(y-x)}} + 12^{x/y}$$

5. Declare h e k dois escalares: $h = 0.9$ e $k = 12.5$. Então, declare os vetores x , y e z tal que $x = 1, 2, 3$ e 4 ; $y = 0.9, 0.8, 0.7$ e 0.6 e $z = 2.5, 3, 3.5$ e 4 . Em seguida, use-os para calcular T na expressão a seguir:

$$T = \frac{xyz}{(h+k)^{x/5}} + \frac{ke^{\left(\frac{z}{x+y}\right)}}{z^h}$$

6. Mostre que $\lim_{n \rightarrow \infty} \left(1 + \frac{1}{n}\right)^n = e$.

Para tanto, crie um vetor n cujos elementos sejam, por exemplo: 1 10 100 500 1000 2000 4000 e 8000. Então, declare um novo vetor y cujos elementos são

determinados a partir dos elementos de n por meio de $\left(1 + \frac{1}{n}\right)^n$.

Por último, compare os elementos de y com o valor de e (digite `exp(1)` no *prompt* do MATLAB).

7. Use o MATLAB para mostrar que a soma da série infinita $\sum_{n=1}^{\infty} \frac{1}{n^2}$ converge a $\pi^2/6$. Faça-o calculando a soma para:

- a) $n = 100$
b) $n = 1,000$
c) $n = 10,000$

Em cada caso, crie um vetor n onde o primeiro elemento é 1, o incremento também vale 1 e o último termo é 100, 1,000 ou 10,000. Então, realizando operações escalares (elemento por elemento), crie um vetor cujos elementos são $1/n^2$. Finalmente, use a função `sum` para somar os termos da série. Compare os resultados das letras a , b e c com o valor de $\frac{\pi^2}{6}$. (Digite ponto-e-vírgula no final de cada comando para evitar a exibição aparentemente interminável dos valores na tela.)

8. Use o MATLAB para mostrar que a série infinita $\sum_{n=0}^{\infty} \frac{1}{(2n+1)(2n+2)}$ converge a $\ln 2$. Faça-o calculando a soma para:
- $n = 50$
 - $n = 500$
 - $n = 5,000$

Em cada caso, crie um vetor n onde o primeiro elemento é 0, o incremento é 1 e o últimos termos são 50, 500 ou 5,000. Então, realizando operações escalares, crie um vetor cujos elementos sejam $\frac{1}{(2n+1)(2n+2)}$. Finalmente, use a função `sum` para somar os termos da série. Compare os resultados das letras a , b e c com o valor de $\ln 2$.

9. Declare as seguintes matrizes:

$$A = \begin{bmatrix} 5 & 2 & 4 \\ 1 & 7 & -3 \\ 6 & -10 & 0 \end{bmatrix} \quad B = \begin{bmatrix} 11 & 5 & -3 \\ 0 & -12 & 4 \\ 2 & 6 & 1 \end{bmatrix} \quad C = \begin{bmatrix} 7 & 14 & 1 \\ 10 & 3 & -2 \\ 8 & -5 & 9 \end{bmatrix}$$

- Calcule $A + B$ e $B + A$ para mostrar que a adição de matrizes é comutativa.
 - Calcule $A + (B + C)$ e $(A + B) + C$ para mostrar que a adição de matrizes é associativa.
 - Calcule $5(A + C)$ e $5A + 5C$ para mostrar que, quando matrizes são multiplicadas por um escalar, a multiplicação obedece à distributividade em relação à soma.
 - Calcule $A*(B + C)$ e $A*B + A*C$ para mostrar que a multiplicação de matrizes é distributiva.
10. Use as matrizes A , B e C do exercício anterior para responder às seguintes questões:
- $A*B = B*A$?
 - $A*(B*C) = (A*B)*C$?
 - $(A*B)^t = B^t*A^t$? (^{t} significa transposta)
 - $(A + B)^t = A^t + B^t$?

11. Resolva o sistema de equações lineares a seguir:

$$5x + 4y - 2z + 6w = 4$$

$$3x + 6y + 6z + 4.5w = 13.5$$

$$6x + 12y - 2z + 16w = 20$$

$$4x - 2y + 2z - 4w = 6$$

12. Um projétil é lançado obliquamente a uma velocidade inicial de 750m/s. Calcule a distância d (isto é, o alcance do projétil) realizando cálculos elemento por elemento, se o ângulo θ inicialmente vale 5° e é ajustado em incrementos regulares de 5° até atingir 85° . Para exibir os resultados, crie uma matriz 17×2 (uma tabela), onde os elementos da primeira coluna representam os ângulos de lançamento e os elementos da segunda coluna representam os alcances correspondentes, arredondados para os inteiros mais próximos.



13. Dois projéteis, A e B , são desferidos obliquamente a partir do mesmo ponto e no mesmo instante de tempo. Os projéteis A e B são desferidos com velocidades iniciais de 680m/s e 780m/s, em ângulos de 65° e 42° , respectivamente. Determine que projétil retorna ao solo primeiro. Em seguida, tome o tempo de voo t_v desse projétil e divida-o em onze partes, originando um vetor t com 11 elementos igualmente espaçados entre si (o primeiro deve ser 0 e o último t_v). Determine a distância relativa entre os dois projéteis para os onze intervalos de tempo do vetor t .

14. O circuito elétrico na figura ao lado possui apenas resistores e fontes de tensão. Determine a corrente em cada resistor usando o método de análise de malhas e a lei de Kirchhoff para as tensões (veja o Problema-exemplo 3-4).

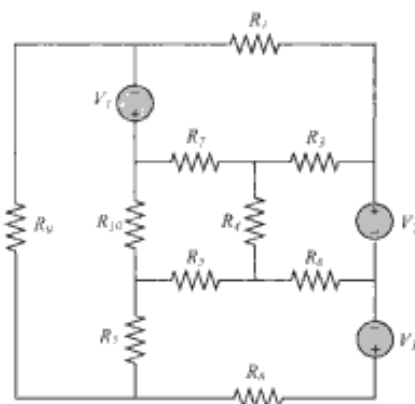
$$V_1 = 38V, \quad V_2 = 20V, \quad V_3 = 24V$$

$$R_1 = 15\Omega, \quad R_2 = 18\Omega, \quad R_3 = 10\Omega$$

$$R_4 = 9\Omega, \quad R_5 = 5\Omega, \quad R_6 = 14\Omega$$

$$R_7 = 8\Omega, \quad R_8 = 13\Omega, \quad R_9 = 5\Omega$$

$$R_{10} = 2\Omega$$



Rotinas (M-Files)

Todos os comandos do MATLAB são executados na janela Command Window. Ao longo deste livro, embora cada comando do MATLAB possa ser executado desse modo, não é conveniente usar a janela Command Window para executar muitos comandos em série – especialmente se eles estiverem encadeados logicamente entre si, isto é, constituírem um programa. Muitas vezes, tentar resolver um determinado problema no MATLAB digitando muitos comandos consecutivos pode ser difícil ou até mesmo impossível. Além disso, a janela Command Window não é interativa. Significa que cada vez que a tecla **Enter** é pressionada, somente o último comando digitado é executado, sendo que o restante do trabalho torna-se inalterável. Se forem necessárias modificações ou correções em um determinado comando da série executada e os resultados gerados por ele afetarem comandos em uma certa sequência encadeada, toda a sequência de comandos deve ser selecionada e executada novamente na linha do *prompt* da janela Command Window, isto é, um comando da série deve ser chamado e executado por vez.

Uma maneira diferente de executar comandos no MATLAB é primeiro criar um arquivo com uma lista de comandos, salvá-la e, então, rodar o arquivo. Quando o arquivo é chamado, os comandos vão sendo executados na ordem em que estão listados no arquivo. Caso sejam necessárias correções ou modificações nesse arquivo, basta abri-lo, modificar os comandos ou a sequência lógica de interesse, salvá-lo e testar o novo arquivo. Os arquivos criados com esse propósito são denominados rotinas, *M-files* ou *script files*.

4.1 OBSERVAÇÕES QUANTO AO USO DE ROTINAS

- Rotina é uma sequência de comandos do MATLAB (também denominada programa).

- Quando uma rotina é testada, o MATLAB executa os comandos na ordem em que eles foram escritos.
- Quando uma rotina possui um comando que gera uma saída (por exemplo, exibição de gráfico, veja Capítulo 5), essa saída é exibida na janela Command Window.
- O uso de rotinas é bastante conveniente, porque elas podem ser reeditadas e executadas muitas vezes.
- É possível digitar e/ou editar rotinas em qualquer editor de texto e, em seguida, colá-las no editor do MATLAB.
- As rotinas do MATLAB são denominadas *M-files*, porque lhes são atribuídas extensões .m ao serem salvas.

4.2 CRIANDO E SALVANDO UMA ROTINA

No MATLAB, rotinas são criadas e editadas na janela Editor/Debugger Window. Essa janela é aberta a partir do Command Window. No menu **File**, selecione **New** e, depois, a opção **M-file**. Um exemplo de janela Editor/Debugger Window é visto na Figura 4-1.

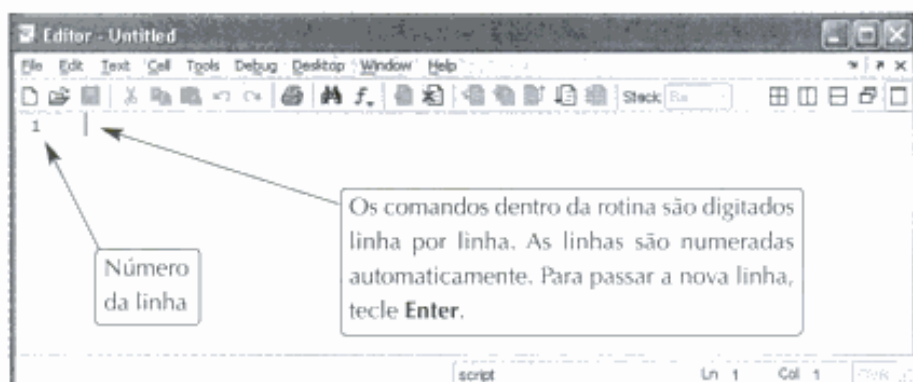


FIGURA 4-1 A janela Editor/Debugger.

Uma vez aberta a janela, os comandos da rotina devem ser digitados linha por linha. O MATLAB enumera automaticamente cada nova linha toda vez que a tecla **Enter** é pressionada. Também é possível digitar os comandos em outro editor de textos e, em seguida, colá-los no Editor/Debugger Window. Uma pequena rotina é mostrada na Figura 4-2. Numa rotina, as primeiras linhas de programa geralmente são dedicadas aos comentários (que não são executados porque diferenciam-se do programa pelo caractere %) e dão uma breve descrição da rotina que se inicia.

Antes de uma rotina ser executada, ela deve ser salva. Para isso, escolha a opção **Save as...** a partir do menu **File** e selecione o caminho e a localização da unidade de armazenamento, preenchendo o campo nome da rotina que está sendo criada. As regras para

se atribuir um nome a uma rotina são as mesmas das variáveis (deve ser iniciada com uma letra, pode incluir dígitos, *underscore* e possuir no máximo 63 caracteres de extensão). Nomes de variáveis declaradas pelo usuário, comandos do MATLAB ou funções nativas não devem ser usados como nome de rotinas.

4.3 RODANDO UMA ROTINA

Uma rotina pode ser executada na janela Command Window por meio da digitação do respectivo nome na linha do *prompt*, seguida da tecla **Enter** ou, então, diretamente da janela Editor Window clicando no ícone **Run** (veja a Figura 4-2). Entretanto, antes disso o usuário deve certificar-se de que o MATLAB consegue encontrar o caminho onde está o arquivo. Conforme o caso, o usuário pode selecionar um caminho diferente no campo Current Directory.

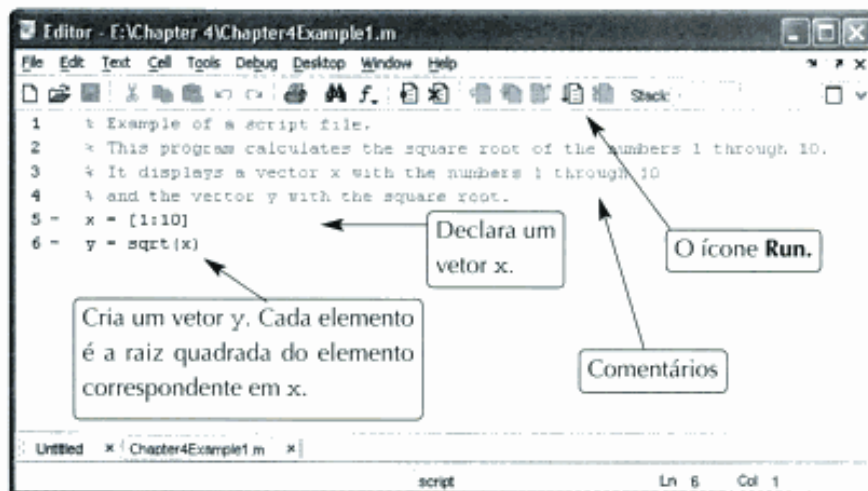


FIGURA 4-2 Um programa digitado na janela Editor/Debugger.

4.3.1 O campo Current Directory

O diretório ou a pasta de trabalho é mostrado no campo Current Directory na barra de ferramentas da janela Command Window (veja a Figura 4-3). O diretório selecionado pode ser modificado nesse mesmo campo ou digitando-se o comando `cd` na linha do *prompt*, seguido do caminho onde se encontra o diretório desejado.

Uma vez que dois ou mais diretórios diferentes são usados em uma sessão, é possível permutar entre os diretórios no campo **Current Directory**. Além do mais, a janela Current Directory pode ser aberta a partir do menu **Desktop**. Essa janela mostra os detalhes dos arquivos em cada diretório e pode ser modificada escolhendo-se uma nova unidade e pasta onde os arquivos serão salvos.

Um modo alternativo de mudar o diretório ou pasta é usar o comando `cd` na janela Command Window. Para escolher uma nova unidade, digite `cd`, espaço e o nome do diretório seguido de dois pontos (:), pressionando **Enter** em seguida. Por exemplo, para mudar o diretório selecionado para a unidade A (geralmente o disco flexível de 3 1/2), di-

gite `cd A:`. Se uma rotina for salva em uma pasta dentro de uma unidade, o caminho completo dessa pasta deve ser especificado. Por exemplo, `cd ('D:\Capitulo_4')` configura o caminho da pasta `Capitulo_4` na unidade D, supondo-se que essa pasta exista na unidade D. A seguir, temos um exemplo onde o diretório selecionado é a unidade A. Então, supondo-se que a rotina da Figura 4-2 tenha sido digitada e salva na unidade A como `Capitulo4_Exemplo1.m`, é executada na linha do *prompt* por meio do nome e da tecla **Enter**.

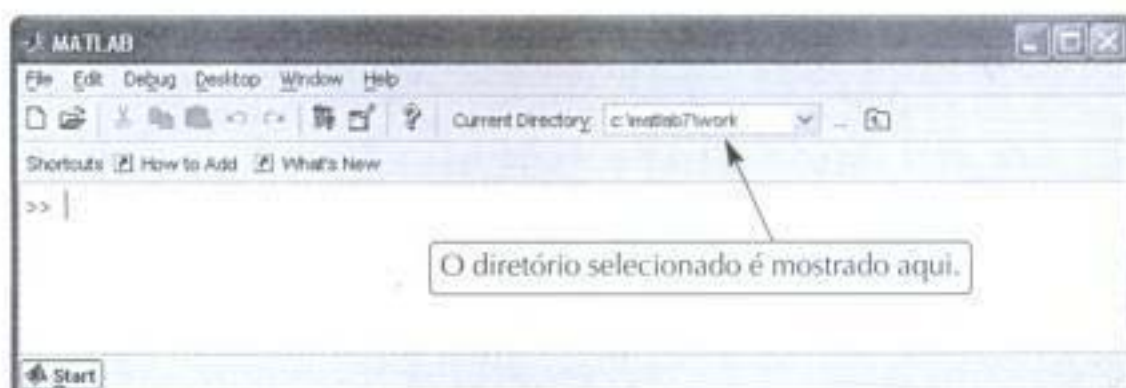


FIGURA 4-3 O campo Current Directory na janela Command Window.

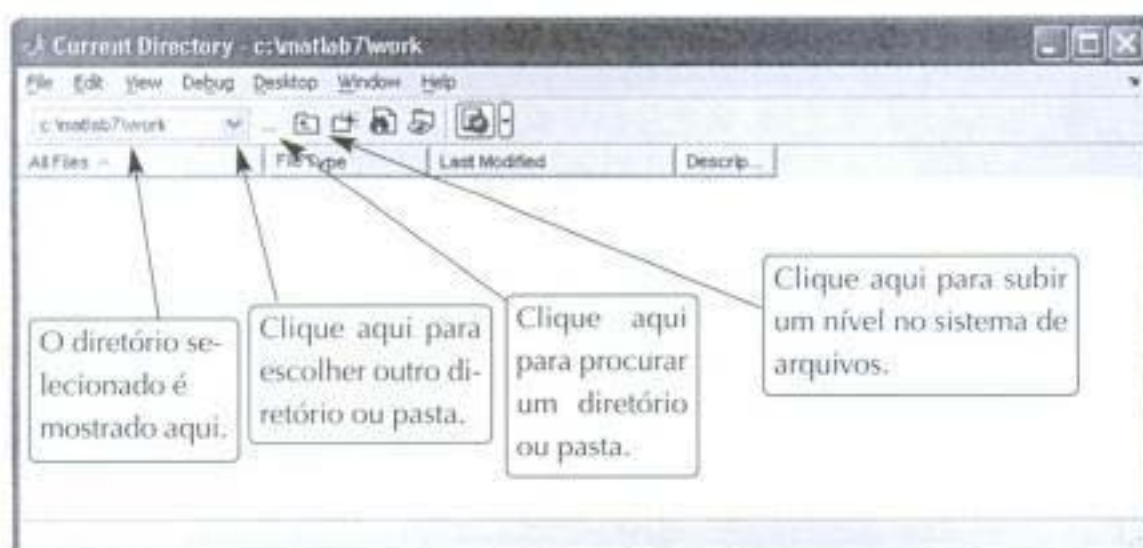
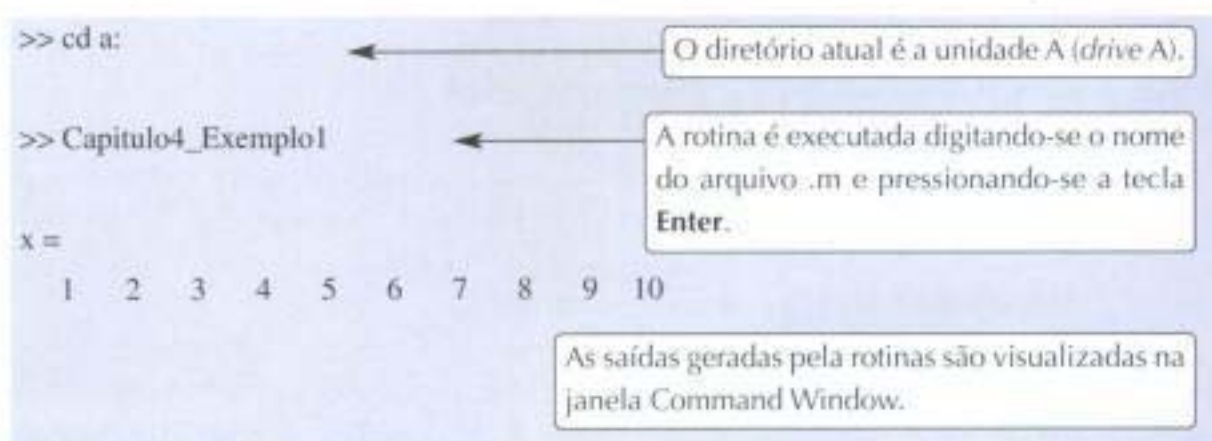


FIGURA 4-4 A janela Current Directory.




```

y =
Columns 1 through 8
    1.0000    1.4142    1.7321    2.0000    2.2361    2.4495    2.6458    2.8284

Columns 9 through 10
    3.0000    3.1623

```

4.3.2 Procurando um diretório ou uma pasta

Quando uma rotina ou uma função é executada, o MATLAB procura pelo arquivo correspondente nos diretórios que estão listados nos caminhos especificados. Os diretórios ou as pastas que estão incluídos nos caminhos procurados são exibidos na janela Set Path, que pode ser aberta selecionando-se **Set Path** no menu **File**. Uma vez aberta, podemos adicionar ou remover novos(as) diretórios (pastas) nessa janela.

4.4 VARIÁVEIS GLOBAIS

Variáveis globais são aquelas que, uma vez criadas em uma parte do MATLAB, são reconhecidas em todas as outras partes do programa. Esse é o caso para variáveis na janela Command Window e as rotinas, visto que ambas agem sobre as variáveis em uma mesma área de trabalho. Quando uma variável é declarada na janela Command Window, pode ser reconhecida e usada em uma rotina. A recíproca também é verdadeira. Em outras palavras, uma variável declarada na janela Command Window ou em uma rotina pode ser usada ou ter o valor modificado a qualquer momento por uma aplicação que faça uso dela.

Há diferentes tipos de arquivos no MATLAB, como os chamados arquivos de funções, que normalmente não compartilham as variáveis com outras partes do programa. Esse aspecto será explicado em detalhes mais adiante, no Capítulo 6 (Seção 6.3).

4.5 ENTRADAS EM ROTINAS

Quando uma rotina é executada, as variáveis usadas nos cálculos dentro do arquivo devem ser inicializadas. A atribuição de um valor a uma variável pode ser feita de três modos diferentes, dependendo de como e onde a variável foi declarada.

1. A variável foi declarada e inicializada dentro da rotina.

Neste caso, a atribuição de um valor à variável é parte integrante da rotina. Se o usuário desejar rodar o arquivo com valores diferentes para a(s) variável(is), o arquivo deve ser reeditado e a(s) variável(is) modificada(s). Então, após o arquivo ser salvo, podemos executá-lo novamente.

A seguir, temos um exemplo. A rotina (salva como `Capitulo4_Exemplo2`) calcula a média dos placares de pontos em três jogos.


```
% Esta rotina calcula a media dos placares em tres jogos.
% A atribuiçao dos valores dos pontos e parte da rotina.
jogo1 = 75;
jogo2 = 93;
jogo3 = 68;
med_pontos = (jogo1 + jogo2 + jogo3)/3
```

As variáveis são inicializadas dentro da rotina.

Rodando o arquivo, a janela Command Window exibe:

```
>> Capitulo4_Exemplo2
```

Para chamar a rotina, basta digitar o nome do arquivo.

```
med_pontos =
    78.6667
>>
```

A variável med_pontos e o respectivo valor são exibidos na janela Command.

2. A variável foi declarada e inicializada na janela Command Window.

Neste caso, a atribuição de um valor à variável é feita na janela Command Window (lembre-se que as variáveis são entendidas dentro da rotina). Se o usuário quiser rodar a rotina com outros valores para a(s) variável(is), basta atribuí-los na janela Command Window e rodar o arquivo novamente.

No exemplo anterior, em que a rotina calculava a média dos escores de pontos em três jogos, a rotina (salva como Capitulo4_Exemplo3) é:

```
% Esta rotina calcula a media dos placares em tres jogos.
% A atribuiçao dos valores dos pontos e parte da rotina.
% jogo1, jogo2 e jogo3 sao inicializadas na janela Command.

med_pontos = (jogo1 + jogo2 + jogo3)/3
```

Rodando o arquivo, a janela Command Window exibe:

```
>> jogo1 = 67;
>> jogo2 = 90;
>> jogo3 = 81;
>> Capitulo4_Exemplo3
```

As variáveis são inicializadas na janela Command.

A rotina é executada na linha do prompt.

```
med_pontos =
    79.3333
```

A saída med_pontos gerada pela rotina para o primeiro conjunto de valores das variáveis jogo1, jogo2 e jogo3 na janela Command.

```

>> jogo1 = 87;
>> jogo2 = 70;
>> jogo3 = 50;

>> Capitulo4_Exemplo3

med_pontos =
    69
>>

```

Novos valores são atribuídos às variáveis jogo1, jogo2 e jogo3.

A rotina é executada novamente.

A saída med_pontos gerada pela rotina para o segundo conjunto de valores das variáveis jogo1, jogo2 e jogo3 na janela Command.

3. A variável foi declarada na rotina, mas a especificação do valor é feita na janela Command Window quando a rotina for executada.

Neste caso, uma variável é declarada dentro do corpo da rotina e, rodando o arquivo, o usuário é solicitado a atribuir um valor à variável na janela Command Window. Para tanto, é necessário utilizar o comando `input` para gerenciar esse tipo de variável.

A forma geral do comando `input` é:

```

nome_da_variavel = input('String com a mensagem a ser
                          exibida na janela Command Window')

```

Executando o comando `input` dentro da rotina, a *string* é exibida na janela Command Window. A *string* deve passar informações sobre o que será atribuído à variável. O usuário deve digitar o valor e pressionar **Enter**. Isso atribui valor à variável. Assim, como acontece com qualquer variável, a variável e o respectivo valor serão exibidos na janela Command Window, se não for utilizado ponto-e-vírgula no final do comando `input`. A seguir, temos uma rotina usando o comando `input` para receber o escore de pontos em cada jogo e calcular a média dos placares.

% Esta rotina calcula a media dos placares em tres jogos.

% O placar de cada jogo e atribuido as variaveis por meio do comando input.

```

jogo1 = input('Entre com o placar do primeiro jogo: ');
jogo2 = input('Entre com o placar do segundo jogo: ');
jogo3 = input('Entre com o placar do terceiro jogo: ');
med_pontos = (jogo1 + jogo2 + jogo3)/3

```

Rodando o arquivo (Capitulo4_Exemplo4), a janela Command Window exibe:

```

>> Capitulo4_Exemplo4
Entre com o placar do primeiro jogo:    67
Entre com o placar do segundo jogo:    91
Entre com o placar do terceiro jogo:    70

```

O computador exibe a mensagem. Em seguida, o valor do placar deve ser digitado pelo usuário e a tecla **Enter** deve ser pressionada.

```
med_pontos =
    76
>>
```

Nesse exemplo, um escalar é atribuído para cada variável. Entretanto, em geral vetores e/ou matrizes também podem receber os valores. Isso é feito digitando-se os elementos do arranjo na ordem em que eles aparecem (colchete à esquerda, digite linha a linha e depois feche o colchete à direita).

Outro uso freqüente do comando `input` ocorre na atribuição de uma *string* a uma variável. É possível fazê-lo de duas formas. Na primeira, usamos o comando da mesma forma recém-discutida e, quando a mensagem aparece no *prompt*, a *string* é digitada entre duas aspas simples (do mesmo modo que uma *string* é atribuída a uma variável sem o comando `input`). A segunda forma requer uma opção no comando `input` que declara os caracteres a serem digitados como uma *string*. A forma geral do comando é:

```
nome_da_variavel = input ('mensagem do prompt', 's')
```

onde a opção 's' dentro do comando define os caracteres que serão digitados como uma *string*. Nesse caso, ao aparecer a mensagem no *prompt*, o texto é digitado sem as aspas e é atribuído à variável como uma *string*. Um exemplo do uso do comando `input` com essa opção é mostrado no Problema-exemplo 7-4.

4.6 COMANDOS DE SAÍDA

Conforme foi discutido anteriormente, o MATLAB gera automaticamente uma saída quando alguns comandos são executados. Por exemplo, quando é atribuído algum valor a uma variável ou o nome de uma variável inicializada previamente é digitado na linha do *prompt* e a tecla **Enter** é pressionada, o MATLAB exibe o nome da variável e o seu respectivo valor. Esse tipo de saída é ocultada simplesmente digitando-se um ponto-e-vírgula no final do comando. Ademais, o MATLAB possui diversos comandos que podem ser utilizados para gerar outros tipos de saídas padronizadas. O conteúdo exibido pode ser uma mensagem informativa, dados numéricos e/ou gráficos. Dois comandos típicos na geração de saídas no MATLAB são o `disp` e o `fprintf`. O comando `disp` exibe uma saída padronizada na tela; o comando `fprintf`, além de gerar uma saída na tela, permite salvá-la em um determinado arquivo. Esses comandos podem ser usados na janela Command Window, em uma rotina e, como veremos nos próximos capítulos, na declaração de funções. Quando esses comandos são utilizados em rotinas, as saídas produzidas são mostradas na janela Command Window.

4.6.1 O comando `disp`

O comando `disp` exibe o conteúdo de uma variável, mas ocultando o nome da mesma, e/ou mostra *strings* na forma de textos. A forma geral do comando `disp` é:

```
disp (nome de uma variável) ou disp ('string de texto')
```

- Toda vez que um comando `disp` é executado, a saída que ele produz é mostrada numa nova linha. Por exemplo:

```
>> abc = [5 9 1; 7 2 4];
```

A variável `abc` é inicializada com uma matriz 2×3 .

```
>> disp(abc)
```

O comando `disp` é utilizado para exibir a matriz `abc`.

```
5 9 1
```

A matriz exibida sem o nome da variável `abc`.

```
7 2 4
```

```
>> disp('O problema nao possui solucao.')
```

```
O problema nao possui solucao.
```

O comando `disp` é utilizado para mostrar uma mensagem.

```
>>
```

O próximo exemplo ensina como usar o comando `disp` em uma rotina para calcular a média dos escores de pontos em três jogos.

```
% Esta rotina calcula a media dos placares em tres jogos.
```

```
% O placar de cada jogo e atribuido as variaveis por meio do
```

```
% comando input.
```

```
% O comando disp e utilizado para mostrar a saída.
```

```
jogo1 = input('Entre com o placar do primeiro jogo: ');
```

```
jogo2 = input('Entre com o placar do segundo jogo: ');
```

```
jogo3 = input('Entre com o placar do terceiro jogo: ');
```

Salta uma linha.

```
med_pontos = (jogo1 + jogo2 + jogo3)/3;
```

```
disp('')
```

Exibe o texto indicando média.

```
disp('A media dos pontos dos tres jogos e: ')
```

```
disp('')
```

Salta uma linha.

```
disp(med_pontos)
```

Exibe o resultado da média.

Quando esse arquivo (Capitulo4_Exemplo5) for executado, a saída padrão na janela Command Window será:

```
>> Capitulo4_Exemplo5
```

```
Entre com o placar do primeiro jogo: 89
```

```
Entre com o placar do segundo jogo: 60
```

```
Entre com o placar do terceiro jogo: 82
```

Quebra de linha.

A media dos pontos dos tres jogos e:

77

A linha texto é exibida.

Outra quebra de linha.

O conteúdo da variável `med_pontos` é exibido.

- Apenas uma variável por vez pode ser utilizada em um comando `disp`. Se os elementos de duas variáveis forem necessários simultaneamente na tela, uma nova variável contendo os elementos dessas duas variáveis deverá ser declarada antes de se utilizar o comando `disp`.

Em muitas situações, é interessante apresentar números na forma de tabela. Nesses casos, primeiro declare um arranjo com os números desejados e, em seguida, use o comando `disp` para mostrar o arranjo na tela. Podemos criar ainda cabeçalhos para as colunas com esse comando. Como o comando `disp` não permite que o usuário formate a saída (a largura das colunas e a distâncias entre elas), a posição do cabeçalho deve ser ajustada às colunas por meio de espaços. No exemplo a seguir, uma rotina ensina como exibir, em uma tabela, os dados populacionais do Capítulo 2.

```
yr = [1984 1986 1988 1990 1992 1994 1996];
```

```
pop = [127 130 136 145 158 178 211];
```

Os dados populacionais são recebidos em dois vetores linha.

```
tabelaYP(:,1) = yr';
```

`yr` é colocado na primeira coluna da matriz `tabelaYP`.

```
tabelaYP(:,2) = pop';
```

`pop` é colocado na primeira coluna da matriz `tabelaYP`.

```
disp('    ANO    POPULACAO')
```

Exibe o cabeçalho na primeira linha da `tabelaYP`.

```
disp('    (MILHOES)')
```

Exibe a unidade na segunda linha da `tabelaYP`.

```
disp('')
```

Salta uma linha

```
disp(tabelaYP)
```

Exibe o conteúdo da matriz `tabelaYP`.

Quando essa rotina (salva como `PopTable`) é executada, a saída visualizada na janela Command Window será:

```
> PopTable
```

```
    ANO    POPULACAO
    (MILHOES)
```

A matriz `tabelaYP` é exibida.

1984	127
1986	130
1988	136
1990	145
1992	158
1994	178
1996	211

A matriz `tabelaYP` é exibida.

Outro exemplo de tabela formatada pode ser encontrado no Problema-exemplo 4-3. Além disso, tabelas também podem ser criadas e exibidas a partir do comando explicado a seguir, o `fprintf`.

4.6.2 O comando `fprintf`

Já foi mencionado que o comando `fprintf` pode ser utilizado para gerar saídas (texto e dados) na tela ou para salvá-las em um arquivo. Ao contrário do `disp`, nesse comando a saída pode ser formatada pelo usuário. Por exemplo, texto e dados numéricos das variáveis podem ser combinados e exibidos em uma linha. Além disso, o formato dos números também pode ser controlado.

Com tantas opções disponíveis é de se esperar que o comando `fprintf` disponha de mais opções no argumento. Para evitar confusões, apresentaremos o comando paulatinamente. Em primeiro lugar, mostraremos como usá-lo para mostrar mensagens na forma de textos, combinar texto e dados numéricos e, por último, como salvar a saída em um arquivo.

Usando o comando `fprintf` para exibir textos:

Para exibir textos, a forma geral do comando `fprintf` é:

```
fprintf ('string de texto')
```

Por exemplo:

```
fprintf ('O problema, como esta, nao possui solucao. Por favor, verifique o problema.')
```

Se essa linha fizer parte de uma rotina, ao executá-la teremos a seguinte mensagem:

```
O problema, como esta, nao possui solucao. Por favor, verifique o problema.
```

Ao utilizar o comando `fprintf` é possível promover quebras de linha em qualquer instância da *string*. Para tanto, basta inserir `\n` (barra invertida n) antes da quebra de linha. Por exemplo, inserindo `\n` após a primeira sentença do exemplo acima, temos:

```
fprintf ('O problema, como esta, nao possui solucao. \nPor favor, verifique o problema.')
```


Ao executá-la, teremos a seguinte mensagem:

O problema, como esta, nao possui solucao.
Por favor, verifique o problema.



O caractere `\n` é denominado caractere de escape. Deve ser utilizado sempre que for necessário controlar o formato das mensagens na tela. Outros caracteres para modificação do modo de exibição dos dados são:

`\b` Retrocesso
`\t` Tabulação

Se um programa possuir mais de um comando `fprintf`, as saídas serão geradas continuamente à medida que cada `fprintf` é executado. Lembre-se que o `fprintf` não inicia uma nova linha ao ser executado. Por isso, se houver outros comandos entre dois `fprintf`, é essencial o uso do modificador `\n`. Observe a seguinte rotina:

```
fprintf('O problema, como esta, nao possui solucao. Por favor, verifique os dados de entrada.')
x = 6; d = 19 + 5*x;
fprintf('Tente rodar o programa mais tarde.')
y = d + x;
fprintf('Use diferentes valores para as entradas').
```

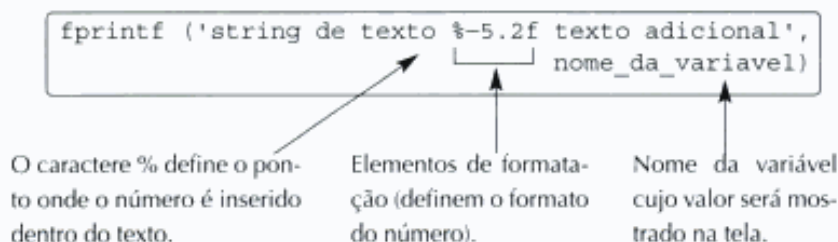
Executando a rotina, a saída na janela Command Window será:

O Problema, como esta, nao possui solucao. Por favor, verifique o programa. Tente rodar o programa a seguir. Use diferentes valores para a entrada.

Assim, o exemplo anterior ressalta que, para iniciar uma nova linha com o comando `fprintf`, é necessário introduzir o modificador `\n` onde se deseja realizar a quebra de linha.

Usando o comando `fprintf` para combinar texto e números na tela:

A combinação de texto e número na função `fprintf` é feita no seguinte formato:



Os elementos de formatação são:



Os caracteres possíveis no campo de *flag* são:

Caractere utilizado no <i>flag</i>	Descrição
- (sinal negativo)	Justifica o número dentro do campo.
+ (sinal positivo)	Imprime um caractere de sinal (+ ou -) à frente do número.
0 (zero)	Adiciona zeros, se o número for menor que o especificado no campo de precisão.

O campo de precisão (5.2 no exemplo anterior) é opcional. O primeiro número (5, no exemplo) é o campo de largura. Especifica o número de dígitos que os números terão ao serem exibidos. Se o número a ser impresso for menor do que a quantidade selecionada, zeros serão introduzidos à esquerda desse número. O segundo número define a precisão em termos de casas decimais (2, no exemplo anterior). Especifica o número de dígitos a serem mostrados à direita da vírgula (ponto, no caso do MATLAB).

O último elemento de formatação é o caractere de conversão (obrigatório). Especifica a representação segundo a qual o número será mostrado na tela. Veja alguns dos possíveis caracteres de conversão:

- e Número em notação científica com o “e” minúsculo (p. ex.: 1.709098e+001).
- E Número em notação científica com o “E” maiúsculo (p. ex.: 1.709098E+001).
- f Ponto flutuante decimal (p. ex.: 17.090980).
- g “f” ou “e” (o mais curto).
- G “f” ou “E” (o mais curto).
- i Inteiro

Mais informações sobre os caracteres de conversão podem ser obtidas no Help do MATLAB. Tome a seguinte rotina como exemplo de combinação texto + dados no `fprintf`:

```

% Esta rotina calcula a media dos placares em tres jogos.
% O placar de cada jogo e atribuido as variaveis por meio do comando input.
% O comando fprintf e utilizado para mostrar a saida.
jogo(1) = input('Entre com os pontos do escore do primeiro jogo: ');
jogo(2) = input('Entre com os pontos do escore do segundo jogo: ');
jogo(3) = input('Entre com os pontos do escore do terceiro jogo: ');
  
```

```
med_pontos = mean(jogo);
```

```
fprintf('A media dos escores dos tres jogos foi %f pontos.', med_pontos)
```

Texto

% indica a
posição do
número.

Texto adi-
cional.

Nome da variável cujo
conteúdo será exibido.

Perceba que, além de usar a função `fprintf`, esse exemplo ilustra o uso de vetor, diferentemente do exemplo apresentado no início do capítulo, pois armazena os escores dos placares nos três primeiros elementos de um vetor, denominado `jogo`, e a média dos escores era calculada usando-se a função `mean`. Rodando este último exemplo, a saída gerada na janela do Command Window é:

```
>> Capitulo4_Exemplo7
```

```
Entre com os pontos do escore do primeiro jogo 75
```

```
Entre com os pontos do escore do segundo jogo 60
```

```
Entre com os pontos do escore do terceiro jogo 81
```

```
A media dos escores dos tres jogos foi 72.00000.
```

```
>>
```

A saída padronizada gerada pelo comando `fprintf` combina texto e número.

O comando `fprintf` não fica limitado a um único elemento de formatação. Basta digitar % seguido de qualquer caractere de formatação no lugar onde se deseja introduzir os números. Em seguida, após a *string* de comando (seguida de vírgula), os nomes das variáveis são digitados na ordem em que devem ser inseridos no texto. Em geral, o formato do comando é:

```
fprintf ('...texto...%g...%g...%f...',variável1,  
variável2,variável3)
```

O exemplo a seguir ilustra esse último formato de `fprintf`:

```
% Este programa calcula o alcance de lançamento de um projétil,
```

```
% dados a velocidade inicial e o ângulo de lançamento.
```

```
% O comando fprintf é utilizado para exibir texto e números.
```

```
v = 1584; % Velocidade inicial (em km/h)
```

```
teta = 30; % ângulo (em graus)
```

```
vms = v*1000/3600;
```

```
t = vms*sin(30*pi/180)/9.81;
```

Calcula a velocidade em m/s.

Determina o tempo até o máximo.

```
d = vms*cos(30*pi/180)*2*t/1000;
```

Calcula o alcance do projétil.

```
fprintf('O projétil foi lançado a %3.2f graus, com velocidade inicial %4.2f km/h, e  
atingira o solo a uma distancia de %g km.', teta, v, d)
```

Salve a rotina com o nome de `Capitulo4_Exemplo8` e execute-a digitando esse nome na linha do *prompt* na janela Command Window. O resultado será:

```
>> Capitulo4_Exemplo8
```

```
O projétil foi lançado a 30 graus, com velocidade inicial 1584.00 km/h, e atingira o solo a  
uma distancia de 17.091 km.
```

```
>>
```

Observações adicionais quanto ao uso do comando `fprintf`:

- Caso seja necessário uma aspa simples no texto de saída, digite duas aspas simples dentro do corpo da *string*, no local onde deverá ser exibido no texto.
- O comando `fprintf` é “vetorizado”. Significa que, se um arranjo do tipo vetor ou matriz estiver incluído no comando, haverá repetição desse comando até que o último elemento do vetor seja exibido na tela. Se a variável for do tipo matriz, os dados serão utilizados coluna por coluna.

Por exemplo, a rotina a seguir cria uma matriz T (2×5) cuja primeira linha são os números de 1 a 5 e, a segunda, as correspondentes raízes:

```
x = 1:5;
```

Declara um vetor x.

```
y = sqrt(x);
```

Declara um vetor y.

```
T = [x; y]
```

Cria a matriz T (2×5); primeira linha é x e a segunda é y.

```
fprintf('Se o numero e %i, sua raiz quadrada e %f\n',T)
```

O comando `fprintf` exibe dois números de T em cada linha.

Executando o arquivo `Capitulo4_Exemplo9` na linha do *prompt*:

```
>> Capitulo4_Exemplo9
```

```
T =
```

```
1.0000 2.0000 3.0000 4.0000 5.0000
```

```
1.0000 1.4142 1.7321 2.0000 2.2361
```

A matriz T (2×5).

```
Se o numero e 1, sua raiz quadrada e 1.000000
```

```
Se o numero e 2, sua raiz quadrada e 1.414214
```

Se o número é 3, sua raiz quadrada é 1.732051

Se o número é 4, sua raiz quadrada é 2.000000

Se o número é 5, sua raiz quadrada é 2.236068

O comando `fprintf` é repetido 5 vezes, usando os números da matriz `T` (coluna por coluna).

Usando o comando `fprintf` para salvar o resultado de saída num arquivo:

O comando `fprintf` pode ser utilizado para gravar os resultados de saída em um arquivo. Os dados salvos podem ser posteriormente usados no MATLAB ou em outras aplicações.

São necessárias três etapas para escrever um resultado gerado pelo `fprintf`:

- Abrir um arquivo usando o comando `fopen`.
- Escrever a saída no arquivo aberto usando o comando `fprintf`.
- Fechar o arquivo usando o comando `fclose`.

Passo a:

Antes de os dados serem gravados em um arquivo, esse arquivo deve ser aberto. O comando `fopen` tem esse propósito. Com ele, é possível abrir um arquivo novo ou algum arquivo já existente. A forma geral do comando `fopen` é:

```
fid = fopen ('nome_do_arquivo', 'permissao')
```

onde `fid` é uma variável denominada identificador do arquivo. Um número é atribuído à variável `fid` ao executar `fopen`. Além disso, o nome do arquivo deve ser escrito, incluindo a extensão, entre aspas simples. O código de permissão é um campo (escrito como *string*) que norteia como o arquivo é aberto. Algumas permissões típicas são:

- 'r' Abrir arquivo para leitura (*default*).
- 'w' Abrir arquivo para escrita. Se o arquivo existir, seu conteúdo será excluído. Caso contrário, um novo arquivo é criado.
- 'a' Idêntico à permissão 'w', exceto que, se o arquivo existir, o dado será anexoado ao conteúdo existente nesse arquivo (no final).

Se uma permissão não for especificada, o arquivo será aberto na opção 'r'. Outras permissões são descritas no menu Help.

Passo b:

Uma vez que o arquivo encontra-se aberto, o comando `fprintf` pode ser usado para escrever nele. O comando deve ser utilizado como na criação de rotinas, exceto que a variável `fid` deve ser inserida dentro do comando. A forma geral do `fprintf` para esses propósitos é:

```
fprintf(fid, 'texto % -5.2f texto adicional', nome_da_variavel)
```

fid é adicionada ao comando `fprintf`.

Passo c:

Ao completar a escrita dos dados no arquivo, ele deve ser fechado por meio do comando `fclose`, que tem a seguinte forma geral:

```
fclose (fid)
```

Notas adicionais quanto ao uso do comando `fprintf` para salvar uma saída num arquivo:

- O arquivo criado é salvo automaticamente no diretório selecionado em Current Directory.
- É possível usar o comando `fprintf` para escrever múltiplos arquivos diferentes. Para isso, os arquivos devem ser abertos, atribuídos a diferentes `fids` (p. ex.: `fid1`, `fid2`, `fid3`, etc.) e, então, escrever o `fid` de um arquivo específico no comando `fprintf` para gravar o conteúdo nesse arquivo.

A seguir é mostrado um exemplo de rotina que utiliza o comando `fprintf` para salvar resultados de saída em dois arquivos. O programa no arquivo gera duas tabelas de conversão. Uma tabela converte velocidades em milhas por hora (mi/h) em velocidades em quilômetros por hora (km/h). A outra tabela converte libras (lb) em Newtons (N). Cada uma das tabelas é salva em um arquivo-texto (extensão `.txt`) diferente.

% Rotina com `fprintf` que escreve os resultados de saída em arquivos.

% Duas tabelas de conversoes sao criadas e salvas em dois arquivos diferentes.

% Uma tabela converte mi/h em km/h e a outra converte lb em N.

clear all

Vmph = 10:10:100;

Gerando um vetor de velocidades em mph.

Vkmh = Vmph.* 1.609;

Convertendo mph em km/h.

TBL1 = [Vmph; Vkmh];

Criando uma tabela (matriz) com duas linhas.

F1b = 200:200:2000;

Gerando um vetor de forças em lb.

FN = F1b.* 4.448;

Convertendo lb em N.

TBL2 = [F1b; FN];

Criando uma tabela (matriz) com duas linhas.

fid1 = fopen ('Vmph_Vkmh.txt', 'w');

Abrindo um arquivo txt chamado Vmph_Vkmh.

fid2 = fopen ('F1b_FN.txt', 'w');

Abrindo um arquivo txt chamado F1b_FN.

fprintf (fid1, 'Tabela de Conversao de Velocidades\n\n');

Escrevendo um título e um espaço no arquivo fid1.


```
fprintf(fid1, '    mi/h    km/h  \n');
```

Escrevendo o cabeçalho em duas colunas do arquivo fid1.

```
fprintf(fid1, '    %8.2f    %8.2f\n', TBL1);
```

Escrevendo os dados da variável TBL1 no arquivo fid1.

```
fprintf(fid2, 'Tabela de Conversao de Forcas\n\n');
```

```
fprintf(fid2, '    Libras    Newtons\n');
```

```
fprintf(fid2, '    %8.2f    %8.2f\n', TBL2);
```

Escrevendo a tabela de conversão de força (dados na variável TBL2) para o arquivo fid2.

```
fclose(fid1);
```

```
fclose(fid2);
```

Fechando os arquivos fid1 e fid2.

A execução dessa rotina gera e salva dois novos arquivos-texto, nomeados Vmph_Vkmh.txt e Flb_FN.txt, no diretório selecionado em Current Directory. Esses arquivos podem ser abertos em qualquer editor de textos. As Figuras 4-5 e 4-6 mostram como os dois arquivos aparecem quando são abertos no Microsoft Word.

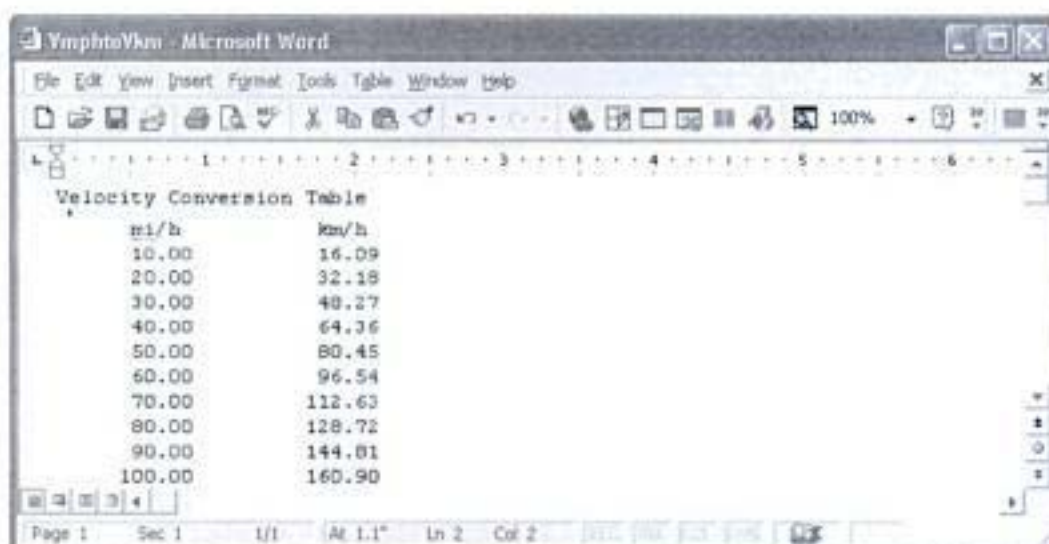


FIGURA 4-5 O arquivo Vmph_Vkmh.txt aberto no Word.

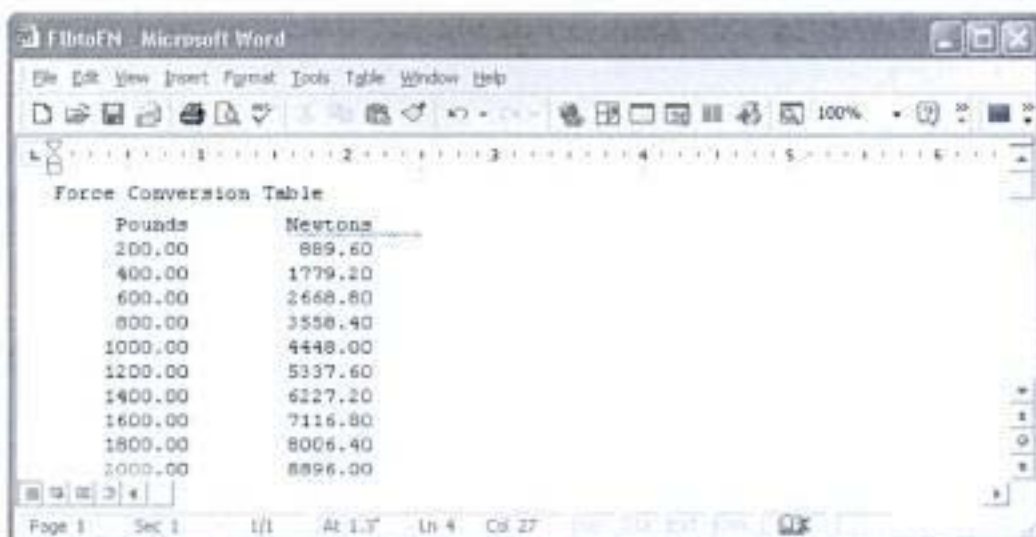


FIGURA 4-6 O arquivo Flb_FN.txt aberto no Word.

4.7 IMPORTANDO E EXPORTANDO DADOS

O MATLAB é usado freqüentemente para analisar dados gravados em experimentos ou gerados por outros programas de computador. Para realizar a análise dos dados, é necessário importá-los para o ambiente MATLAB. Outras vezes, dados gerados no MATLAB precisam ser utilizados por outras aplicações. Há muitos tipos de dados que o MATLAB pode tratar: numéricos, textos, áudio, gráficos e imagens. Esta seção descreve apenas como importar e exportar dados numéricos, o que provavelmente é a aplicação mais comum utilizada pelos novos usuários do MATLAB. Para outros tipos de transferência de dados, consulte o Help do programa e selecione a opção File I/O.

A importação de dados pode ser feita por comandos ou utilizando-se o Assistente de Importação (Import Wizard). Os comandos são bastante úteis quando o formato dos dados a serem importados é conhecido previamente. A diversidade de comandos do MATLAB destinados à importação de dados numéricos é impressionante de tão completa. Os comandos de importação de dados também podem fazer parte de uma rotina, de modo que os dados sejam importados apenas quando a rotina for executada. O Assistente de Importação mostra-se útil quando o formato dos dados ou o comando aplicável ao tipo de dados que se deseja importar é desconhecido. O Assistente de Importação se encarrega de determinar o formato dos dados e importá-los automaticamente.

4.7.1 Comandos para importação e exportação de dados

Esta seção descreve em detalhes como transferir dados para dentro e para fora de uma planilha do Excel. O Microsoft Excel é utilizada largamente para organizar/armazenar dados; o Excel é compatível com muito dispositivos de gravação e aplicações de computador. Além disso, as pessoas freqüentemente preferem importar e exportar vários formatos de dados para dentro e para fora do Excel. O MATLAB também possui comandos para transferir dados diretamente para e de outros formatos, como o csv e ASCII, e para planilhas padrão Lotus 123. Os detalhes desses e de muitos outros comandos podem ser encontrados no Help do MATLAB, na seção File I/O.

Importando e exportando dados para e do Excel:

A importação de dados do Excel é feita por meio do comando `xlsread`. Quando o comando é executado, os dados da planilha são organizados em um arranjo para uma variável. A forma mais simples do comando `xlsread` é:

```
Nome_variavel = xlsread ('Nome_arquivo')
```

- 'Nome_arquivo' (digitado como uma *string*) é o nome do arquivo do Excel. O diretório do Excel deve concordar com o diretório especificado no campo Current Directory ou listado no caminho de procura.
- Se o arquivo do Excel tiver mais de uma folha, os dados serão importados apenas da primeira folha.

Quando um arquivo do Excel tiver mais de uma folha, o comando `xlsread` é capaz de especificar a folha de origem dos dados. A forma do comando passa a ser então:

```
Nome_variavel = xlsread ('Nome_arquivo', 'Nome_folha')
```

- O nome da folha deve ser digitado como uma *string*, isto é, entre aspas simples.

Outra opção é importar somente uma parte dos dados contidos em uma planilha. Para tanto, digite o argumento adicional no comando `xlsread`:

```
Nome_variavel = xlsread ('Nome_arquivo', 'Nome_folha',  
                        'Intervalo')
```

- O 'Intervalo' (digitado como *string*) é uma região retangular da planilha de onde se deseja fazer aquisição dos dados, endereçada na notação do Excel. Por exemplo, 'C2:E5' é uma região 4 por 3 de linhas 2, 3 e 4 e colunas C, D e E.

A exportação de dados do MATLAB para uma planilha do Excel é feita por meio do comando `xlswrite`. A forma mais simples desse comando é:

```
xlswrite ('Nome_arquivo', Nome_variavel)
```

- 'Nome_arquivo' (digitado como *string*) é o nome do arquivo do Excel para o qual os dados devem ser exportados. O arquivo deve ser especificado no Current Directory. Se o arquivo não existir, um novo arquivo do Excel com o nome especificado deve ser criado.
- Nome_variavel é o nome da variável do MATLAB de onde os dados serão exportados.
- Os argumentos 'Nome_folha' e 'Intervalo' podem ser utilizados no argumento do comando `xlswrite` nas exportações de dados para uma folha específica ou para um intervalo específico de uma folha, respectivamente.

Como exemplo, os dados mostrados na planilha do Excel da Figura 4-7 serão importados para o MATLAB por meio do comando `xlsread`.

	A	B	C	D	E	F	G	H	I	J
1	11	2	34	14	-6	0	8			
2	15	6	-20	8	0.56	33	5			
3	0.9	10	3	12	-25	-0.1	4			
4	55	9	1	-0.555	17	5	-30			
5										

FIGURA 4-7 Planilha do Excel alimentada com dados.

A planilha foi salva em um arquivo denominado `TestData1` em um disquete na unidade A. Especificando o `Current Directory` para o drive A, os dados são importados para o MATLAB e atribuídos à variável `DADOS`:

```
>> DADOS=xlsread('TestData1')
```

`DADOS =`

11.0000	2.0000	34.0000	14.0000	-6.0000	0	8.0000
15.0000	6.0000	-20.0000	8.0000	0.5600	33.0000	5.0000
0.9000	10.0000	3.0000	12.0000	-25.0000	-0.1000	4.0000
55.0000	9.0000	1.0000	-0.5550	17.0000	6.0000	-30.0000

4.7.2 Usando o assistente de importação

Utilizar o Assistente de Importação (Import Wizard) é provavelmente o modo mais simples de se importar dados para o MATLAB, já que o usuário não precisa especificar o formato dos dados. O Assistente de Importação é ativado selecionando-se **Import Data** no menu **File** da janela `Command Window` (também podemos inicializá-lo digitando o comando `uiimport` na linha do *prompt*). O Assistente de Importação inicia exibindo a caixa de seleção de arquivos que o Assistente consegue reconhecer. O usuário seleciona então o arquivo de onde os dados devem ser importados e clica em **Open**. O Assistente de Importação abre o arquivo e exibe uma pequena porção dos dados truncados na caixa denominada *preview* do assistente, de modo que o usuário possa verificar a autenticidade dos dados importados. O Assistente de Importação tenta processar os dados e, se a importação for bem-sucedida, são exibidas as variáveis criadas especificamente para receber os dados importados. Em seguida, o usuário clica em **next** e o assistente mostra o `Column Separator` (separador de colunas) utilizado. Se a variável receber os dados corretamente, o usuário poderá continuar a utilizar o Assistente clicando em **next**; caso contrário, poderá escolher um separador de colunas diferente.

Na próxima janela, o assistente mostra o nome e o tamanho da variável criada no MATLAB para receber os dados (quando todos os dados são numéricos, a variável do MATLAB recebe o mesmo nome do arquivo de onde os dados foram importados). Quando o assistente é finalizado (clicando-se em **finish**), os dados são importados para o MATLAB.

A seguir vamos ilustrar o processo de importação de dados numéricos no formato ASCII via Assistente de Importação. Vamos importar um arquivo salvo no formato `txt`. Os dados salvos com o nome de `TestData` são mostrados na Figura 4-8.

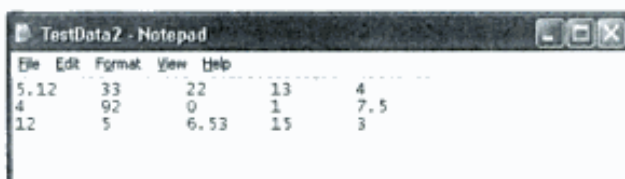


FIGURA 4-8 Dados numéricos no formato ASCII.

As telas exibidas durante o processo de utilização do Assistente de Importação para importar o arquivo TestData2 são mostradas nas Figuras 4-9 a 4-11. A terceira figura mostra que o nome da variável do MATLAB que irá receber os dados é TestData2 e o tamanho dela é 3 por 5. Na janela Command Window, os dados importados são mostrados digitando-se o nome da variável sem ponto-e-vírgula no final.

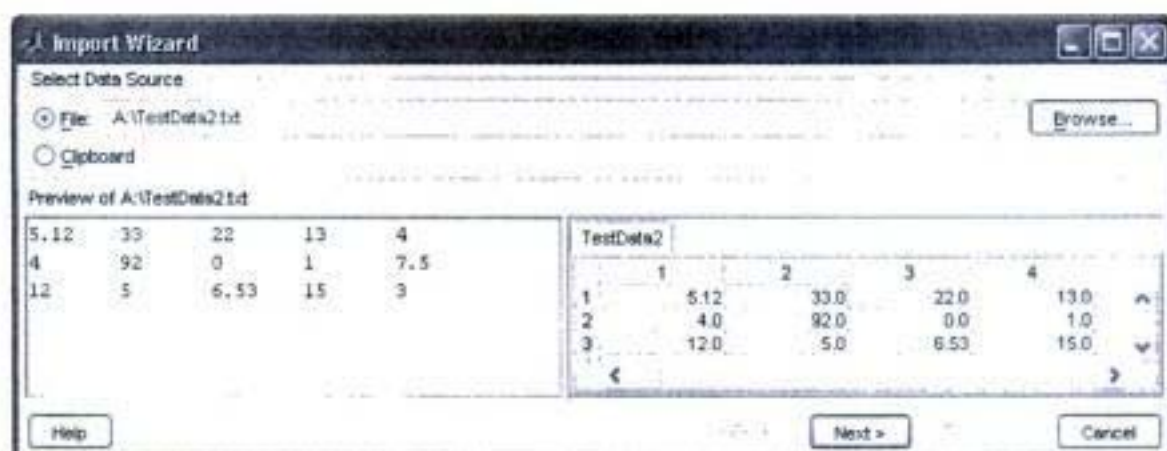


FIGURA 4-9 Assistente de importação, primeira janela.

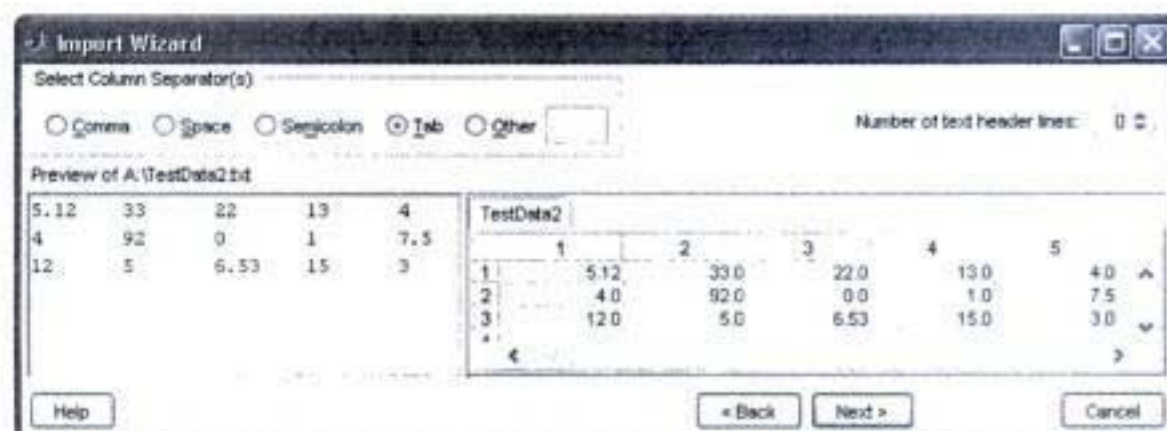


FIGURA 4-10 Assistente de importação, segunda janela.

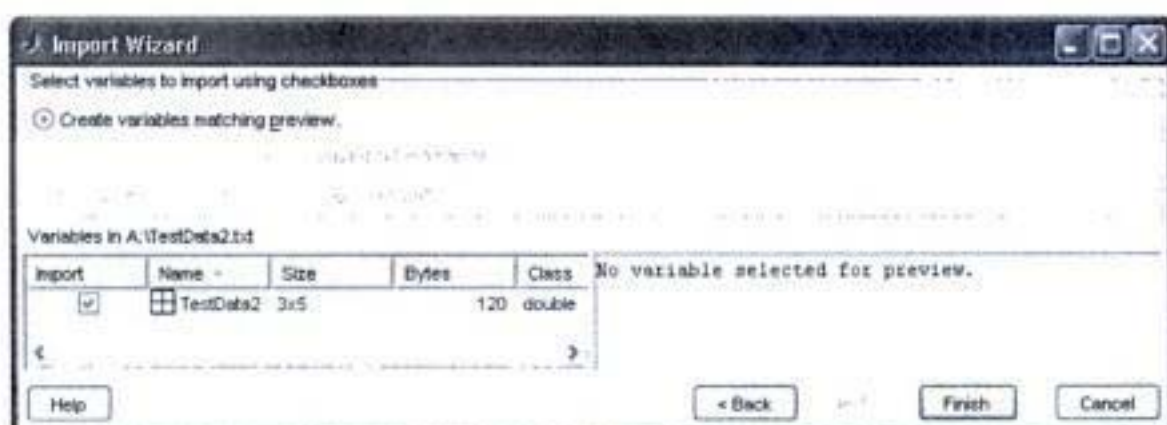


FIGURA 4-11 Assistente de importação, terceira janela.

```
>> TestData2
```

```
TestData2 =
```

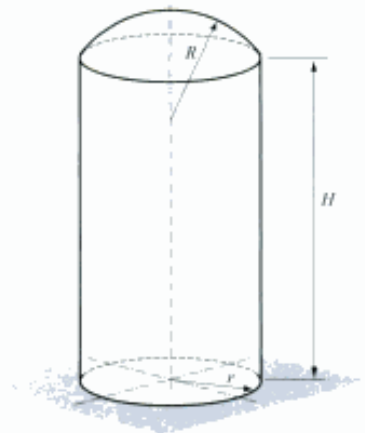
5.1200	33.0000	22.0000	13.0000	4.0000
4.0000	92.0000	0	1.0000	7.5000
12.0000	5.0000	6.5300	15.0000	3.0000

4.8 EXEMPLOS DE APLICAÇÕES DO MATLAB

Problema-exemplo 4-1: Altura e área de superfície de um silo

Um silo cilíndrico de raio r possui uma cobertura na forma de uma calota esférica de raio R . A altura da porção cilíndrica é H . Escreva uma rotina que determine a altura H dados os valores de r , R e o volume V . Em seguida, o programa também deve calcular a área da superfície do silo.

Finalmente, use o programa para calcular a altura e a área da superfície de um silo de $r = 30\text{m}$, $R = 45\text{m}$ e volume de 120.000m^3 . Atribua valores a r , R e V na linha do *prompt* em Command Window.



Solução

O volume total do silo é o obtido por adição dos volumes da parte cilíndrica e da calota esférica. O volume do cilindro é dado por:

$$V_{\text{cilin}} = \pi r^2 H$$

e o volume da calota esférica é:

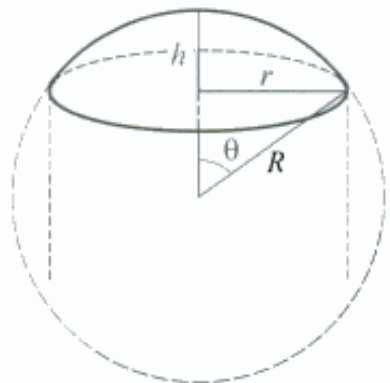
$$V_{\text{cal}} = \frac{1}{3} \pi h^2 (3R - h)$$

onde $h = R - R \cos \theta = R(1 - \cos \theta)$ e θ é calculado de $\sin \theta = \frac{r}{R}$. Utilizando as equações acima, a altura H do cilindro pode ser expressa como:

$$H = \frac{V - V_{\text{cal}}}{\pi r^2}$$

A área de superfície do silo é obtida pela adição das áreas de superfície da parte cilíndrica (A_{cilin}) e esférica (A_{cal}):

$$S = S_{\text{cilin}} + S_{\text{cal}} = 2\pi r H + 2\pi R h$$



Uma rotina que resolve o problema acima é apresentada a seguir:

```
teta = asin(r/R);
h = R*(1-cos(teta));
Vcal = pi*h^2*(3*R - h)/3;
H = (V-Vcal)/(pi*r^2);
S = 2*pi*(r*H + R*h);
fprintf('Altura H e: %.2f m.', H)
fprintf('\nA area de superficie do silo e: %.2f m2.', S)
```

Calculando θ .

Calculando h .

Calculando o volume da calota esférica.

Calculando H .

Calculando a área de superfície S .

Na janela Command Window, executando o arquivo `Capitulo4_Exemplo10` temos:

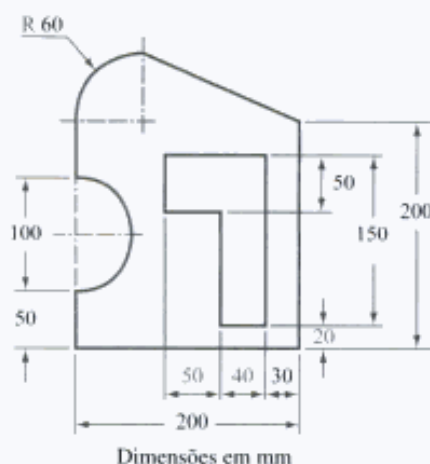
```
>> r=30; R= 45; V = 200000;
>> Capitulo4_Exemplo10
Altura H e: 36.43 m.
A area de superficie do silo e: 10107.44 m2.
```

Atribuindo valores a r , R e V .

Executando o arquivo `Capitulo4_Exemplo10`.

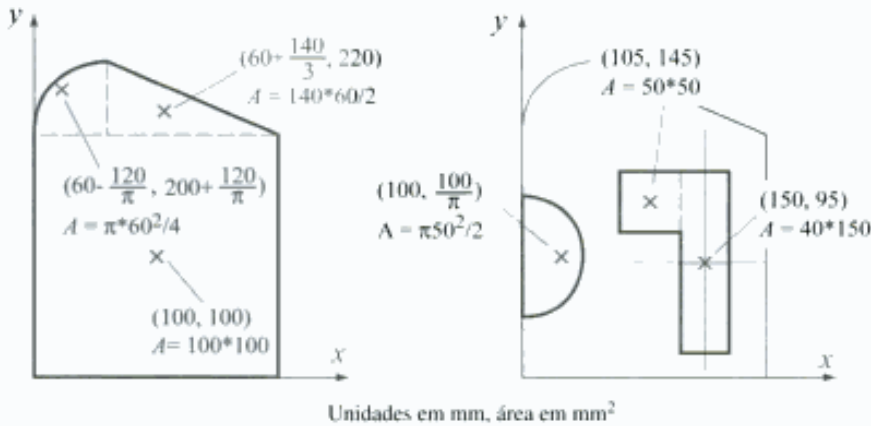
Problema-exemplo 4-2: Centro geométrico (centróide) de um objeto composto

Escreva uma rotina que calcule as coordenadas do centro geométrico de uma figura (o objeto composto pode ser decomposto em seções cujos centróides são conhecidos). Divida o objeto em seções e descubra as coordenadas dos centros geométricos de cada seção. Ao executar a rotina, o programa deve solicitar três números, que devem ser armazenados em uma linha de uma matriz. O número de linhas da matriz deve concordar com o número de seções que o objeto possuir. Além disso, uma seção que representa um buraco no objeto deve possuir área negativa. Como saída, o programa deve exibir as coordenadas do centróide do objeto composto. Use o programa para calcular o centróide da área mostrada na figura.



Solução

A área é dividida em seis seções, como mostra a figura a seguir. A área total é a soma das áreas das seções à esquerda menos a soma das três áreas à direita. A localização e as coor-



denadas do centróide de cada seção estão assinaladas na figura, assim como a área de cada seção.

As coordenadas do centróide $\bar{X}$ e $\bar{Y}$ do objeto composto são dados por: $\bar{X} = \frac{\sum A\bar{x}}{\sum A}$ e $\bar{Y} = \frac{\sum A\bar{y}}{\sum A}$, onde $\bar{x}$, $\bar{y}$ e A são as coordenadas do centróide e a área de cada seção, respectivamente.

Um programa escrito em uma rotina para calcular as coordenadas do centróide do objeto composto é mostrado a seguir:

```
% Este programa calcula as coordenadas do centroide de um objeto composto.
```

```
Clear C xs ys As
```

```
C = input('Entre com uma matriz onde cada linha possui tres elementos.\nCada linha deve receber as coordenadas do centroide (x e y) e a area de cada seção A. \n');
```

```
xs = C(:,1)';
```

Criando um vetor linha para a coordenada x de cada seção (primeira coluna de C).

```
ys = C(:,2)';
```

Criando um vetor linha para a coordenada y de cada seção (segunda coluna de C).

```
As = C(:,3)';
```

Criando um vetor linha para área de cada seção (terceira coluna de C).

```
A = sum(As);
```

Calculando a área total.

```
x = sum(As.*xs)/A;
```

Calculando as coordenadas do centróide do objeto composto.

```
y = sum(As.*ys)/A;
```

```
fprintf('As coordenadas do centroide sao: (%.f, %.f)', x, y)
```

A rotina foi salva com o nome `Capitulo4_Exemplo11`. A seguir, temos o resultado da execução da rotina na janela Command Window:

```
>> Capitulo4_Exemplo11
```

Entre com uma matriz onde cada linha possui três elementos.

Cada linha deve receber as coordenadas do centroide (x e y) e a área de cada seção.

```
[100 100 200*200
```

```
60-120/pi 200+120/pi pi*60^2/4
```

```
60+140/3 220 140*60/2
```

```
100 100/pi -pi*50^2/2
```

```
150 95 -40*150
```

```
105 145 -50*50]
```

As coordenadas do centroide são: (87.691, 140.823)

```
>>
```

Digitando as entradas na matriz C. Cada coluna possui três elementos: x, y e A de uma seção.

Problema-exemplo 4-3: Divisor de tensão

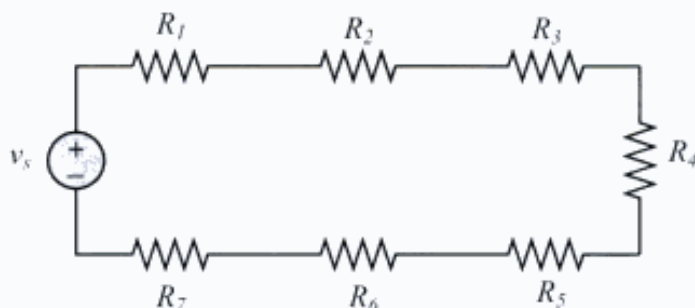
Quando resistores são conectados para formar um circuito série, a queda de tensão em cada um deles pode ser determinada pela regra do divisor de tensão:

$$v_n = \frac{R_n}{R_{eq}} v_s$$

onde v_n e R_n são a queda de tensão sobre o resistor n e a resistência desse resistor, respectivamente, $R_{eq} = \sum R_n$ é a resistência equivalente ou total e v_s é a tensão da fonte. A potência dissipada em cada resistor é dada por:

$$P_n = \frac{R_n}{R_{eq}^2} v_s^2$$

A figura abaixo mostra, por exemplo, um circuito com sete resistores conectados em série.



Escreva um programa em uma rotina que calcula a queda de tensão e a potência dissipada em cada resistor de um circuito série com 7 resistores. Executando a rotina, o usuário deve ser solicitado a digitar em um vetor o valor da fonte de tensão e a resistência de cada resistor. O programa exibirá uma tabela com as resistências listadas na primeira coluna, a queda de tensão na segunda coluna e a potência dissipada na terceira coluna. Após a tabela, o programa deve exibir a corrente no circuito e a potência total dissipada. Teste a rotina com os seguintes valores de v_s e R_s :

$$v_s = 24\text{V}, R_1 = 20\Omega, R_2 = 14\Omega, R_3 = 12\Omega, R_4 = 18\Omega, R_5 = 8\Omega, R_6 = 15\Omega \text{ e } R_7 = 10\Omega$$

Solução

Uma rotina escrita para solucionar esse programa é mostrada a seguir.

```
% Este programa calcula a queda de tensao em cada resistor num circuito serie.
vs = input('Entre com o valor da fonte de tensao');
Rn = input('Entre com os valores dos resistores como elementos de um vetor linha.\n');
Req = sum(Rn);
vn = Rn*vs/Req;
Pn = Rn*vs^2/Req^2;
i = vs/Req;
Ptotal = vs*i;
Tabela = [Rn',vn',Pn'];
disp('')
disp(' Resistencia      Tensao      Potencia')
disp(' (Ohms)          (Volts)      (Watts)')
disp('')
disp(Tabela)
disp('')
fprintf('A corrente no circuito e %.3f Amps', i)
fprintf('\nA potencia total dissipada no circuito e %.3f Watts', Ptotal)
```

Calcula a resistência equivalente.

Aplica a regra do divisor de tensão.

Determina a potência em cada resistor.

Calcula a corrente no circuito.

Calcula a potência total no circuito.

Cria uma variável denominada `tabela` com os vetores `Rn`, `vn` e `Pn` como colunas.

Exibe as unidades de medidas das colunas.

Exibe o rótulo das colunas.

Imprime uma linha em branco.

Exibe os dados da tabela.

Executando a rotina (Capitulo4_Exemplo12) o resultado será:

```
>> Capitulo4_Exemplo12
```

Nome da rotina.

Entre com o valor da fonte de tensao 24

Inicializa a variável fonte de tensão.

Entre com os valores dos resistores como elementos de um vetor linha.

```
[20 14 12 18 8 15 10]
```

Inicializa o vetor Rn com os valores dos resistores.

Resistencia (Ohms)	Tensao (Volts)	Potencia (Watts)
20.0000	4.9485	1.2244
14.0000	3.4639	0.8571
12.0000	2.9691	0.7346
18.0000	4.4536	1.1019
8.0000	1.9794	0.4897
15.0000	3.7113	0.9183
10.0000	2.4742	0.6122

A corrente no circuito e 0.247 Amps.

A potencia total dissipada no circuito e 5.938 Watts.

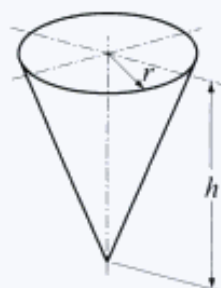
4.9 PROBLEMAS

Nota: assim como os problemas a seguir, os problemas ao final do Capítulo 3 podem ser resolvidos por meio de rotinas.

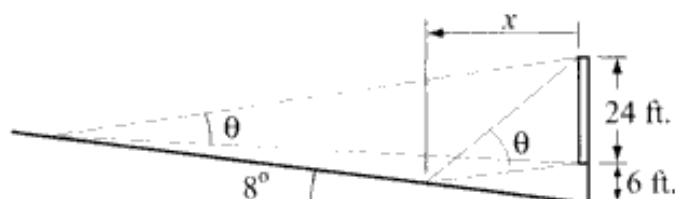
Resolva os seguintes problemas escrevendo em primeiro lugar a rotina, executando-a depois na linha do *prompt* na janela Command Window.

1. Um cone de papel foi construído de modo a possuir um volume de 250cm^3 . Determine o raio r da base e a área da superfície (S) do cone, considerando as seguintes alturas h do cone: 5, 6, 7, 8 e 9 cm. O volume V e a área da superfície do cone são dados por:

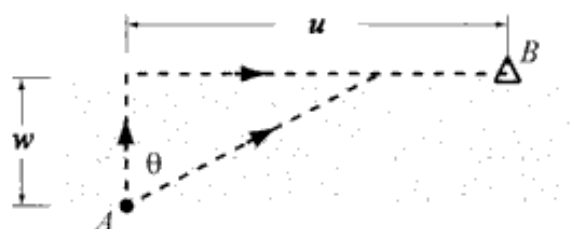
$$V = \frac{1}{3}\pi r^2 h \quad S = \pi r \sqrt{r^2 + h^2}$$



2. Numa sala de cinema, o ângulo θ de observação da platéia depende da distância x de cada espectador em relação à tela. A figura mostra as dimensões de uma sala de cinema. Determine o ângulo θ em graus para espectadores distantes 9m; 13.5m; 18m; 22.5m e 27m da tela.



3. A população de um determinado país é 50 milhões e a expectativa é de que ela duplique em 20 anos. Determine a população em 5, 10 e 15 anos, partindo da população inicial, declarando um vetor t com três elementos e realizando cálculos escalares, isto é, elemento por elemento. O crescimento populacional segue a equação $P = P_0 2^{t/d}$, onde P é a população no tempo t , P_0 é a população em $t = 0$ e d é o tempo necessário para que a população duplique.
4. Um banhista precisa cruzar uma certa faixa de areia de praia, partindo do ponto A em direção ao acampamento no ponto B . Ele pode fazer isso atravessando perpendicularmente a faixa de areia e andando o restante ao longo de uma trilha ou cruzando-a em um ângulo θ , até um certo ponto, e andando o restante ao longo da trilha. Considerando-se que o banhista caminha a 3.5km/h na areia e a 5km/h na trilha, determine o tempo para que ele chegue ao acampamento para $\theta = 0, 10, 20, 30, 40, 50$ e 60 graus.



As distâncias w e u são 4,5km e 14km, respectivamente. Escreva um programa em uma rotina para resolver o problema. Declare todas as variáveis dentro da rotina. Mostre os resultados em duas colunas de uma tabela, onde a primeira coluna representa o ângulo θ e a segunda coluna corresponde ao tempo t .

5. Escreva uma rotina que determina o saldo de uma conta de poupança no final de cada ano. Considere que a conta não será movimentada durante 10 anos, que o investimento inicial é \$1,000.00 e que a taxa de juros anual é 6.5%. Apresente a informação numa tabela.

Supondo que o investimento inicial seja A e a taxa de juros r , o saldo da conta B após n anos é dado:

$$B = A \left(1 + \frac{r}{100} \right)^n$$

6. A velocidade v e a distância d , em função do tempo, de um carro que acelera uniformemente a partir do repouso são dadas, respectivamente, por:

$$v(t) = at \quad \text{e} \quad d(t) = \frac{1}{2}at^2$$

sendo a a aceleração do carro. Se a aceleração do carro é $a = 1.55 \text{ m/s}^2$, determine d e v do carro para cada segundo, após os primeiros 10s. Apresente os resultados numa tabela de três colunas, onde a primeira é o tempo em segundos (s), a segunda é a distância em metros (m) e a terceira é a velocidade em quilômetro por hora (km/h).

7. Quando resistores são conectados em paralelo num circuito elétrico, a corrente que flui através de cada resistor é:

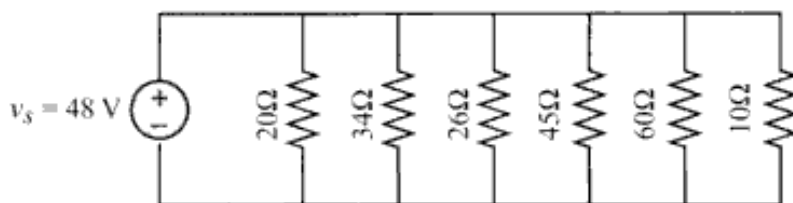
$$i_n = \frac{v_s}{R_n}$$

onde i_n e R_n são a corrente e a resistência do resistor n , respectivamente, e v_s é a tensão da fonte. A resistência equivalente (R_{eq}) pode ser determinada a partir da equação:

$$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2} + \dots + \frac{1}{R_n}$$

A fonte de corrente é dada por $i_s = v_s / R_{eq}$ e a potência dissipada em cada resistor é dada por $P_n = v_s i_n$.

Escreva um programa numa rotina que calcule a corrente e a potência em cada resistor do circuito paralelo. Ao executar a rotina, o usuário será solicitado primeiro a entrar em um vetor com o valor da fonte de tensão e, em seguida, com os valores de resistência dos resistores. O programa deve apresentar uma tabela com as resistências listadas na primeira coluna, a corrente na segunda colu-

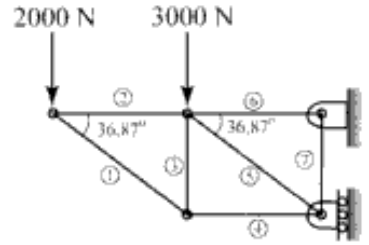


na e a potência dissipada na terceira coluna. Após a tabela, o programa deve mostrar a corrente e a potência total da fonte.

Teste a rotina no circuito apresentado a seguir:

8. O gráfico da função $f(x) = ax^3 + bx^2 + cx + d$ passa pelos pontos $(-2; -3,4)$, $(-0.5; 5.525)$, $(1; 16.7)$ e $(2.5; 70.625)$. Determine as constantes a , b , c e d . (Sugestão: escreva um sistema de quatro equações e quatro incógnitas e use o MATLAB para resolvê-lo.)

9. Treliza é o nome de uma estrutura feita de várias vigas unidas pelas extremidades (em junções ou nós, considerados pontuais nos cálculos teóricos). No caso da treliza mostrada na figura, as forças que agem nas sete vigas são determinadas resolvendo-se o sistema de equações a seguir:



$$F_1 \sin(36.87^\circ) = -2000$$

$$F_1 \cos(36.87^\circ) + F_2 = 0$$

$$F_3 + F_1 \sin(36.87^\circ) = 0$$

$$F_4 - F_1 \cos(36.87^\circ) = 0$$

$$-F_3 - F_5 \sin(36.87^\circ) = 3000$$

$$F_6 + F_5 \cos(36.87^\circ) - F_2 = 0$$

$$F_5 \sin(36.87^\circ) + F_7 = 0$$

Escreva as equações na forma matricial e use o MATLAB para determinar as forças nas vigas. Uma força positiva indica força de tensão e uma força negativa indica força de compressão. Apresente os resultados numa tabela.

CAPÍTULO 5

Gráficos Bidimensionais

Gráficos são ferramentas poderosas quando se deseja interpretar visualmente os resultados. Um recurso especialmente interessante nas engenharias e demais ciências exatas, onde o MATLAB encontra as maiores aplicações. O MATLAB possui muitos comandos passíveis de serem utilizados para criar diferentes tipos de gráficos. Fazem parte dessa lista comandos gráficos padrão com escalas lineares e logarítmicas (monolog ou bilog); gráficos em barras e escadas; polares; efeito visual 3-D, em pizza e muito mais. Além disso, os gráficos podem ser formatados atendendo vários aspectos: tipo de linha (cheia, pontilhada, etc), a cor e a espessura são facilmente especificadas; é possível também adicionar marcadores de linhas e *grids* e, é claro, títulos e comentários estão entre as opções básicas de formatação de gráficos. É possível exibir vários gráficos em uma única saída e muitos gráficos podem ser colocados em uma mesma janela. Quando uma saída exibe vários gráficos e/ou dados simultaneamente, é possível adicionar legendas para identificá-los.

Este capítulo descreve como usar o MATLAB para criar e formatar gráficos bidimensionais (2-D). Os gráficos tridimensionais (3-D) serão tratados separadamente no Capítulo 9. Um gráfico 2-D típico, gerado pelo MATLAB, é mostrado na Figura 5-1. A figura mostra duas curvas onde se estuda o comportamento teórico e experimental da intensidade de luz em função da distância. A curva experimental foi construída a partir de medidas experimentais e a curva teórica traz o comportamento previsto pelo modelo teórico. Foram especificados dois eixos lineares nos gráficos e diferentes tipos de linha ajudam a diferenciá-los. A curva teórica é a mostrada em linha cheia e os dados experimentais são conectados com uma linha tracejada. Cada ponto plotado experimentalmente no gráfico foi marcado com um círculo. A linha tracejada ligando os pontos experimentais é exibida em vermelho na janela Figure Window. O gráfico da Figura 5-1 foi formatado de modo a possuir título, rótulo dos eixos, legenda, marcadores e uma caixa de texto.

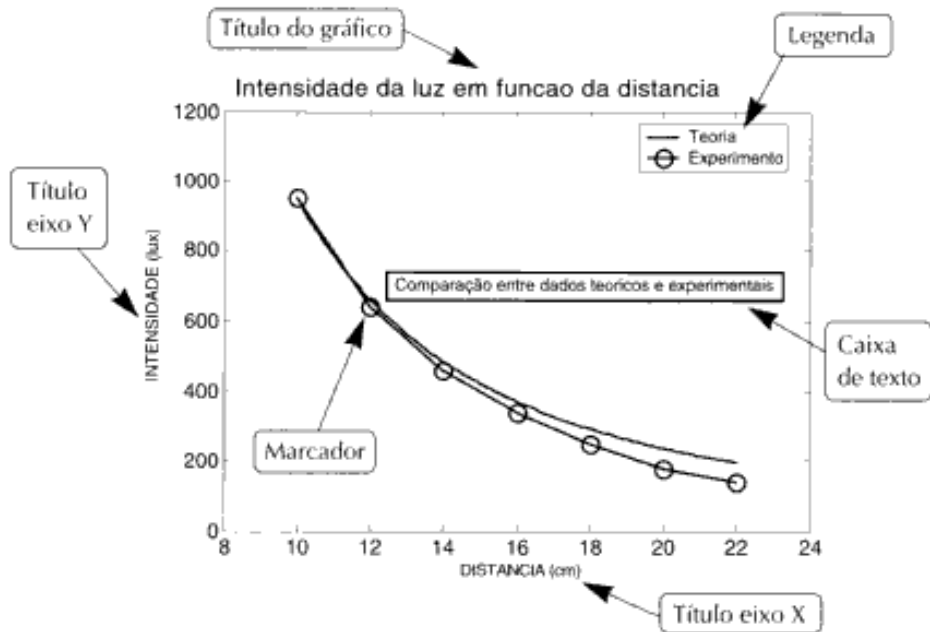


FIGURA 5-1 Exemplo de gráfico 2-D formatado.

5.1 O COMANDO `plot`

O comando `plot` é utilizado para criação de gráficos bidimensionais. A forma (sintaxe) mais simples desse comando é:

`plot (x,y)`
 ↑ ↑
 Vetor Vetor

Cada um dos argumentos `x` e `y` deve ser um vetor (arranjo unidimensional). Além disso, os dois vetores **devem possuir** a mesma quantidade de elementos. Ao executar o comando `plot`, é gerada uma saída na janela Figure Window. Caso a janela Figure Window esteja fechada, ela será aberta automaticamente após a execução do comando. O gráfico padrão produzido pelo comando `plot` traz uma única curva, sendo os valores de `x` são dispostos na abscissa (eixo horizontal) e os valores de `y` na ordenada (eixo vertical). A curva é construída a partir de segmentos de reta que ligam os pontos cujas coordenadas são definidas pelos elementos dos vetores `x` e `y`. Evidentemente, os vetores podem ter qualquer nome. Por último, o primeiro vetor digitado no argumento de `plot` é usado como eixo horizontal e, o segundo, como eixo vertical.

O gráfico gerado possui os eixos escalonados linearmente em uma faixa-padrão. Por exemplo, se um vetor `x` possui os elementos 1, 2, 3, 4, 7, 7.5, 8, 10, e um vetor `y` os elementos 2, 6.5, 7, 7, 5.5, 4, 6, 8, o gráfico `y versus x` pode ser criado digitando-se, tão-somente, os seguintes comandos na janela Command Window:


```
>> x = [1 2 3 5 7 7.5 8 10];
>> y = [2 6.5 7.7 5.5 4 6 8];
>> plot(x,y)
```



Uma vez executado o comando `plot`, a janela Figure Window é inicializada e o gráfico exibido é semelhante à Figura 5-2:

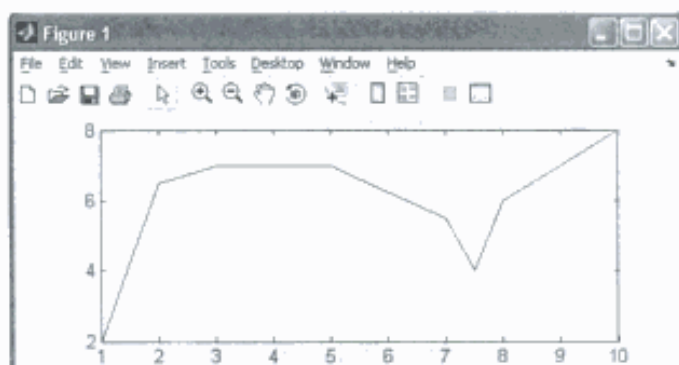


FIGURA 5-2 Um gráfico na janela Figure Window.

A cor-padrão (*default*) do gráfico na tela é o azul.

Entretanto, é possível adicionar outros argumentos ao comando `plot` para permitir especificar a cor, o estilo da linha, a cor e o estilo dos marcadores. Com essas novas opções, a sintaxe-padrão do comando `plot` é:

```
plot (x, y, 'Especificadores de linha', 'Propriedade', Valor_propriedade)
```

Vetor

Vetor

(Opcional) Especifica o tipo e a cor da linha e os tipos de marcadores.

(Opcional) Propriedades adicionais usadas para definir a espessura da linha, o tamanho do marcador e da borda e a cor de preenchimento.

Especificadores de linha:

Esse campo é opcional e pode ser utilizado para definir o estilo, a cor da linha e o tipo de marcador (se necessários). Os especificadores para estilos de linhas são:

Estilo de linha	Especificador
Sólida (padrão)	-
Tracejada	--

Estilo de linha	Especificador
Pontilhada	:
Traço-ponto	-.

Os especificadores de cores são:

Cor da linha	Especificador
Vermelha	r (<i>red</i>)
Verde	g (<i>green</i>)
Azul	b (<i>blue</i>)
Ciano	c

Cor da linha	Especificador
Magenta	m
Amarelo	y (<i>yellow</i>)
Preto	k (<i>black</i>)
Branco	w (<i>white</i>)

Especificadores de tipo de marcador são:

Tipo de marcador	Especificador
Sinal	+
Círculo	O
Asterisco	*
Ponto	.

Tipo de marcador	Especificador
Quadrado	s
Diamante	d
Estrela de 5 pontas	p
Estrela de 6 pontas	h

Observações quanto ao uso de especificadores:

- Os especificadores são digitados como *strings* dentro do argumento do comando `plot`.
- Dentro da *string*, os especificadores podem ser digitados em qualquer ordem.
- Os especificadores são opcionais. Significa que nenhum, um, dois ou todos os especificadores podem ser incluídos em um comando.

Exemplos:

<code>plot (x, y)</code>	Uma linha cheia na cor azul interliga os pontos sem marcadores (<i>default</i>).
<code>plot (x, y, 'r')</code>	Os pontos são interligados por uma linha cheia em vermelho.
<code>plot (x, y, '-y')</code>	Uma linha amarela tracejada interliga os pontos.
<code>plot (x, y, '*')</code>	Os pontos são marcados com um * (não há linhas interligando os pontos).
<code>plot (x, y, 'g:d')</code>	Uma linha pontilhada verde interliga os pontos marcados com losangos.

Propriedades e os respectivos valores:

As propriedades também são adicionais e podem ser utilizadas na especificação da espessura da linha, do tamanho e das cores dos marcadores de linhas e para preenchimento. As propriedades são digitadas como *string*, são separadas por vírgula e sucedidas do valor da propriedade. Tudo isso no argumento do comando `plot`.

A seguir são mostradas quatro propriedades com os respectivos valores possíveis:

Nome da propriedade	Descrição	Valores possíveis
LineWidth (ou linewidth)	Especifica a largura da linha.	Um número em quantidade de pontos (<i>default</i> 0.5).
MarkerSize (ou markersize)	Especifica o tamanho do marcador.	Um número em unidade de pontos.
MarkerEdge-Color (ou markeredge-color)	Especifica a cor do marcador ou a cor da borda da linha que delimita o marcador.	Especificadores de cor da tabela anterior digitados como <i>string</i> .
MarkerFace-Color (ou markerface-color)	Especifica a cor de preenchimento para marcadores.	Especificadores de cor da tabela anterior digitados como <i>string</i> .

Por exemplo, o comando

```
plot(x,y, '-mo', 'LineWidth', 2, 'markersize', 12, 'MarkerEdge-Color', 'g', 'markerfacecolor', 'y')
```

cria um gráfico interligando os pontos com uma linha magenta cheia, adicionando marcadores na forma de círculos em cada ponto. A linha tem espessura de dois pontos e o círculo marcador tem tamanho de 12 pontos. Por último, os marcadores têm a linha de borda verde e são preenchidos de amarelo.

Uma observação quanto aos especificadores de linha e às propriedades:

Os três especificadores de linha (estilo, cor da linha e tipo de marcador) podem ser atribuídos como um argumento nas propriedades, seguido por um argumento no campo *Valores_das_propriedades*. As propriedades para os especificadores de linha são:

Especificador	Propriedade	Valores possíveis
Line Style	linestyle ou LineStyle	Especificador de estilo de linha citado na primeira tabela.
Line Color	color ou Color	Especificadores de cor citados na segunda tabela.
Marker	marker ou Marker	Especificadores de tipo de marcador citados na terceira tabela.

Como qualquer comando do MATLAB, o comando `plot` pode ser digitado na linha do *prompt* da janela Command Window ou ser incluído em uma rotina. Também é possível utilizá-lo na declaração de uma função (Capítulo 6). Cabe ressaltar que, antes de o comando `plot` ser executado, os vetores *x* e *y* devem ter sido declarados e inicializados. Esse as-

pecto foi explicado no Capítulo 2, atribuindo-se diretamente valores aos vetores na linha do *prompt* ou indiretamente por meio de operações matemáticas. As duas próximas subseções mostram exemplos simples de gráficos construídos a partir do comando `plot`.

5.1.1 Gráfico de uma tabela de dados

Nesse tipo de gráfico, é necessário dispor os dados numa tabela e atribuí-los a vetores antes que o comando `plot` seja executado. Por exemplo, a tabela abaixo possui dados relativos às vendas de uma empresa de 1988 a 1994.

Ano	1988	1989	1990	1991	1992	1993	1994
Vendas (milhões)	8	12	20	22	18	24	27

Para plotar esses dados, a lista de anos é atribuída ao vetor `ano` e as vendas correspondentes são atribuídas ao vetor `vendas`. A sequência de comandos na janela Command Window é mostrada a seguir:

```
>> ano=[1988:1:1994];
>> vendas=[8 12 20 22 18 24 27];
>> plot(ano,vendas,'-r*', 'linewidth',2,'markersize',12)
>>
```

Especificadores de linha: linha vermelha tracejada e marcador tipo asterisco.

Nome e valor da propriedade: a largura da linha é 2 pontos e o tamanho dos marcadores é 12 pontos.

Uma vez executado o comando `plot`, a janela Figure Window exibe o gráfico referente aos dados (veja Figura 5-3). O gráfico aparece em vermelho na tela, conforme designação do especificador de linha.

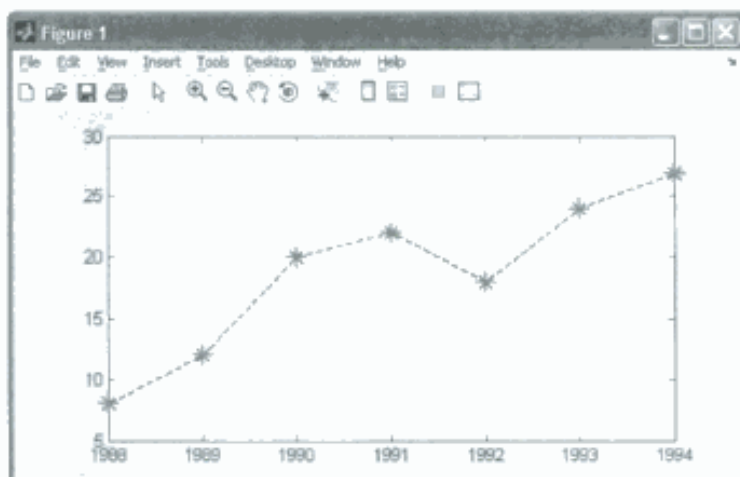


FIGURA 5-3 Um gráfico representativo dos dados da tabela.

5.1.2 Gráfico de uma função

É bastante freqüente estudarmos o comportamento gráfico de uma função. No MATLAB, a geração de gráficos se dá de duas formas diferentes: comandos `plot` e `fplot`. A utilização do comando `plot` para esse propósito está sendo estudada aqui. Já o comando `fplot` é o objeto de estudo da próxima seção.

Uma função $y = f(x)$ pode ser desenhada pelo comando `plot` atribuindo a um vetor os possíveis valores do domínio da variável independente x . Em seguida, o vetor y é gerado a partir dos valores assumidos por $f(x)$, realizando operações escalares (elemento por elemento) no vetor inicializado com os valores de x (veja Capítulo 3). Uma vez que os dois vetores tenham sido criados, o comando `plot` pode ser executado.

Por exemplo, vamos plotar o gráfico da função $y = 3.5^{-0.5x} \cos(6x)$, $-2 \leq x \leq 4$. Um programa que calcula e desenha esse gráfico é mostrado na rotina a seguir:

```
% Uma rotina para gerar o grafico da funcao: 3.5.^(-0.5*x).*cos(6*x).
```

```
x = [-2:0.01:4];
```

Declara vetor x com o domínio da função.

```
y = 3.5.^(-0.5*x).*cos(6*x);
```

Cria vetor y com os valores da função.

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

Plota y em função de x.

Rodando a rotina na linha do *prompt* do Command Window, o resultado é o que mostra a Figura 5-4.

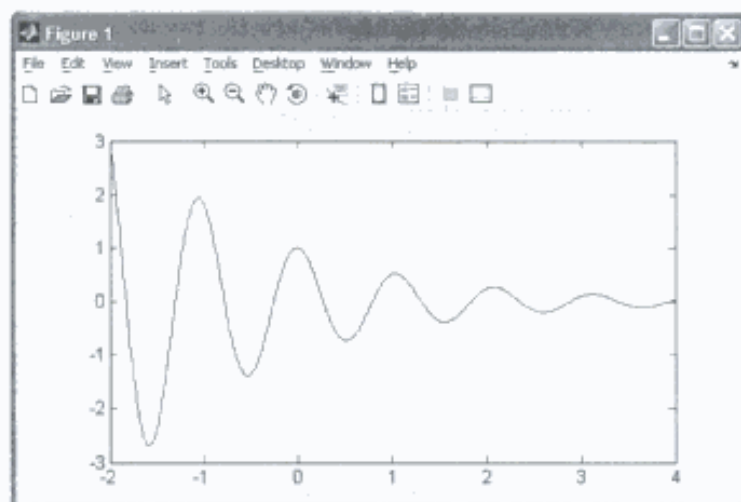


FIGURA 5-4 Gráfico da função $y = 3.5^{-0.5x} \cos(6x)$.

Visto que um gráfico no MATLAB é traçado com segmentos de linha ligando os pontos adjacentes, para obter a precisão desejada basta determinar o incremento entre os elementos do domínio (vetor x). Funções que variam muito rapidamente requerem incrementos menores. No exemplo, um incremento de 0.01 produziu o gráfico mostrado

na Figura 5-4. Entretanto, se a mesma função for desenhada no mesmo intervalo de domínio, com incrementos maiores (por exemplo, 0.3), a precisão requerida oculta detalhes do gráfico da função (Figura 5-5). Perceba também que na Figura 5-4 o gráfico é mostrado no ambiente em que foi gerado, isto é, na janela *Figure Window*. A Figura 5-5 traz somente o gráfico. Um gráfico pode ser copiado da janela *Figure Window*. Para isso, no menu **Edit**, selecione a opção **Copy Figure**. Em seguida, basta colar em outras aplicações de interesse.

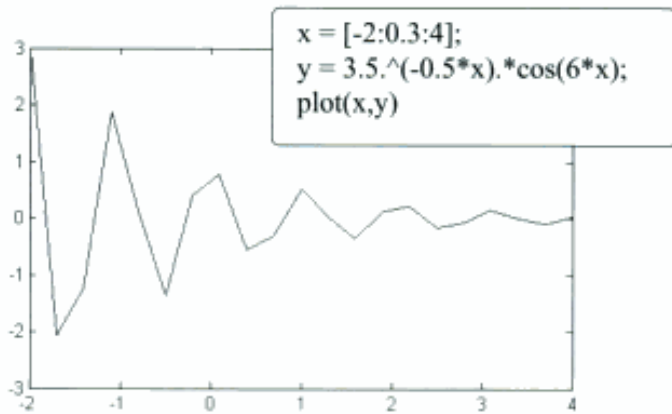


FIGURA 5-5 Gráfico de função $y = 3.5^{-0.5x} \cos(6x)$ com grandes incrementos.

5.2 O COMANDO `fplot`

O comando `fplot` esboça uma função $y = f(x)$ entre os limites especificados. A forma geral do comando é:

```
fplot ('funcao', limites, especificadores de linha)
```

Função a ser plotada.

Os valores de x (domínio da função) e, opcionalmente, os limites do eixo y .

Especificadores que definem o tipo, cor da linha e marcadores (opcional).

'funcao': a função deve ser digitada diretamente como uma *string* dentro do comando. Por exemplo, se a função que se deseja plotar é $f(x) = 8x^2 + 5\cos(x)$, devemos digitá-la como `'8*x^2+5*cos(x)'`. A função desejada pode incluir funções nativas do MATLAB e/ou funções criadas pelo usuário (veja Capítulo 6).

- Não importa o nome da variável independente digitado dentro do argumento da função. Por exemplo, a função anterior poderia ser `'8*z^2+5*cos(z)'` ou `'8*t^2+5*cos(t)'`.
- A função não pode incluir no argumento variáveis já declaradas no programa. Por exemplo, na função acima não é possível atribuir o número 8 a uma variável `e`, depois, usá-la no argumento do comando `fplot`.

limites: os limites devem ser introduzidos como um vetor de dois elementos que especifica o intervalo do domínio x [$x_{\min}$, $x_{\max}$] ou um vetor com quatro elementos especificando o intervalo do domínio x e do contradomínio y onde se deseja vislumbrar o gráfico da função [$x_{\min}$, $x_{\max}$, $y_{\min}$, $y_{\max}$].

especificadores de linha: os especificadores de linha são os mesmos descritos para o comando `plot`. Por exemplo, um gráfico da função $y = x^2 + 4\sin(2x) - 1$, $-3 \leq x \leq 3$, pode ser criado por meio do comando `fplot` digitando

```
>> fplot('x^2+4*sin(2*x)-1',[-3.3])
```

na linha do *prompt* da janela Command Window. A Figura 5-6 mostra o resultado desse comando.

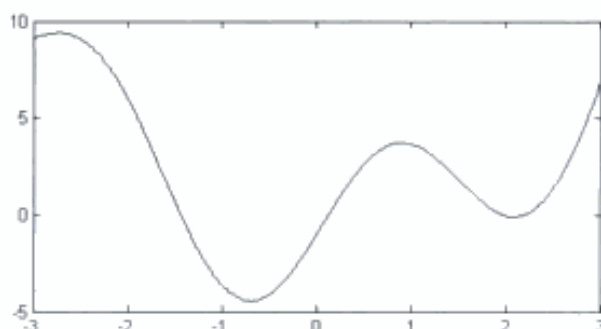


FIGURA 5-6 Um gráfico da função $y = x^2 + 4\sin(2x) - 1$.

5.3 PLOTANDO GRÁFICOS MÚLTIPLOS EM UMA MESMA SAÍDA

Muitas vezes é necessário reunir vários gráficos diferentes em uma mesma saída. A Figura 5-1 ilustra um caso. Existem três maneiras de plotar muitos gráficos juntos. O primeiro e mais elementar é por meio do comando `plot`. Os outros dois usam os comandos `hold on` (`hold off`) e o comando `line`.

5.3.1 Usando o comando `plot`

Dois ou mais gráficos podem ser colocados em uma mesma saída digitando-se pares de vetores nos argumentos do comando `plot`. O comando

```
plot (x, y, u, v, t, h)
```

cria três gráficos y versus x , v versus u e h versus t , todos na mesma saída. Os vetores em cada par devem possuir o mesmo tamanho. O MATLAB gera gráficos automaticamente em três cores diferentes, de modo que eles possam ser diferenciados. Também é possível adicionar especificadores de linha após cada par de variáveis. Por exemplo, o comando

```
plot (x, y, '-b', u, v, '--r', t, h, 'g:')
```


traça o gráfico y versus x com uma linha cheia azul, v versus u com uma linha tracejada vermelha, h versus t com uma linha pontilhada verde.

Problema-exemplo 5-1: Plotando uma função e as suas duas primeiras derivadas

Trace o gráfico da função $y = 3x^3 - 26x + 10$, juntamente com a primeira e segunda derivadas dessa função, no intervalo $-2 \leq x \leq 4$, em uma mesma saída.

Solução

A derivada primeira da função é: $y' = 9x^2 - 26$.

A derivada segunda da função é: $y'' = 18x$.

Uma rotina que, dado um vetor x , calcula o valor de y , y' e y'' é:

```
x = [-2:0.01:4];
```

Declara o vetor x contendo o domínio da função.

```
y = 3*x.^3-26*x+10;
```

Declara o vetor y contendo os valores da função (contradomínio).

```
yd = 9*x.^2 - 26;
```

Declara o vetor yd contendo os valores da derivada primeira de y .

```
ydd = 18*x;
```

Declara o vetor ydd contendo os valores da derivada segunda de y .

```
plot(x,y,'-b',x,yd,'--r',x,ydd,':k')
```

Gera três gráficos na mesma figura: y vs. x , yd vs. x e ydd vs. x .

Os gráficos gerados são mostrados na Figura 5-7.

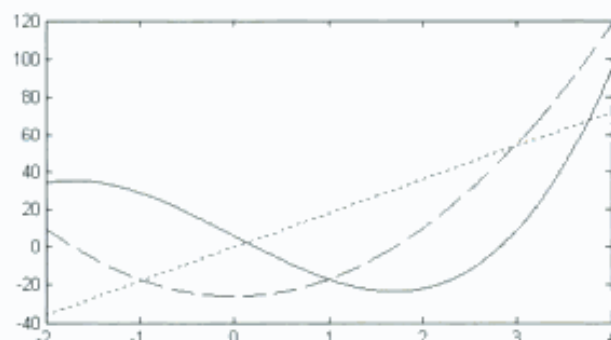


FIGURA 5-7 Um gráfico da função $y = 3x^3 - 26x + 10$ e das derivadas y' e y'' .

5.3.2 Usando os comandos `hold on` e `hold off`

Plotar vários gráficos simultaneamente, usando os comandos `hold on` e `hold off`, é feito da seguinte maneira: primeiramente, um gráfico é traçado. Depois, o comando `hold on` é digitado e executado. Isso mantém a janela Figure Window aberta exibindo o primeiro gráfico da série, incluindo as propriedades dos eixos e a formatação (veja Seção 5.4). Os gráficos adicionais são colocados na mesma saída, um a um, por meio do comando `plot`. Cada comando `plot` adiciona exatamente um gráfico na saída. Quando se deseja encerrar o processo de anexação de gráficos, digita-se o comando `hold off` e o MATLAB retorna ao modo *default*, onde um próximo comando `plot` apaga todos os resultados de saída e as propriedades dos eixos.

Por exemplo, abaixo pode ser observada a solução do Problema-exemplo 5-1 com a utilização dos comandos `hold on` e `hold off` na linha do *prompt* da janela Command Window:

```
x = [-2:0.01:4];
y = 3*x.^3 - 26*x + 6;
yd = 9*x.^2 - 26;
ydd = 18*x;
plot(x,y, '-b')
hold on
plot(x,yd, '--r')
plot(x,ydd, 'k')
hold off
plot(yd,ydd, '-y')
```

O primeiro gráfico é plotado.

Mais dois gráficos são anexados à figura.

A janela Figure Window retorna ao modo default.

5.3.3 Usando o comando `line`

Através do comando `line`, as linhas esboçadas em um gráfico podem ser modificadas/adicionadas. A sintaxe geral desse comando é:

```
line (x, y, 'Propriedade', Valor_propriedade)
```

(Opcional) Propriedades e os respectivos valores, usados para especificar o estilo, a cor e largura de linha; tipo, tamanho, borda e cor de preenchimento do marcador.

A sintaxe do comando `line` é quase a mesma do comando `plot` (veja Seção 5.1). A diferença essencial está na inexistência do campo Especificadores de Linha. Entretanto, o estilo/cor da linha e o marcador podem ser especificados nas Propriedades e nos valores característicos da mesma. As propriedades são opcionais e, caso não sejam especificadas, o MATLAB utiliza propriedades e valores-padrão (*default*) na estrutura do comando. Por exemplo, o comando

```
line(x, y, 'linestyle', '--', 'color', 'r', 'marker', 'o')
```

adicionará uma linha vermelha tracejada, com marcadores circulares, a um gráfico na saída.

A maior diferença entre os dois comandos é que o comando `plot` é chamado com o intuito de plotar um novo gráfico, enquanto que o comando `line` adiciona linhas a gráficos já plotados na saída. Um modo rápido de se traçar gráficos múltiplos é o seguinte: primeiro é digitado um comando `plot`. Em seguida, os comandos `line` são digitados (um a um) e executados para adicionar gráficos logo após o respectivo comando `plot`. (Se um comando `line` for digitado antes de um comando `plot`, uma mensagem de erro será mostrada no *prompt*.)

Outro modo de solucionar o Problema-exemplo 5-1, para produzir os gráficos da Figura 5-7, inclui uma sequência de comandos `plot` e `line`, conforme sugere a rotina abaixo:

```
>> x=[-2:0.01:4];
>> y = 3*x.^3 - 26*x + 6;
>> yd = 9*x.^2 - 26;
>> ydd = 18*x;
>> plot(x,y,'LineStyle','-',color','b')
>> line(x,yd,'LineStyle','-',color','r')
>> line(x,ydd,'LineStyle','.',color','k')
```

5.4 FORMATANDO UM GRÁFICO

Os comandos `plot` e `fplot` foram utilizados da maneira mais simples possível. Desse modo, eles permitem, tão-somente, a criação de gráficos. Entretanto, é usual formatar um gráfico de forma a exibir informações sobre as escalas e informações específicas, além do gráfico em si. É possível incluir títulos para os eixos, título do gráfico, legenda, *grids*, faixa de valores para as escalas e caixas de texto.

A formatação de gráficos pode acontecer em dois momentos: após os comandos `plot`/`fplot` ou usando, interativamente, o editor das propriedades dos eixos (menu Edit) na janela Figure Window. O primeiro modo é bastante útil quando um comando `plot` (`fplot`) faz parte de uma rotina. Ao se utilizar comandos de formatação em uma rotina, toda vez que ela for executada, um gráfico formatado resulta na saída da janela Figure Window. Por outro lado, a formatação pode ocorrer na janela Figure Window, no menu Edit, mas deverá ser repetida toda vez que um gráfico for gerado na saída.

5.4.1 Formatando um gráfico de comandos

Os comandos de formatação devem ser digitados e executados após os comandos `plot` e `fplot`. As diversas possibilidades de formatação de gráficos são:

Os comandos `xlabel` e `ylabel`:

Os comandos `xlabel` e `ylabel` servem para rotular os eixos `x` e `y` da figura. O formato geral desses comandos é:

```
xlabel('string de texto')
ylabel('string de texto')
```

O comando title:

Para adicionar um título a um gráfico, o comando

```
title('string de texto')
```

insere um texto no topo da figura na forma de título.

O comando text:

Uma caixa de texto pode ser inserida em um gráfico por meio dos comandos `text` ou `gtext`:

```
text(x,y,'string de texto')
gtext('string de texto')
```

O comando `text` insere uma caixa de texto de tal forma que o primeiro caractere é inserido na posição indicada pelas coordenadas `x`, `y` (de acordo com os eixos da figura). O comando `gtext` insere uma caixa de texto na posição especificada pelo usuário. Executando o comando `gtext`, a janela `Figure Window` é aberta e o usuário é solicitado a especificar a posição da caixa com o *mouse*.

O comando legend:

O comando `legend` adiciona legenda ao gráfico. Uma legenda é útil para fazer distinção entre diversos tipos de gráficos que forem plotados na saída. Através da especificação de cor da linha atribuída a cada um é adicionado um texto (*string* – indicada pelo usuário) ao lado da linha que designa o gráfico. O formato geral do comando é:

```
legend('string1', 'string2', . . . . ., pos)
```

As *strings* especificam os textos a serem colocados na legenda. Elas devem entrar na ordem em que os gráficos devem ser criados para legendar um ou mais gráficos. O campo `pos` especifica um número opcional onde a legenda do gráfico deve ser inserida. Essas opções podem assumir os seguintes valores:

- `pos = - 1` Insere a legenda à direita da figura, fora do contorno delimitado pelos eixos.
- `pos = 0` Insere a legenda dentro da figura, numa posição que interfira o mínimo no gráfico.
- `pos = 1` Insere a legenda no canto direito superior do gráfico (*default*).
- `pos = 2` Insere a legenda no canto esquerdo superior do gráfico.
- `pos = 3` Insere a legenda no canto esquerdo inferior do gráfico.
- `pos = 4` Insere a legenda no canto direito inferior do gráfico.

Formatando o texto nos comandos xlabel, ylabel, title, text e legend:

Os textos inseridos pelos comandos de formatação de gráficos também podem ser formatados. A formatação é utilizada para definir o tipo, o tamanho, a posição (subscrito ou sobrescrito), o estilo (itálico, negrito, etc.) da fonte; a cor dos caracteres e do segundo

plano e muitos outros detalhes. Os comandos de formatação mais comuns são descritos abaixo. Uma explanação completa dessas características de formatação está disponível na janela Help Window, opções Text e Text Properties. O modo-padrão de formatar um texto é adicionar modificadores dentro da *string* de texto ou adicionar Propriedades, com os respectivos valores das mesmas, nos argumentos das *strings*.

Os modificadores são caracteres especiais inseridos dentro da *string*. Os modificadores mais comuns incluem:

Modificador	Efeito	Modificador	Efeito
<code>\bf</code>	Fonte em negrito	<code>\nome_fonte</code>	Especifica fonte a ser utilizada.
<code>\it</code>	Estilo itálico	<code>{fontname}</code>	
<code>\rm</code>	Fonte normal	<code>\tamanho_fonte</code>	Especifica o tamanho da fonte.
		<code>{fontsize}</code>	

Esses modificadores agem no texto, partindo do ponto onde são inseridos, até o fim da *string*. É possível modificar uma linha da *string* de texto digitando o modificador e o texto a ser modificado entre chaves `{ }`.

Subscrito e sobrescrito:

Digitando-se `_` (sublinhar ou *underscore*) ou `^` antes de um caractere, esse caractere aparecerá como subscrito ou sobrescrito, respectivamente. Uma cadeia de caracteres pode ser inserida como subscrito ou sobrescrito. Basta digitar os caracteres dentro de chaves `{ }`, após `_` ou `^`.

Caracteres gregos:

Caracteres gregos podem ser incluídos em um texto digitando-se `\nome_letra_grega` dentro da *string*. Para exibir letras gregas minúsculas, o nome da letra deve ser digitado todo em letras minúsculas. Para escolher uma letra grega maiúscula, apenas a inicial da letra deve aparecer maiúscula. Por exemplo:

Caracteres na <i>string</i>	Letra grega	Caracteres na <i>string</i>	Letra grega
<code>\alpha</code>	α	<code>\Phi</code>	Φ
<code>\beta</code>	β	<code>\Delta</code>	Δ
<code>\gamma</code>	γ	<code>\Gamma</code>	Γ
<code>\theta</code>	θ	<code>\Lambda</code>	Λ
<code>\pi</code>	π	<code>\Omega</code>	Ω
<code>\sigma</code>	σ	<code>\Sigma</code>	Σ

A formatação do texto que será exibido pelos comandos `xlabel`, `ylabel`, `title` e `text` pode ser feita adicionando-se, logo após a *string* de texto, *Propriedade* e *Valor_propriedade* no argumento do comando. Por exemplo, com essas opções, o comando `text` é escrito na forma:

```
text (x, y, 'string de texto', Propriedade, Valor_propriedade)
```

Os outros três comandos têm formato geral semelhante ao comando `text` mostrado acima. O nome da Propriedade é digitado como uma *string* e o Valor_propriedade entra como um número, se o valor da propriedade for um número, ou como uma *string*, se o valor da propriedade for uma palavra ou um caractere. Algumas Propriedades e os valores correspondentes são:

Propriedade	Descrição	Valores possíveis
Rotation	Especifica a orientação do texto.	Escalar (graus) Padrão (default): 0
FontAngle	Especifica estilo dos caracteres (itálico ou normal).	normal, itálico Padrão (default): normal
FontName	Especifica a fonte para o texto.	Nome da fonte disponível no sistema
FontSize	Especifica o tamanho da fonte.	Escalar (pontos) Padrão (default): 10
FontWeight	Especifica a largura dos caracteres.	light, normal, bold Padrão (default): normal
Color	Especifica a cor do texto.	Especificadores de cor (veja Seção 5.1)
BackgroundColor	Especifica a cor do fundo retangular em volta do texto.	Especificadores de cor (veja Seção 5.1)
EdgeColor	Especifica a cor da borda de uma caixa retangular em volta do texto.	Especificadores de cor (veja Seção 5.1) Default: nenhum
LineWidth	Especifica a largura da borda de uma caixa retangular em volta do texto.	Escalar (pontos) Padrão (default): 0.5

O comando `axis`:

Quando um comando `plot x, y` é executado, o MATLAB cria eixos cujos limites estão baseados nos valores mínimo e máximo dos vetores `x` e `y`. O comando `axis` permite que a faixa de valores avaliados seja modificada no gráfico. Muitas vezes a faixa de valores ajusta-se insatisfatoriamente no gráfico e é necessário estendê-la, para que detalhes escondidos no gráfico sejam vistos. A seguir, vemos algumas formas possíveis do comando `axis`:

<code>axis ([xmin, xmax, ymin, ymax])</code>	Indica os limites tanto do eixo x quanto do eixo y (xmin, xmax, ymin e ymax são números).
<code>axis equal</code>	Configura escalas iguais para os eixos x e y.
<code>axis square</code>	Configura as escalas x e y, de modo que a região do gráfico seja um quadrado.
<code>axis tight</code>	Configura os limites dos eixos para a faixa de dados.
O comando grid:	
<code>grid on</code>	Insere linhas de <i>grid</i> no gráfico.
<code>grid off</code>	Remove linhas de <i>grid</i> do gráfico.

A seguir, um exemplo de uma rotina criada para gerar o gráfico correspondente à Figura 5-1, usando os comandos descritos anteriormente.

```
x = [10:0.1:22];
y = 95000./x.^2;
xd = [10:2:22];
yd = [950 640 460 340 250 180 140];
plot(x, y, '-', 'LineWidth', 1.0)
xlabel('Distancia (cm)')
ylabel('Intensidade (lux)')
title('fontname{Arial} Intensidade da luz em funcao da distancia', 'FontSize', 14)
axis([8 24 0 1200])
text(14, 700, 'Comparacao entre dados teoricos e experimentais.', 'EdgeColor', 'r', 'LineWidth', 2)
hold on
plot(xd, yd, 'ro--', 'linewidth', 1.0, 'markersize', 10)
legend('Teoria', 'Experimento', 0)
hold off
```

5.4.2 Formatando um gráfico da janela Figure Window

Um gráfico pode ser formatado interativamente na janela Figure Window clicando-se sobre o gráfico e/ou usando menus. A Figura 5-8 é a repetição do gráfico da Figura 5-1. O botão Edit Plot pode ser utilizado para introduzir novos itens de formatação ou para modificar a formatação existente no gráfico (gerada por comandos).

5.5 GRÁFICOS COM ESCALAS LOGARÍTMICAS

Muitas aplicações nas engenharias e demais áreas das ciências exatas requerem gráficos com escalas monolog ou bilog*. Escalas logarítmicas possibilitam a apresentação gráfica de dados que cobrem uma faixa extensa de valores. Outra finalidade dessa escala é per-

* N. de T.: Em inglês, *semilog* e *loglog*, respectivamente.

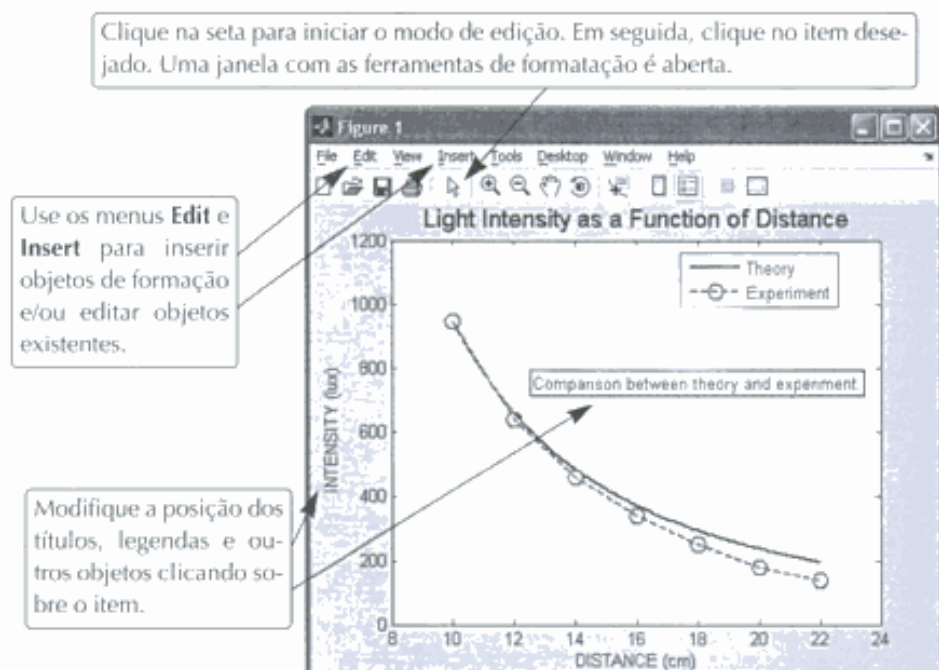


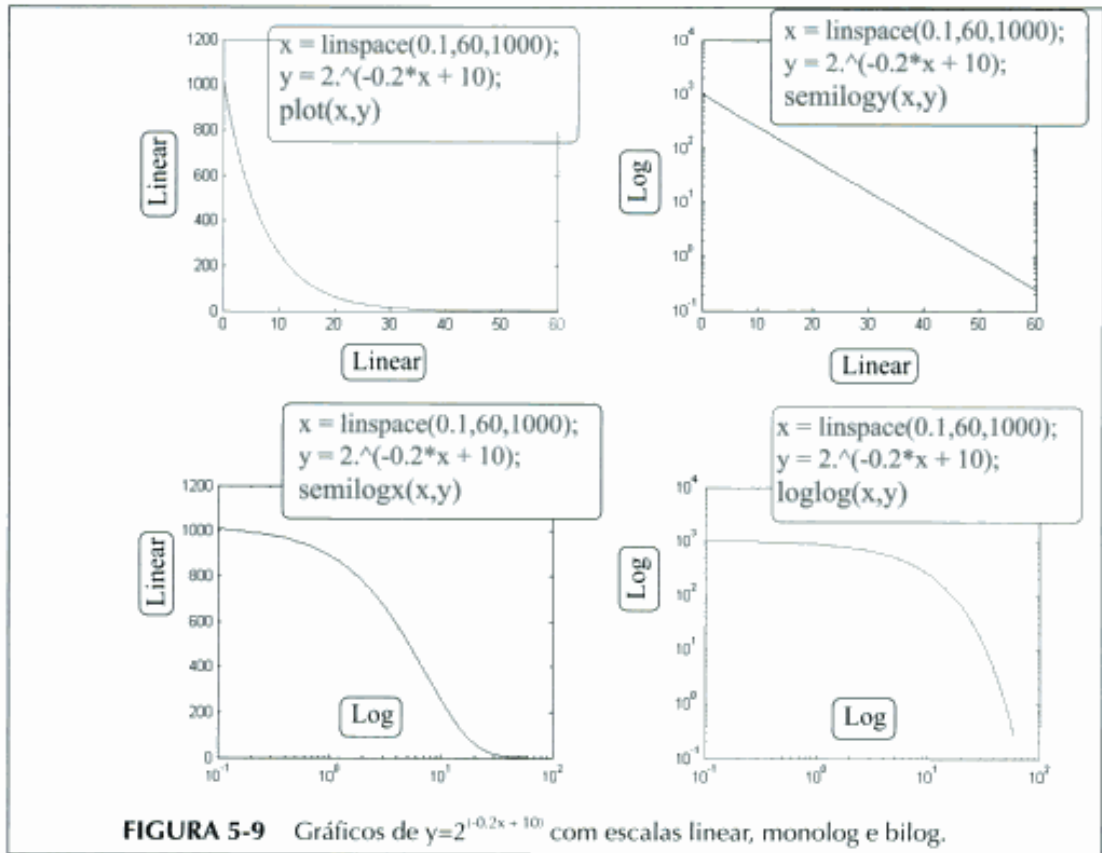
FIGURA 5-8 Formatando um gráfico usando o Edit plot.

mitir a identificação das constantes de ajuste em relações matemáticas entre as variáveis de um problema durante a modelagem de dados (por exemplo, em uma regressão linear – veja a Seção 8.2.2).

Os comandos do MATLAB para a seleção de escalas logarítmicas são:

<code>semilogy (x, y)</code>	Plota <i>y versus x</i> , monolog em <i>y</i> ; isto é, uma escala logarítmica (base 10) é atribuída ao eixo <i>y</i> e uma escala linear ao eixo <i>x</i> .
<code>semilogx (x, y)</code>	Plota <i>y versus x</i> , monolog em <i>x</i> ; isto é, uma escala logarítmica (base 10) é atribuída ao eixo <i>x</i> e uma escala linear ao eixo <i>y</i> .
<code>loglog (x, y)</code>	Plota <i>y versus x</i> , bilog; isto é, logarítmica (base 10) para os eixos <i>x</i> e <i>y</i> .

Especificadores de linha, Propriedades e Valor_da_propriedade podem ser inseridos nesses comandos (opcionalmente), assim como no comando `plot`. Por exemplo, a Figura 5-9 mostra gráficos da função $y = 2^{(-0.2x + 10)}$ para $0.1 \leq x \leq 60$. Todos os gráficos representam a mesma função (da esquerda para a direita, de cima para baixo): no primeiro, foram utilizadas escalas lineares para os dois eixos; no segundo, o eixo *y* recebeu uma escala logarítmica e o eixo *x* recebeu uma escala linear; no terceiro, o eixo *x* recebeu uma escala logarítmica e o eixo *y* recebeu uma escala linear e, finalmente, no quarto, ambos os eixos receberam escalas logarítmicas.



Observações quanto ao uso de escalas logarítmicas:

- Não é possível atribuir o número zero ao início de uma escala logarítmica (porque o logaritmo de zero não está definido).
- Números negativos não podem ser atribuídos a escalas logarítmicas (porque o logaritmo de números negativos não está definido).

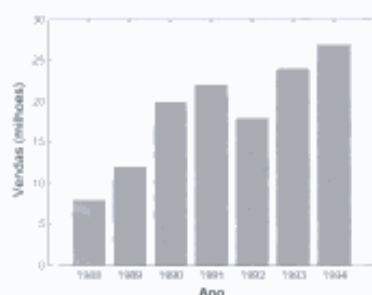
5.6 GRÁFICOS PARA FINS ESPECÍFICOS

Todos os gráficos gerados ao longo deste capítulo foram traçados a partir da união finita de linhas, isto é, cada par de valores (x, y) foi interligado ao vizinho mais próximo por linhas de diferentes tipos. Em algumas situações, gráficos de diferentes tipos ou geometria apresentam os dados de um modo mais eficaz. Estão incluídos nessa categoria os gráficos em barras, colunas, degraus, pizzas, hastes e muitos outros. A seguir, são mostrados alguns exemplos de gráficos especiais fáceis de serem criados no MATLAB. Uma lista completa de gráficos e funções que o MATLAB pode plotar e a maneira de usá-las pode ser encontrada no Help Window. Nessa janela, selecione “Functions by Category”; em seguida, a opção “Graphics” e, por último, selecione “Basic Plots and Graphics” ou “Specialized Plotting”.

Barras
verticais

Sintaxe da
função:

`bar (x,y)`



```
>> ano=[1988:1994];
vendas=[8 12 20 22 18 24 27];
bar(ano,vendas,'r')
```

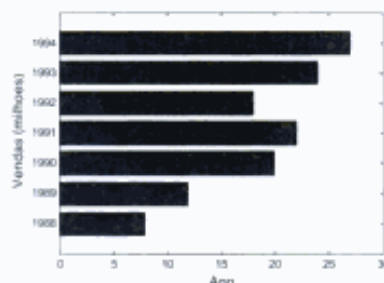
Barras em vermelho

```
xlabel('Ano')
ylabel('Vendas (milhões)')
```

Barras
horizontais

Sintaxe da
função:

`barh (x,y)`

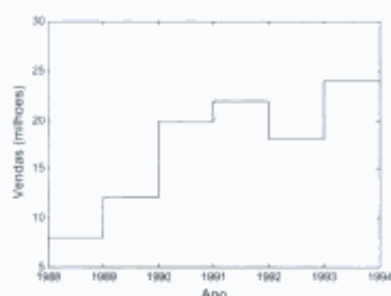


```
>> ano = [1988:1994];
vendas = [8 12 20 22 18 24 27];
barh(ano,vendas)
xlabel = ('Ano')
ylabel('Vendas (milhões)')
```

Degraus

Sintaxe da
função:

`stairs`
`(x,y)`

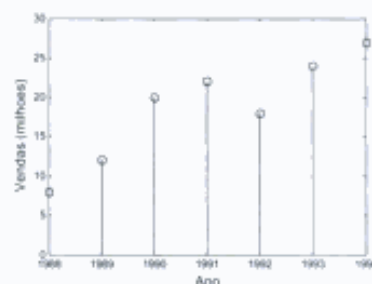


```
>> ano = [1988:1994];
vendas = [8 12 20 22 18 24 27];
stairs(ano,vendas)
```

Hastes

Sintaxe da
função:

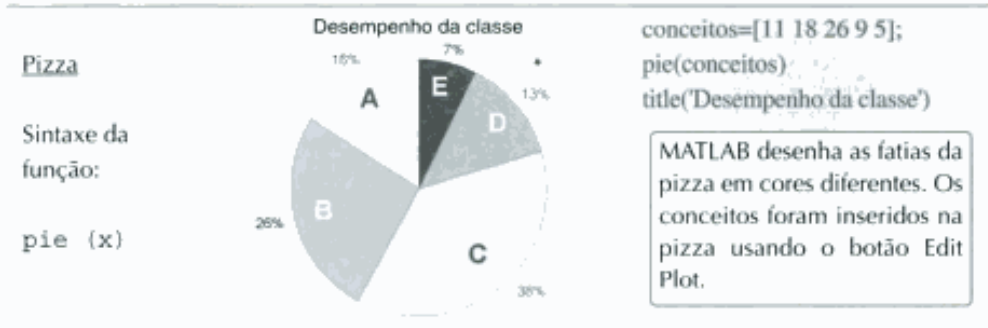
`stem (x,y)`



```
>> ano = [1988:1994];
vendas = [8 12 20 22 18 24 27];
stem(ano,vendas)
```

Gráficos na forma de pizza são úteis para visualização relativa de quantidades que juntas perfazem um todo. Por exemplo, a tabela a seguir apresenta as notas obtidas pelos alunos de uma determinada classe. Os dados na tabela são utilizados para criar um gráfico de pizza.

Conceito	A	B	C	D	E
Quantidade de alunos	11	18	26	9	5



5.7 HISTOGRAMAS

Histogramas são gráficos destinados à apresentação de uma distribuição de dados. Toda a faixa de um determinado conjunto de dados é dividida em intervalos menores (denominados *bins* no MATLAB); o histograma mostra quantos pontos do conjunto de dados estão em cada intervalo. Tecnicamente, histograma é uma representação gráfica de uma distribuição de frequência em que as frequências das classes são representadas pelas áreas de retângulos verticais adjacentes, com as bases colineares e proporcionais aos intervalos das classes. Os histogramas são criados no MATLAB do comando `hist`. A forma mais simples desse comando é:

```
hist (y)
```

`y` é um vetor com os dados das classes. O MATLAB divide o intervalo de dados em 10 subintervalos igualmente espaçados e, então, plota a frequência de classes nesse subintervalo.

Por exemplo, os dados a seguir correspondem à medição diária de temperatura (em °F) em Washington DC, durante o mês de abril de 2002: 58, 73, 73, 53, 50, 48, 56, 73, 73, 66, 69, 63, 74, 82, 84, 91, 93, 89, 91, 80, 59, 69, 56, 64, 63, 66, 64, 74, 63, 69 (dados do U.S. National Oceanic and Atmospheric Administration). Um histograma desses dados é obtido a seguir.

```
>> y=[58 73 73 53 50 48 56 73 73 66 69 63 74 82 84 91 93 89 91 80 59 69 56 64 63 66 64 74 63 69];
>> hist(y)
```

O gráfico gerado é apresentado na Figura 5-10 (os rótulos dos eixos foram inseridos através do botão Edit Plot). O menor valor no conjunto de pontos é 48 e o maior é 93. Isso implica que o intervalo de valores medidos de temperatura é 45 e que cada subintervalo (*bin*) mede 4.5°F. Portanto, o primeiro subintervalo vai de 48 a 52.5 e contém dois pontos. A faixa do segundo subintervalo vai de 52.5 a 57 e contém três pontos, e assim sucessivamente. Os intervalos 75 a 79.5 e 84 a 89.5 não contém pontos.

Como a divisão em 10 intervalos igualmente espaçados pode não ser satisfatória, o número de subintervalos pode ficar a critério do usuário. Para tanto, o usuário especifica

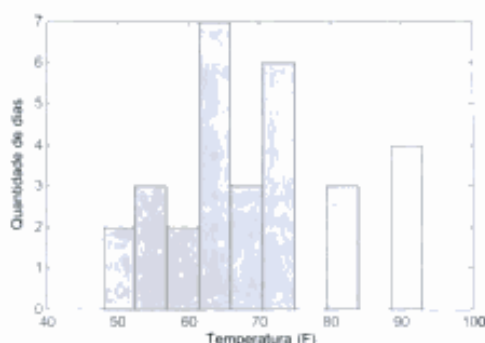


FIGURA 5-10 Histograma dos dados relativos à medição de temperatura no mês abril 2002.

o número de *bins* requeridos, ou então, indica onde deve ficar o ponto central de cada *bin*. Os formatos dos comandos `hist` para essas duas opções são:

`hist (y, nbins)` ou `hist (x,y)`

nbins indica a quantidade de subintervalos. O MATLAB divide o intervalo original em subintervalos iguais.

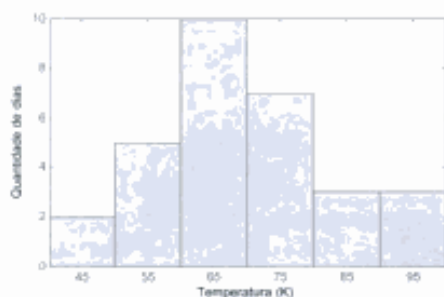
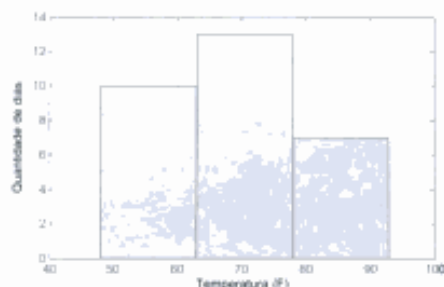
x é um vetor que especifica a localização do centro de cada subintervalo (a distância entre os centros não será necessariamente a mesma para todos os subintervalos). As bordas dos subintervalos estão no ponto médio entre os centros.

No exemplo a seguir, o usuário redefine a faixa de temperaturas do exemplo anterior em três subintervalos. Isso é feito a partir da execução do comando

```
>> hist (y, 3)
```

O resultado da execução desse comando mostra a divisão em três subintervalos.

A quantidade e a largura dos subintervalos também podem ser especificados pelo vetor **x**, cujos elementos definem os centros dos subintervalos. Por exemplo, a figura ao lado é um histograma apresentando a faixa de temperaturas em 6 subintervalos, espaçados de 10 graus Fahrenheit. Os elementos do vetor **x** (45, 55, 65, 75, 85, 95) especificam os centros dos subintervalos. A figura pode ser gerada executando-se os comandos



```
>> x = [45:10:95]
x =
    45    55    65    75    85    95
>> hist(y,x)
```

O comando `hist` ainda pode ser utilizado com outras opções que retornam dados estatísticos sobre um histograma. Esses dados podem ser obtidos a partir de um dos seguintes comandos:

```
n = hist (y)
```

```
n = hist (y, nbins)
```

```
n = hist (y, x)
```

A variável de saída (`n`) será um vetor. A quantidade de elementos desse vetor é igual ao número de subintervalos e cada elemento de `n` representa o número de frequências dentro do subintervalo. Por exemplo, os dados do histograma da Figura 5-10 também podem ser criados com o seguinte comando:

```
>> n = hist(y)
n =
    2    3    2    7    3    6    0    3    0    4
```

O vetor `n` mostra quantos elementos existem em cada subintervalo.

Como se pode ver, o vetor `n` mostra que o primeiro subintervalo possui duas ocorrências de temperatura, o segundo tem 3 e assim sucessivamente.

Para finalizar, o centro de cada subintervalo pode ser vislumbrado pelo usuário. Os comandos a seguir mostram como fazê-lo:

```
[n xout] = hist (y)
```

```
[n xout] = hist (y, nbins)
```

`xout` é um vetor onde o valor de cada elemento representa a localização do centro do subintervalo correspondente. Por exemplo, para o histograma da Figura 5-10:

```
>> [n xout] = hist(y)
n =
    2    3    2    7    3    6    0    3    0    4
xout =
    50.2500    54.7500    59.2500    63.7500    68.2500    72.7500    77.2500    81.7500    86.2500    90.7500
```

O vetor `xout` exibe o centro do primeiro subintervalo (50.25), do segundo (54.75), etc.

5.8 GRÁFICOS POLARES

As coordenadas polares especificam a localização de um ponto no plano dado um ângulo (θ) e uma distância radial (r) relativamente à origem do sistema de coordenadas. Essas

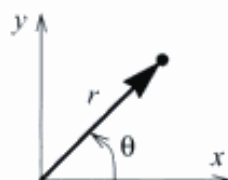
coordenadas são muito importantes em problemas nas engenharias e ciências em geral. O comando `polar` é utilizado para plotar funções em coordenadas polares. O formato geral do comando é:

```
polar (teta, raio, 'especificadores de linha')
```

Vetor

Vetor

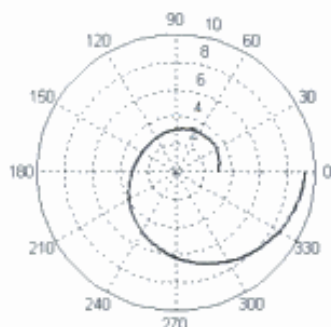
(opcional) Especificadores que definem o tipo, cor da linha e marcadores



onde `teta` e `raio` são vetores cujos elementos definem as coordenadas dos pontos a serem plotados. O comando `polar` plota os pontos e desenha o *grid* polar (gráfico radar). Os especificadores de linha são os mesmos discutidos no comando `plot`. Para plotar uma função $r = f(\theta)$, dado um certo domínio, deve ser criado um vetor com os valores de θ e, em seguida, o vetor `r` com correspondentes valores de $f(\theta)$ é criado usando operações escalares com o vetor (elemento por elemento). Então, os dois vetores podem ser utilizados como argumentos do comando `polar`.

Por exemplo, vamos gerar o gráfico da função $r = 3\cos^2(0.5\theta) + \theta$, para $0 \leq \theta \leq 2\pi$:

```
>> teta = linspace(0,2*pi,200);
>> r = 3*cos(0.5*teta).^2 + teta;
>> polar(teta,r)
```

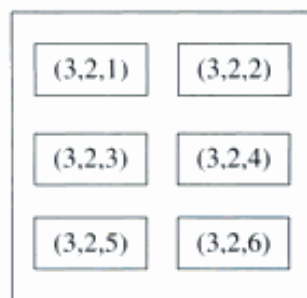


5.9 MÚLTIPLOS GRÁFICOS NA MESMA JANELA DE SAÍDA

O comando `subplot` é destinado à exibição de múltiplos gráficos na mesma janela. A forma geral desse comando é:

```
subplot (m, n, p)
```

Esse comando divide a janela Figure Window em $m \times n$ subjanelas onde os gráficos serão plotados. Os gráficos são organizados como elementos de uma matriz $m \times n$, onde cada elemento da matriz é um gráfico. As subjanelas são enumeradas de 1 até $m \cdot n$. A primeira subjanela está localizada no topo da matriz, no alto mais à esquerda, e a última subjanela é aquela na base da matriz, em baixo mais à direita. A numeração das subjanelas cresce da esquerda para a direita e de cima para baixo. O

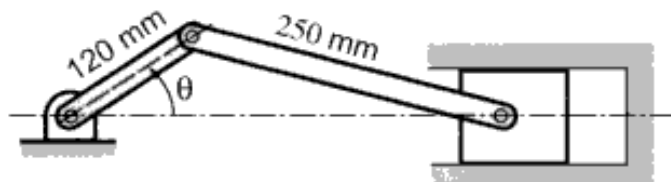


comando `subplot(m,n,p)` indica que o gráfico a ser plotado está localizado na janela endereçada por `p`. Isso significa que o próximo comando `plot` e quaisquer comandos de formatação serão criados nessa subjanela. Por exemplo, o comando `subplot(3,2,1)` cria 6 subjanelas organizadas em 3 linhas e 2 colunas, selecionando a subjanela localizada no alto, mais à esquerda. No Problema-exemplo 5-2 é apresentada uma solução envolvendo o comando `subplot`.

5.10 EXEMPLOS DE APLICAÇÕES DO MATLAB

Problema-exemplo 5-2: Mecanismo pistão-manivela

O mecanismo constituído de uma manivela de êmbolo é usado em muitas aplicações na engenharia. No mecanismo mostrado a seguir, a manivela gira a uma velocidade constante de 500 rpm.



Calcule e esboce a posição, a velocidade e a aceleração do êmbolo (pistom) para uma revolução da manivela. Construa três gráficos na mesma janela de saída. Como condições iniciais, faça $\theta = 0^\circ$ em $t = 0$.

Solução

O êmbolo está girando com uma velocidade angular constante. Utilizando as condições iniciais acima ($\theta = 0^\circ$ quando $t = 0$), pode-se afirmar que o ângulo θ num determinado instante de tempo t é dado por $\theta = \dot{\theta}t$ e que $\ddot{\theta} = 0$, para todo t .

Baseado na figura, as distâncias d_1 e h são dadas por:

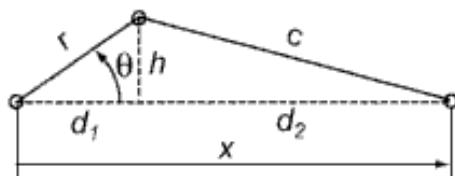
$$d_1 = r \cos \theta \text{ e } h = r \sin \theta.$$

Conhecendo-se h , a distância d_2 pode ser calculada aplicando-se o Teorema de Pitágoras:

$$d_2 = (c^2 - h^2)^{1/2} = (c^2 - r^2 \sin^2 \theta)^{1/2}$$

A posição x do pistão é dada por:

$$x = d_1 + d_2 = r \cos \theta + (c^2 - r^2 \sin^2 \theta)^{1/2}$$



Derivando-se* x em relação a t obtemos a velocidade do êmbolo:

$$\dot{x} = -r\dot{\theta}\sin\theta - \frac{r^2\dot{\theta}\sin 2\theta}{2(c^2 - r^2\sin^2\theta)^{1/2}}$$

A derivada segunda de x em relação a t é a aceleração do êmbolo:

$$\ddot{x} = -r\ddot{\theta}\cos\theta - \frac{4r^2\dot{\theta}^2\cos 2\theta(c^2 - r^2\sin^2\theta) + (r^2\dot{\theta}\sin 2\theta)^2}{4(c^2 - r^2\sin^2\theta)^{3/2}}$$

observando-se que na equação acima foi considerado que $\ddot{\theta}$ é zero.

Uma rotina no MATLAB para calcular e esboçar a posição, a velocidade e a aceleração do êmbolo para uma revolução da manivela é mostrada abaixo:

```
TETAPrpm = 500; r = 0.12; c = 0.25;
```

Declara θ , r e c .

```
TETAP = TETAPrpm*2*pi/60;
```

Converte a unidade $\dot{\theta}$ rpm para rad/s.

```
tf = 2*pi/TETAP;
```

Determina o tempo de uma revolução da manivela.

```
t = linspace(0, tf, 200);
```

Cria um vetor tempo com 200 elementos.

```
TETA = TETAP*t;
```

Determina θ para cada t .

```
d2s = c^2 - r^2*sin(TETA).^2;
```

Calcula o quadrado de d_2 para cada θ .

```
x = r*cos(TETA) + sqrt(d2s);
```

Calcula x para cada θ .

```
xp = -r*TETAP*sin(TETA) - (r^2*TETAP*sin(2*TETA))/(2*sqrt(d2s));
```

```
xpp = -r*TETAP^2*cos(TETA) - (4*r^2*TETAP^2*cos(2*TETA).*d2s + ...
```

```
(r^2*sin(2*TETA)*TETAP.^2)/(4*d2s.^(3/2));
```

Calcula $\dot{x}$ e $\ddot{x}$ para cada θ .

```
subplot(3,1,1);
```

```
plot(t,x)
```

Plota x versus t .

```
grid
```

Formata o primeiro gráfico.

```
xlabel('Tempo (s)')
```

```
ylabel('Posicao (m)')
```

```
subplot(3,1,2)
```

```
plot(t, xp)
```

Plota a velocidade $\dot{x}$ em função de t .

```
grid
```

Formata o segundo gráfico.

* N. de T.: Perceba que a variável x não foi escrita explicitamente como uma função de t . Note, entretanto, que o ângulo θ está relacionado a t (através da constante $\dot{\theta}$). A regra da cadeia foi utilizada para derivar x em relação a t .

```
xlabel('Tempo (s)')
```

```
ylabel ('Velocidade (m/s)')
```

```
subplot(3,1,3)
```

```
plot(t, xpp)
```

Plota a aceleração $\ddot{x}$ em função de t .

```
grid
```

Formata o terceiro gráfico.

```
xlabel('Tempo (s)')
```

```
ylabel ('Aceleracao (m/s^2)')
```

Executando a rotina na linha do *prompt* do Command Window, serão exibidos três gráficos na mesma janela (veja a Figura 5-11). O gráfico velocidade *versus* tempo mostra que, num período de revolução da manivela, a velocidade do êmbolo vale zero nas extremidades. A aceleração é máxima (dirigida para a esquerda) quando o êmbolo está na extremidade mais à direita.

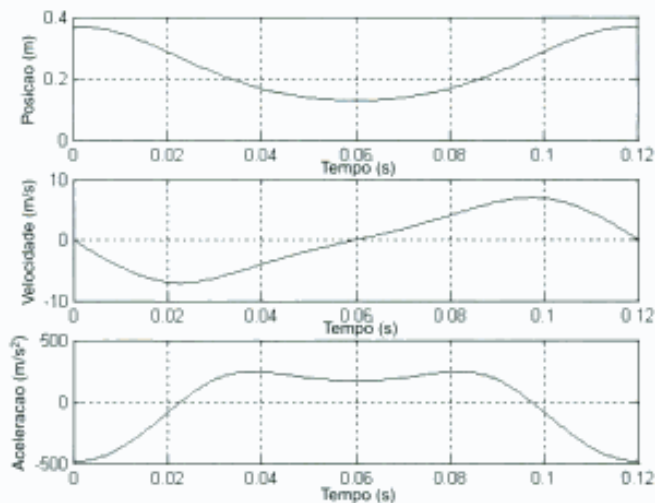


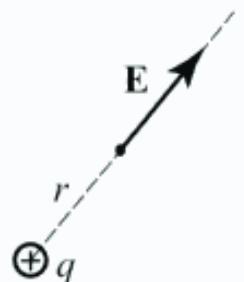
FIGURA 5-11 Gráficos da posição, velocidade e aceleração do êmbolo em função do tempo.

Problema-exemplo 5-3: Dipolo elétrico

O campo elétrico em um ponto do espaço devido à presença de uma carga pontual q é um vetor $\mathbf{E}$ cujo módulo E é dado pela lei de Coulomb:

$$E = \frac{1}{4\pi\epsilon_0} \frac{q}{r^2}$$

onde $\epsilon_0 = 8.8541878 \times 10^{-12} \frac{\text{C}^2}{\text{N}\cdot\text{m}^2}$ é a permissividade do vácuo, q é o módulo da carga elétrica e r é a distância entre a carga e o



ponto em questão. A direção do campo $\mathbf{E}$ situa-se ao longo da linha que interliga a carga ao ponto. Se a carga q é positiva, o campo elétrico $\mathbf{E}$ tem o sentido afastando-se da carga. Caso q seja negativa, o sentido de $\mathbf{E}$ aponta em direção à carga, ao longo da linha que interliga o ponto à carga. Um dipolo elétrico é um sistema composto de duas cargas elétricas pontuais, de mesmo valor, mas de sinais opostos, separados por uma distância pequena. O campo elétrico produzido por essa configuração de cargas é a superposição dos campos elétricos individuais.

A figura ao lado ilustra um dipolo elétrico com o módulo da carga $q = 12 \times 10^{-9} \text{C}$. Determine e esboce o módulo do campo elétrico ao longo do eixo x , entre $x = -5 \text{cm}$ e $x = 5 \text{cm}$.

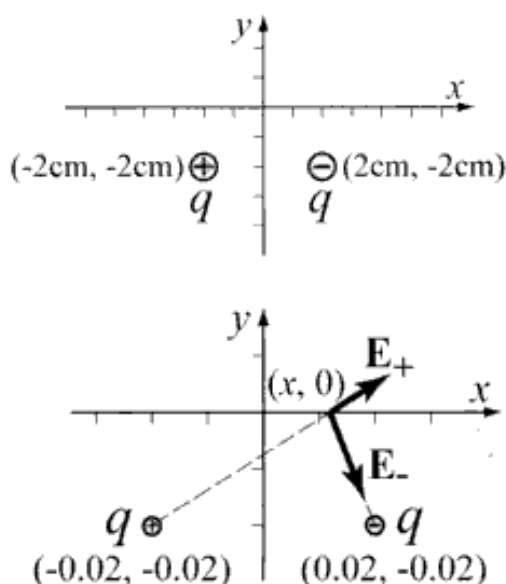
Solução

O campo elétrico $\mathbf{E}$ em um ponto arbitrário $(x, 0)$, ao longo do eixo x , é obtido superpondo-se os vetores campo elétrico devido a cada uma das cargas individuais:

$$\mathbf{E} = \mathbf{E}_- + \mathbf{E}_+$$

O módulo do campo elétrico é a intensidade do vetor campo elétrico $\mathbf{E}$.

O problema será resolvido da seguinte maneira:



Passo 1: Declararemos um vetor x para representar os pontos ao longo do eixo x .

Passo 2: Determinaremos a distância (e a distância²) de cada carga relativamente ao eixo x .

$$r_- = \sqrt{(0.02 - x)^2 + 0.02^2} \quad r_+ = \sqrt{(x + 0.02)^2 + 0.02^2}$$

Passo 3: Escreveremos vetores unitários nas direções das linhas que ligam as cargas aos pontos sobre o eixo x .

$$\mathbf{e}_- = \frac{1}{r_-} ((0.02 - x)\mathbf{i} - 0.02\mathbf{j})$$

$$\mathbf{e}_+ = \frac{1}{r_+} ((x + 0.02)\mathbf{i} + 0.02\mathbf{j})$$

Passo 4: Determinaremos o módulo dos vetores $\mathbf{E}_-$ e $\mathbf{E}_+$ em cada ponto usando a lei de Coulomb:

$$|\mathbf{E}_-| = \frac{1}{4\pi\epsilon_0} \frac{q}{r_-^2} \quad |\mathbf{E}_+| = \frac{1}{4\pi\epsilon_0} \frac{q}{r_+^2}$$

Passo 5: Determinaremos os vetores $\mathbf{E}_-$ e $\mathbf{E}_+$ multiplicando os vetores unitários pelos módulos $|\mathbf{E}_-|$ e $|\mathbf{E}_+|$.

Passo 6: Escreveremos o vetor E superpondo os vetores E_- e E_+ .

Passo 7: Determinaremos E , o módulo do campo E .

Passo 8: Esboçaremos E em função de x .

A rotina abaixo resolve o problema seguindo os passos anteriores:

```
q = 12e-9;
```

```
epsilon0 = 8.8541878e-12;
```

```
x = [-0.05: 0.001: 0.05];
```

```
rmenosS = (0.02-x).^2 + 0.02^2; rmenos = sqrt(rmenosS);
```

```
rmaisS = (x + 0.02).^2 + 0.02^2; rmais = sqrt(rmaisS);
```

```
EmenosUV = [((0.02 - x)/ rmenos), (-0.02./rmenos)];
```

```
EmaisUV = [((0.02 + x)/ rmais), (0.02./rmais)];
```

```
EmenosMOD = (q/(4*pi*epsilon0))./rmenosS;
```

```
EmaisMOD = (q/(4*pi*epsilon0))./rmaisS;
```

```
Emenos = [EmenosMOD.*emenos(:,1), EmenosMOD.*Emenos(:,2)];
```

```
Emais = [EmaisMOD.*eais(:,1), EmaisMOD.*Emais(:,2)];
```

```
E = Emenos + Emais;
```

```
EMOD = sqrt (E(:,1).^2 + E(:,2).^2);
```

```
plot (x, EMOD, 'k', 'linewidth', 1)
```

```
xlabel ('Posicao ao longo do eixo x (m)', 'FontSize', 12)
```

```
ylabel ('Modulo do campo elétrico (N/C)', 'FontSize', 12)
```

```
title ('CAMPO ELETRICO DEVIDO A UM DIPOLO ELETRICO', 'FontSize', 12)
```

Cria o vetor coluna x .

Passo 2: cada variável é declarada como um vetor coluna.

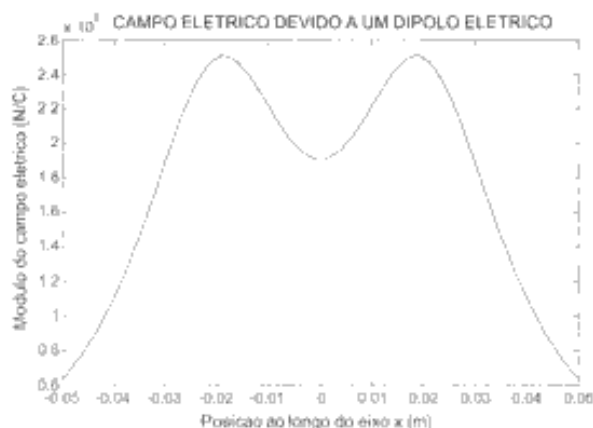
Passos 3 e 4: cada variável é uma matriz de duas colunas. Cada linha é o vetor para o correspondente valor de x .

Passo 6

Passo 7

Passo 5

Salvando e executando a rotina na linha do *prompt* da janela Command Window, obtemos:



5.11 PROBLEMAS

1. Construa dois gráficos da função $f(x) = 0.6x^5 - 5x^3 + 9x + 2$ separadamente; no primeiro, use o intervalo $-4 \leq x \leq 4$ e, no segundo, use o intervalo $-2.7 \leq x \leq 2.7$.

2. Esboce a função $f(x) = \frac{x^2 - x + 1}{x^2 + x + 1}$ para $-10 \leq x \leq 10$.

3. Use o comando `fplot` para esboçar a função

$$f(x) = 0.01x^5 - 0.03x^4 + 0.4x^3 - 2x^2 - 6x + 5$$

no domínio $-4 \leq x \leq 6$.

4. Esboce a função, $f(x) = \frac{1.5x}{x-4}$ $-10 \leq x \leq 10$. Perceba que a função é descontínua em $x = 4$. Plote a função criando dois vetores para o domínio x . O primeiro vetor ($x1$) terá elementos -10 a 3.7 , à esquerda da descontinuidade, e o segundo vetor ($x2$) terá elementos de 4.3 a 10 , à direita da descontinuidade. Para cada vetor x , crie um vetor y (denote-os $y1$ e $y2$) com os valores correspondentes da função. Esboce a função colocando as duas curvas no mesmo gráfico na janela de saída ($y1$ vs. $x1$ e $y2$ vs. $x2$).

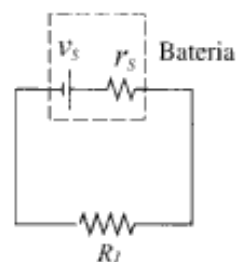
5. Esboce a função, $f(x) = \frac{x^2 - 5x + 10}{x^2 - 2x - 3}$ $-10 \leq x \leq 10$. Perceba que a função é descontínua em dois pontos. Plote a função dividindo o domínio x em três partes; o primeiro domínio partindo do ponto $x = -10$ até um limite lateral esquerdo da menor descontinuidade, o segundo domínio situando entre as duas descontinuidades e o terceiro iniciando à direita da maior descontinuidade e estendendo-se até $x = 10$. Fixe os valores do contradomínio (y) entre -20 e 20 .

6. Trace o gráfico da função $f(x) = 3x \sin(x) - 2x$ e da sua derivada, no mesmo gráfico na janela de saída, para $-2\pi \leq x \leq 2\pi$. Plote a função usando uma linha cheia e a derivada usando uma linha tracejada. Insira uma legenda e coloque título nos eixos.

7. Um circuito elétrico série possui uma fonte de tensão v_s , dotada de uma resistência interna r_s , e recebe acoplada na saída uma resistência de carga R_L (veja a figura).

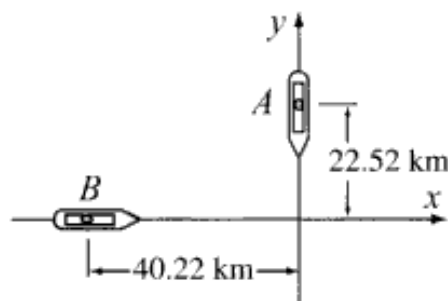
A potência dissipada na carga P é dada por:

$$P = \frac{v_s^2 R_L}{(R_L + r_s)^2}$$



Plote o gráfico da potência P em função da resistência de carga para $1 \leq R_L \leq 10$, dado que $v_s = 12\text{V}$ e $r_s = 2.5\Omega$.

8. Um barco A viaja para o sul a uma velocidade de 12.87km/h e um barco B viaja para o leste a uma velocidade de 25.74km/h . Quando relógios sincronizados em cada barco indicam 7 horas da manhã, os dois barcos estão posicionados de acordo com a figura. Plote um gráfico da distância relativa entre



os dois barcos como uma função do tempo, para um intervalo de tempo de 4 horas, a partir da situação mostrada na figura. Construa o gráfico de tal modo que o eixo horizontal represente as horas do dia, iniciando às 7 horas, e o eixo vertical mostre a distância correspondente. Em seguida, coloque títulos nos eixos. Se a visibilidade é 12.87km , estime o intervalo de tempo em que as pessoas nos barcos podem enxergar umas às outras.

9. O saldo V de uma conta de poupança, aplicadas P unidades monetárias, sendo paga uma taxa de juros anual r , é dado por: $V = P \left(1 + \frac{r}{m} \right)^{mt}$, onde m é a frequência da aplicação da taxa dos juros compostos em um ano e t é o período de investimento em anos (Se a taxa de juros é composta anualmente $m = 1$, trimestralmente $m = 4$, etc.) Se os juros são aplicados continuamente, o valor do investimento é dado por: $V = P e^{rt}$.

Considere um investimento inicial de \$ 5,000, aplicados durante 15 anos, a uma taxa de juros anual de 7.5%. Apresente a diferença no saldo da conta quando os juros incidem anual, trimestral e continuamente, construindo um gráfico do saldo V em função do tempo (em anos) para cada método. Plote os três casos no mesmo gráfico na janela de saída, use linhas diferentes para cada gráfico, titule os eixos, crie legenda e adicione título aos gráficos.

10. O monumento Gateway Arch, em St. Louis, tem um formato de acordo com a equação:

$$y = 693.8 - 68.8 \cosh \left(\frac{x}{99.7} \right) \text{ ft.}$$

Faça um gráfico da função que representa esse monumento.

11. A intensidade M de um terremoto na escala Richter é dada por:

$$M = \frac{2}{3} \log \frac{E}{10^{4.4}}$$

onde E é a energia (em Joules) liberada no terremoto. Faça um gráfico de E (ordenada) vs. M (abscissa) para $3 \leq M \leq 8$. Use uma escala logarítmica para E e outra linear para M . Titule os eixos e o gráfico.

12. A posição x de uma partícula, em função do tempo, que se move ao longo de uma linha reta é dada por:

$$x(t) = 0.4t^3 - 2t^2 - 5t + 13 \text{ m}$$

A velocidade da partícula $v(t)$ é dada pela derivada primeira de $x(t)$ em relação ao tempo t . Já a aceleração $a(t)$ é determinada pela derivada de $v(t)$ em relação ao tempo t .

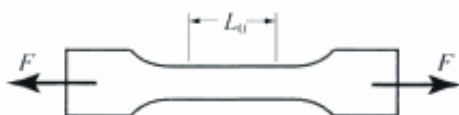
Determine as expressões para a velocidade e aceleração da partícula. Em seguida, construa os gráficos da posição, velocidade e aceleração em função de t ($0 \leq t \leq 7$ s). Use o comando `subplot` para colocar os três gráficos na mesma janela de saída. Os gráficos da posição, velocidade e aceleração devem ocupar, respectivamente, a posição de cima, no meio e de baixo na janela. Titule os eixos e gráficos, inserindo, apropriadamente, as unidades de medida nos eixos.

13. Em um teste de tração de um corpo de prova (amostra na forma de um osso), as extremidades da amostra são forçadas recebendo uma tensão mecânica (veja a figura). Durante o teste, a força F necessária para distender a amostra de um comprimento L é medida por um calibre de deformação. Esses dados são usados para plotar a curva de tensão-deformação do material que constitui o corpo. Existem duas definições usuais para tensão e deformação, uma bastante utilizada na engenharia e uma outra de uso geral. É claro que as duas definições coincidem em um certo regime analítico. As definições de tensão (σ_e) e deformação (ϵ_e) utilizadas na engenharia são:

$\sigma_e = \frac{F}{A_0}$ e $\epsilon_e = \frac{L - L_0}{L_0}$, onde L_0 e A_0 são o comprimento e a área da seção reta iniciais da amostra, respectivamente. As definições gerais de tensão (σ_l) e deformação (ϵ_l) são:

$$\sigma_l = \frac{F}{A} \frac{L}{L_0} \text{ e } \epsilon_l = \ln \frac{L}{L_0}.$$

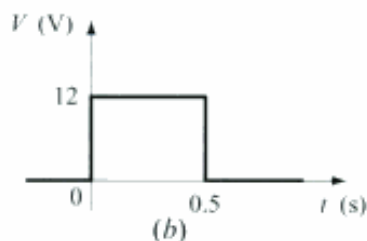
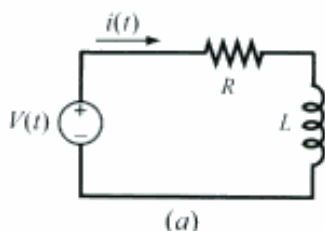
A tabela a seguir apresenta dados medidos da força e deformação em uma amostra de alumínio. A amostra possui área da seção reta circular de raio 6.4mm (antes do teste). Considere o comprimento inicial da amostra $L_0 = 25$ mm. Use os dados da tabela para calcular e construir as curvas de tensão-deformação usando as definições da engenharia e a geral, ambas na mesma janela de saída, mas em gráficos diferentes (usando o comando `subplot`), para o mesmo material. Insira títulos nos eixos e adicione título aos gráficos. Unidades



de medida: a força é medida em Newton (N) e a área em m^2 , a unidade de tensão é o Pascal (Pa).

$F(\text{N})$	0	13345	26689	40479	42703	43592	44482	44927
$L(\text{mm})$	25	25.037	25.073	25.113	25.122	25.125	25.132	25.144
$F(\text{N})$	45372	46276	47908	49035	50265	53213	56161	
$L(\text{mm})$	25.164	25.208	25.409	25.646	26.084	27.398	29.150	

14. Um resistor $R = 4\Omega$ e um indutor $L = 1.3\text{H}$ são conectados em série com uma fonte de tensão, conforme ilustra a Figura (a) (circuito RL). Quando a fonte de tensão produz pulsos de tensão retangulares de amplitude 12V e duração 0.5s [veja Figura (b)], a corrente $i(t)$ no circuito é uma função do tempo de acordo com:



$$i(t) = \frac{V}{R}(1 - e^{-Rt/L}) \text{ para } 0 \leq t \leq 0.5 \text{ s}$$

$$i(t) = e^{-Rt/L} \frac{V}{R}(e^{(0.5R/L)} - 1) \text{ para } 0.5 \leq t \text{ s}$$

Construa um gráfico da corrente em função do tempo para o intervalo $0 \leq t \leq 2\text{s}$.

15. A órbita dos planetas ao redor do Sol pode ser modelada aproximadamente pela equação polar:

$$r = \frac{eP}{1 - e \cos \theta}$$

Os valores das constantes P e e para quatro planetas são mostrados abaixo. Esboce a órbita dos quatro planetas em uma única janela de saída (use o comando `hold on`).

Planeta	$P (\times 10^6 \text{m})$	e	Planeta	$P (\times 10^6 \text{m})$	e
Mercúrio	269.2	0.206	Terra	8964	0.0167
Vênus	15913	0.00677	Marte	2421	0.0934

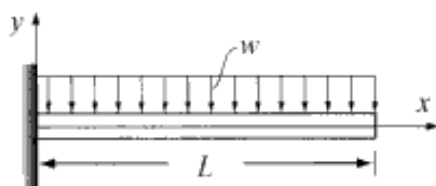
16. A densidade de probabilidade radial $P_r(r)$ para o primeiro estado excitado do átomo de hidrogênio (números quânticos $n = 2$ e $l = 0$) é dada por:

$$P_r(r) = \frac{1}{8a_0} \left(\frac{r}{a_0} \right)^2 \left(2 - \frac{r}{a_0} \right) e^{(-r/a_0)}$$

onde $a_0 = 52.92 \times 10^{-11} \text{ m}$ é o raio de Bohr.

Faça um gráfico de P_r em função de r/a_0 para $0 \leq r/a_0 \leq 15$.

17. Uma viga em balanço (cantiléver) possui uma extremidade permanentemente presa a uma parede e a outra extremidade livre para deflexionar. Aplicando uma carga (w) uniformemente sobre o corpo da viga, a deflexão y em um determinado ponto x é dada pela equação

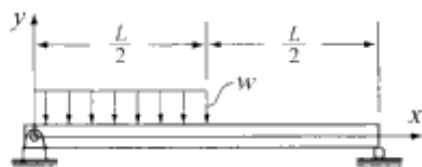


$$y = \frac{-w}{24EI} (x^4 - 4Lx^3 + 6L^2x^2)$$

onde E é a módulo de elasticidade, I é o momento de inércia e L é o comprimento da viga. Na viga mostrada na figura, considere que $L = 6 \text{ m}$, $E = 70 \times 10^9 \text{ Pa}$ (alumínio), $I = 9.19 \times 10^{-6} \text{ m}^4$ e $w = 80 \text{ N/m}$.

Plote um gráfico da deflexão da viga y em função de x .

18. Uma viga suspensa por duas extremidades fixas foi submetida a uma carga constante (w) (uniformemente distribuída) na metade do comprimento da viga. A deflexão y , em função de x , é dada pelo par de equações:



$$y = \frac{-wx}{384EI} (16x^3 - 24Lx^2 + 9L^3) \text{ para } 0 \leq x \leq \frac{L}{2}$$

$$y = \frac{-wL}{384EI} (8x^3 - 24Lx^2 + 17L^2x - L^3) \text{ para } \frac{L}{2} \leq x \leq L$$

onde E é o módulo de elasticidade, I é o momento de inércia e L é o comprimento da viga. Para a viga mostrada na figura, considere $L = 20 \text{ m}$, $E = 200 \times 10^9 \text{ Pa}$ (aço), $I = 348 \times 10^{-6} \text{ m}^4$ e $w = 5 \times 10^3 \text{ N/m}$.

Esboce o gráfico da deflexão y da viga em função de x .

19. O limite de escoamento (valor da tensão nominal de tração na transição do regime elástico para o regime plástico) da maioria dos metais é sensível à taxa com que o material é submetido à carga. Os dados a seguir dão o limite de escoamento de um certo tipo de aço em várias taxas de deformação.

Taxa de deformação (s^{-1})	0.00007	0.0002	0.05	0.8	4.2	215	3500
Limite de escoamento (MPa)	345	362	419	454	485	633	831

O limite de escoamento em função da taxa de deformação pode ser modelado por meio da equação

$$\sigma_{yld} = 350 \left[\left(\frac{\dot{\epsilon}_t}{210} \right)^{0.16} + 1 \right]$$

onde σ_{yld} é o limite de escoamento em MPa e $\dot{\epsilon}_t$ é a taxa de deformação em s^{-1} . Faça um gráfico do limite de escoamento (eixo vertical) *versus* a taxa de deformação (eixo horizontal). Use escala linear para o limite de escoamento e uma escala logarítmica para a taxa de deformação. Utilize marcadores para destacar o conjunto de pontos e trace o gráfico utilizando uma linha cheia. Insira os títulos convenientemente nos eixos e adicione uma legenda ao gráfico.

CAPÍTULO 6

Funções

Matematicamente, a função $f(x)$ é um objeto que relaciona dois conjuntos do domínio e contradomínio, onde a cada elemento x do domínio corresponde um só elemento do contradomínio y . É comum se expressar uma função na forma $y = f(x)$, onde $f(x)$ é, usualmente, uma expressão matemática em termos da variável x . Cada valor atribuído à variável x (entrada) corresponde a um valor para a variável y (saída). Existem muitas funções nativas do MATLAB, isto é, residentes na estrutura interna do programa. Para serem utilizadas em expressões matemáticas, basta digitar o nome da função na linha do *prompt*, com o(s) devido(s) argumento(s) (veja a Seção 1.5). Alguns exemplos de funções nativas: `sin(x)`, `cos(x)`, `sqrt(x)`, `exp(x)`, etc. Entretanto, a maioria dos programas requer, frequentemente, funções que não possuem estruturas declaradas, i.e. residentes, dentro do MATLAB. Nesses casos, a função de interesse deve ser declarada, isto é personalizada, dentro do MATLAB. Quando se faz necessário usar uma determinada função apenas uma vez dentro de um programa é possível digitá-la como parte do programa em si. Porém, quando o programa fizer várias chamadas a uma mesma função para diferentes valores dos argumentos é conveniente personalizar essa função. Uma vez criada e salva uma função, essa nova função adquire os mesmos privilégios das funções nativas do MATLAB.

O modo de personalizar uma função no MATLAB é escrever um programa, salvá-la como um arquivo e usá-la do mesmo modo que uma função nativa. A função pode se constituir em uma única expressão matemática ou em conjunto de expressões complexas, envolvendo uma série de cálculos. Em muitos casos, uma função é uma sub-rotina dentro de um programa no MATLAB. A principal característica de uma função é produzir uma resposta de saída de acordo com uma solicitação de entrada. Isso significa que as operações realizadas dentro da função são desenvolvidas com base nos dados de entrada e os resultados dos cálculos são transferidos para a saída. Tanto a entrada quanto a saída

podem possuir uma ou várias variáveis e cada variável pode ser um escalar, um vetor, uma matriz ou um arranjo genérico. Esquemáticamente, uma função pode ser ilustrada como segue:



Um exemplo de declaração de uma função simples é a função que calcula a altura máxima atingida por uma bola arremessada na vertical com uma certa velocidade inicial: supondo que a velocidade inicial seja denotada por v_0 , a altura máxima é $h_{\max} = \frac{v_0^2}{2g}$, onde g é a aceleração da gravidade. A dependência funcional poderia ser explicitada tal que $h_{\max}(v_0) = \frac{v_0^2}{2g}$. Nesse caso, o dado de entrada é a velocidade inicial (um número) e a saída é a altura máxima (outro número) atingida pela bola. Por exemplo, em unidades do SI, se a entrada vale 15m/s, a saída será 11,47m (para $g = 9,81\text{m/s}^2$). Esquemáticamente:



Uma função declarada e salva como um arquivo no MATLAB pode ser chamada em qualquer parte do MATLAB. Desse modo, programas muito grandes podem ser segmentados em blocos menores passíveis de serem testados independentemente. Nesses casos, as funções assemelham-se às sub-rotinas do Basic e Fortran, aos *procedures* do Pascal e às funções em C/C++.

A declaração de uma função é explicada nas Seções 6.1 a 6.7. Além desse tipo de função declarada e salva como um arquivo independente, será discutido na Seção 6.8 um tipo de função matemática personalizada pelo usuário dentro da própria unidade de código do programa (sem que haja necessidade de criar um arquivo em separado). Isso será feito através do comando `inline`.

6.1 DECLARANDO UMA FUNÇÃO NO MATLAB

Funções são criadas e editadas, como rotinas, na janela Editor/Debugger. Já vimos que essa janela é inicializada a partir da janela Command Window. No menu **File**, selecione **New** e, então clique em **M-file**. Uma vez aberta, a janela Editor/Debugger assemelha-se à Figura 6-1, onde os comandos podem ser digitados um a um, linha após linha. A primeira linha deve conter a linha de declaração da função que será discutida na próxima seção.

6.2 ESTRUTURA DE UMA FUNÇÃO

A estrutura típica de uma função é mostrada na Figura 6-2. Essa função particular foi criada para calcular o pagamento de um empréstimo (as parcelas mensais e o valor total). A função recebe como entradas o valor total do empréstimo, a taxa de juros anual

e o tempo de duração do empréstimo (em anos). A função calcula e exibe na saída o valor parcelado mensalmente e o valor final do empréstimo (corrigido pela taxa de juros). A seguir, vemos a descrição da estrutura de uma função em detalhes.

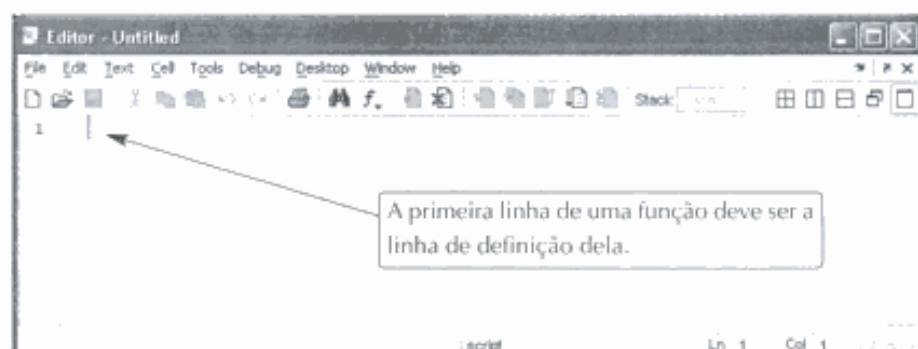


FIGURA 6-1 Janela/Editor Debugger.

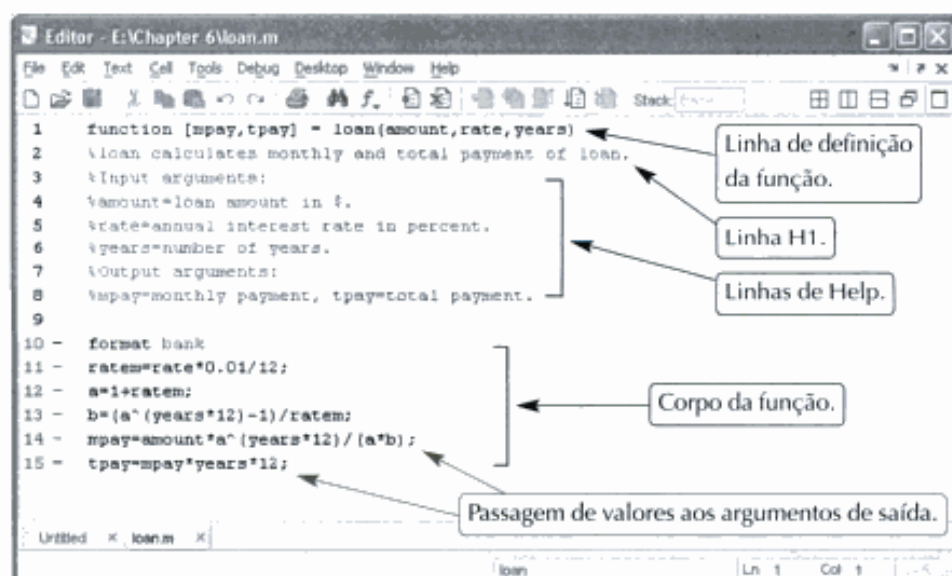


FIGURA 6-2 Estrutura típica de uma função.

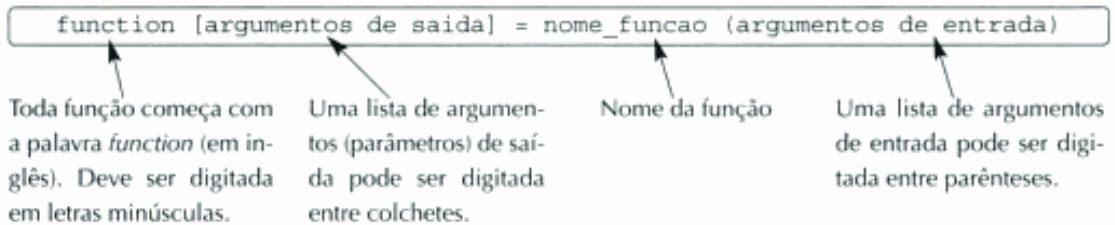
6.2.1 Linha de declaração da função

A primeira linha executável no arquivo **deve** ser a linha de declaração da função. De outro modo, o MATLAB tratará o arquivo como uma rotina. A linha de declaração de uma função:

- Define o arquivo como uma função (diferenciando-o de uma rotina).
- Define o nome da função.

- Define o número e a ordem das variáveis de entrada, além de especificar o que a função irá retornar (variáveis de saída).

O formato característico da linha de declaração de uma função é:



A palavra *function*, digitada em letras minúsculas, deve preceder quaisquer outros objetos (comandos) na declaração de uma função. Você perceberá que, ao digitá-la na janela Editor/Debugger, o MATLAB muda a cor do comando para azul. O nome da função vem logo após o sinal de igualdade (operador de atribuição). Em geral, os nomes característicos de uma função podem conter letras, números e o caractere sublinhar (*underscore*). Ou seja, as regras de atribuição de nomes às funções são as mesmas já comentadas para as variáveis (descritas na Seção 1.6.2). Durante a declaração de funções, é uma prática recomendada evitar a atribuição de nomes de funções nativas e variáveis que fazem parte de um determinado problema e/ou são variáveis/constantes predefinidas pelo MATLAB.

6.2.2 Argumentos de entrada e saída

Os argumentos de entrada e saída são utilizados na passagem de um ou mais parâmetros (dados ou variáveis) para a função. Conforme indicado, os argumentos de entrada seguem logo após o nome da função. Usualmente, toda função tem pelo menos um argumento de entrada, embora seja possível declarar funções que não recebam argumentos de entrada. Entretanto, é mais freqüente declarar funções com múltiplas entradas. Nesses casos, devemos separar os argumentos de entrada por vírgulas. O código de programa que a função desempenha deve, estritamente, ser escrito em termos dos argumentos de entrada, e tais argumentos devem ser inicializados numericamente, antes que a função seja chamada. Significa que as expressões matemáticas na declaração da função precisam ser escritas de acordo com as dimensões dos argumentos, visto que os argumentos são, em geral, grandezas escalares, vetoriais, matriciais ou arranjos genéricos quaisquer. No exemplo mostrado na Figura 6-2, há três argumentos de entrada (*quantia*, *taxa*, *anos*) e, nas expressões matemáticas, eles são usados como escalares. Os valores verdadeiros dos argumentos de entrada de uma função são passados na ocasião da chamada da função na linha do *prompt* da janela Command Window. Se forem esperados argumentos de entrada tipo vetores ou matrizes, as expressões matemáticas no corpo da função devem ser escritas de acordo com as regras da álgebra linear ou de modo a realizar operações elemento por elemento do vetor/matriz.

Os argumentos de saída (quando requeridos) precisam ser listados dentro de colchetes na linha de declaração da função, à esquerda do operador de atribuição, e devem conter os parâmetros cujos valores serão retornados pela função. Entretanto, não é obrigatório que as funções retornem valores. Caso algum retorno seja necessário, um ou vários

parâmetros de saída podem recebê-los ao mesmo tempo. Se houver necessidade de retornar mais de um valor simultaneamente, os parâmetros de retorno devem ser separados mediante o uso de vírgulas. Além disso, se houver apenas um parâmetro de retorno na função é possível digitá-lo sem os colchetes. Em suma, **para que uma função retorne valores corretamente, os argumentos de saída devem possuir os valores das atribuições recebidas quando da execução das linhas de código dentro do corpo da função.** No exemplo da Figura 6-2, existem dois parâmetros de saída [mpag, tpag]. Finalmente, se uma função não possui argumentos de saída, o operador de atribuição na linha de declaração da função pode ser omitido. Uma função que não retorna valores pode, por exemplo, gerar um gráfico ou salvar dados em um arquivo de saída.

É possível ainda passar *strings* para uma função. Basta digitar a *string* como parte das variáveis de entrada (o texto deve aparecer entre aspas simples). Frequentemente, as *strings* são utilizadas para passar nomes de outras funções, nativas ou personalizadas, para dentro de uma determinada função.

Usualmente, todas os parâmetros de entrada e saída de uma função são passados ou retornados à ou pela função, respectivamente. Entretanto, todas as características de entrada e saída estudadas para as rotinas (Capítulo 4) são válidas e podem ser utilizadas em funções. Isso significa, por exemplo, que toda variável declarada nas linhas de código da função terá o conteúdo exibido na tela, a menos que seja digitado ponto-e-vírgula no fim do comando. Adicionalmente, o comando `input` pode ser usado para passar dados interativamente a uma função e os comandos `disp`, `fprintf` e `plot` podem exibir informações na tela, salvar um arquivo ou plotar figuras, assim como em uma rotina. A seguir, alguns exemplos de linhas de declaração de funções, com diferentes combinações de argumentos de entrada e saída.

Linha de declaração da função	Comentários
<code>function [mpag, tpag] = emprestimo (quantia, taxa, anos)</code>	Três argumentos (parâmetros) de entrada e dois parâmetros de saída.
<code>function [A] = RetArea(a,b)</code>	Dois argumentos de entrada e um parâmetro de saída.
<code>function A = RetArea(a,b)</code>	Mesma função declarada acima. Função com um único parâmetro de saída pode ser digitada sem os colchetes.
<code>function [V,S] = EsferaVolArea(r)</code>	Uma variável de entrada e duas variáveis de saída.
<code>function trajetoria(v,h,g)</code>	Três parâmetros de entrada e nenhum parâmetro de saída.

6.2.3 A linha de descrição da função (linha H1) e as linhas de comentários

A linha H1 e as linhas de comentários são precedidas do caractere %, porque não fazem parte do código do programa em si, e são escritas logo após a linha de declaração da função. Uma função não precisa contê-los, mas é praxe usá-los para tornar inteligível a

codificação da função. Geralmente, a linha H1 é a primeira linha de comentário da função e recomenda-se que ela faça menção ao nome da função, enfatizando a funcionalidade da mesma. Quando o usuário digita `lookfor uma_palavra_chave` na linha do *prompt* da janela Command Window, o MATLAB procura a palavra chave requerida em todas as linhas H1 de todas as funções nativas e personalizadas, e, se a palavra procurada estiver em alguma linha H1, o MATLAB exibirá o conteúdo dessa linha.

Todas as demais linhas de comentários devem suceder a linha H1. Essas linhas geralmente trazem um descritivo completo da função e instruções sobre os argumentos de entrada e saída da função. Tais linhas de comentários, junto com a linha H1, podem ser vislumbradas quando o usuário digita `help nome_da_função` na janela Command Window. As linhas H1 e de comentários das funções nativas do MATLAB também podem ser vislumbradas por meio do comando `help`. Por exemplo, para a função *empréstimo* da Figura 6-2, digitando `help empréstimo` na linha do *prompt* da janela Command Window, desde que o caminho onde se encontra o arquivo da função seja especificado no campo Current Directory e que a função tenha sido criada, o resultado será:

```
>> help empréstimo
```

A função *empréstimo* calcula os valores mensal e total do empréstimo.

Argumentos de entrada:

quantia = valor tomado emprestado em \$.

taxa = taxa de juros anual em percentagem.

anos = tempo de vigência do empréstimo (em anos).

Argumentos de saída:

mpag = parcela mensal, *tpag* = pagamento total.

É claro que uma função pode possuir linhas de comentários no corpo da função. Essas linhas de comentário são ignoradas pelo comando `help`.

6.2.4 O corpo da função

O corpo da função deve conter o código de programa necessário à execução, cálculo e inicialização de todos os parâmetros da função. As linhas de código podem trazer tudo o que o MATLAB tem a oferecer: operadores de aritméticos, lógicos e atribuição e relacionais (veja o Capítulo 7), funções nativas ou personalizadas, controle de fluxo (sentenças condicionais ou laços, explicado no Capítulo 7), comentários, linhas em branco, entradas e saídas interativas, etc.

6.3 VARIÁVEIS LOCAIS VERSUS VARIÁVEIS GLOBAIS

Todas as variáveis de uma função são locais, i.e., os valores dos parâmetros de entrada e saída estão definidos e são reconhecidos somente dentro da estrutura da função. Quando a

função é chamada, o MATLAB se utiliza de um espaço de memória diferente do espaço ocupado pela área de trabalho (o espaço de memória onde é executada a janela Command Window e as rotinas). Em uma função, as variáveis de entrada são inicializadas toda vez que a função é chamada na linha do *prompt*. Assim, as variáveis podem ser utilizadas em cálculos dentro do corpo da função. Terminada a execução da função, os valores dos argumentos de saídas são passados às variáveis utilizadas para chamar a função. Logo, uma função pode possuir variáveis com nomes idênticos às variáveis declaradas na janela Command Window ou em uma rotina. Em síntese, a chamada de uma função não irá indicar um conflito entre variáveis internas e externas à mesma. Portanto, os valores atribuídos às variáveis internas às funções não serão modificados por nenhum tipo de atribuição feito fora do ambiente da função.

Toda função possui um conjunto particular de variáveis locais que não são compartilhadas com outras funções ou com a área de trabalho da janela Command Window e/ou das rotinas. Entretanto, é possível tornar uma variável comum (pública) às várias funções e, possivelmente, a toda a área de trabalho. Para declarar uma variável global é necessário especificar o comando `global` utilizando-se a seguinte sintaxe:

```
global = Nome_variavel
```

A declaração múltipla de variáveis globais pode ser feita listando e separando as variáveis por meio de espaços após um único comando `global`. Por exemplo:

```
global ACELERA_GRAVIDADE Coef_Atrito
```

- A variável tem que ser declarada global em toda função que o usuário quiser que ela seja reconhecida, em todas as partes do MATLAB. Desse modo, a variável torna-se comum a essas funções.
- O comando `global` deve aparecer antes de a variável ser utilizada. Recomenda-se empregar esse comando no início da função (após as linhas de declaração, H1 e dos comentários).
- O comando `global` deve ser digitado na janela Command Window e/ou dentro da rotina, para que a variável seja reconhecida na área de trabalho (*workspace*).
- O valor da variável pode ser atribuído ou modificado em qualquer parte comum do domínio de validade dessa variável.
- É recomendado usar nomes descritivos extensos (ou usar letras maiúsculas) para variáveis globais, de maneira a distingui-las das variáveis locais.

6.4 SALVANDO UMA FUNÇÃO

Uma função deve ser salva antes que se faça uso dela. Semelhantemente às rotinas, escolha a opção **Save as...** no menu **File**, selecionando a localização (muitos estudantes salvam no *drive A:*, no *zip drive*, em uma unidade USB ou em uma unidade de rede) e digitando o nome do arquivo a ser identificado como função. É recomendável que o arquivo seja salvo com um nome idêntico ao da função, sugerido pelo MATLAB no ato de salvá-la. Desse modo, chamar a função fica bastante fácil para o aluno. (Se uma função foi sal-

va com um nome diferente, o nome dado ao arquivo deve ser utilizado quando a função for chamada.) Os arquivos de funções são salvos com a extensão `.m`. Exemplos:

Linha de declaração da função	Nome do arquivo
<code>function [mpag, tpag] = emprestimo (quantia, taxa, anos)</code>	<code>emprestimo.m</code>
<code>function [A] = RetArea(a,b)</code>	<code>RetArea.m</code>
<code>function [V,S] = EsferaVolArea(r)</code>	<code>EsferaVolArea.m</code>
<code>function trajetoria(v,h,g)</code>	<code>trajetoria.m</code>

6.5 CHAMANDO UMA FUNÇÃO

Uma função personalizada é utilizada do mesmo modo que as funções nativas do MATLAB. A função pode ser chamada na janela Command Window, em uma rotina ou em uma outra função. Para usar a função, o diretório (pasta) onde foi salva a função precisa ser especificado no campo Current Directory ou em um caminho específico utilizado pelo usuário (veja as Seções 4.3.1 e 4.3.2).

Uma função muitas vezes é utilizada para atribuir um valor a uma ou mais variáveis, como parte de uma expressão matemática, como um argumento em outra função, em rotinas ou, simplesmente, usada como um comando na linha do *prompt* da janela Command Window. Em todos os casos, o usuário deve ter em mente quais são os argumentos (parâmetros) de entrada e saída. Em geral, um argumento de entrada são valores/expressões numéricas ou, então, uma variável previamente inicializada. Os parâmetros vão sendo utilizados pelo MATLAB de acordo com a posição que ocupam na lista de argumentos de entrada e saída na linha de declaração da função.

Os dois modos de utilização de uma função estão ilustrados a seguir, baseados na função `emprestimo` que calcula as parcelas mensais e o valor total de um empréstimo (os dois argumentos de saída). Os argumentos de entrada são a quantia tomada emprestada, a taxa de juros anual e o período de financiamento do empréstimo (variável `anos`). No primeiro exemplo, a função `emprestimo` recebe diretamente os parâmetros numéricos de entrada na linha do *prompt*:

```
>> [mes total]=emprestimo(25000,7.5,4)
mes =
600.72
total =
28834.47
```

Primeiro argumento é a quantia tomada emprestada, o segundo é a taxa de juros e o terceiro é o tempo de duração do empréstimo (em anos).

Já no segundo exemplo, a função `emprestimo` utiliza as variáveis `a` e `b`, previamente declaradas, mais uma constante numérica correspondente ao tempo total de financiamento (`anos`).


```
>> a=70000; b=6,5;
```

```
>> [x y]=emprestimo(a,b,30)
```

```
x =
```

```
440.06
```

```
y =
```

```
158423.02
```

Declara as variáveis a e b.

Usa as variáveis a, b e o tempo de duração do empréstimo (30) como argumentos de entrada; x (parcela mensal) e y (pagamento total) como argumentos de saída.

6.6 EXEMPLOS DE FUNÇÕES SIMPLES

Problema-exemplo 6-1: Função matemática

Escreva uma função que retorne o resultado da função $f(x) = \frac{x^4 \sqrt{3x+5}}{(x^2+1)^2}$. Salve-a como

Capitulo6_Exemplo1. O parâmetro de entrada da função é a variável x e o parâmetro de saída é a função $f(x)$. Escreva a função de modo que a variável x seja um vetor. Teste da função nos seguintes casos:

a) $f(x)$ para $x = 6$

b) $f(x)$ para $x = 1, 3, 5, 7, 9$ e 11

Solução

A declaração da função correspondente à função matemática $f(x)$ é:

```
function y = Capitulo6_Exemplo1(x)
```

Linha de declaração da função.

```
y = (x.^4.*sqrt(3*x+5))./(x.^2+1).^2;
```

Atribuição da expressão de $f(x)$ ao argumento de saída (y).

Perceba que a expressão matemática no corpo da função exige cálculos envolvendo elemento por elemento do vetor. Desse modo, se x é um vetor, y também será um vetor. A função deve ser salva e o diretório (pasta) onde ela se encontra deve ser indicado no campo Current Directory. Teste a função na janela Command Window:

a) Para calcular a função em $x = 6$, basta digitar `Capitulo6_Exemplo1(6) + ENTER` na linha do *prompt* que o resultado é exibido na saída padronizada (ans), ou então, atribuído a uma nova variável:

```
>> Capitulo6_Exemplo1(6)
```

```
ans =
```

```
4.5401
```

```
>> F=Capitulo6_Exemplo1(6)
```

```
F =
```

```
4.5401
```

- b) Para calcular a função se o argumento x é um vetor, basta criar e inicializar o vetor x e, então, utilizá-lo como argumento da função `Capitulo6_Exemplo1(x)`.

```
>> x = 1:2:11
x =
    1    3    5    7    9   11
>> Capitulo6_Exemplo1(x)
ans =
    0.7071    3.0307    4.1347    4.8971    5.5197    6.0638
```

O outro modo é digitar o vetor x diretamente no argumento da função `Capitulo6_Exemplo1(x)`.

```
H = ([1:2:11])
H =
    0.7071    3.0307    4.1347    4.8971    5.5197    6.0638
```

Problema-exemplo 6-2: Convertendo escalas de temperatura

Escreva uma função que converta temperaturas em graus Fahrenheit (F) em graus Celsius (C). Salve a função como `FparaC` e utilize-a para resolver o seguinte problema: a variação no comprimento de um objeto (ΔL), devido à variação de temperatura (ΔT), é dada por: $\Delta L = \alpha L \Delta T$, onde α é o coeficiente de dilatação linear e L é o comprimento do objeto. Determine a variação na área de uma chapa de alumínio de dimensões $4.5\text{m} \times 2.25\text{m}$ ($\alpha = 23 \cdot 10^{-6} \text{ } ^\circ\text{C}^{-1}$), se a temperatura variar de 40°F para 92°F .

Solução

A seguir é apresentada a declaração da função que converte a escala Fahrenheit na escala Celsius:

```
function C = FparaC(F)
% FparaC converte a escala Fahrenheit na escala Celsius
C = 5*(F-32)/9;
```

Linha de declaração da função.

Atribuição da expressão ao parâmetro de saída C.

Em seguida, é mostrada uma rotina (salva como `Capitulo6_Exemplo2`) para calcular a variação na área da chapa devido à mudança na temperatura:

```
a1 = 4.5; b1 = 2.25; T1 = 40; T2 = 92; alfa = 23e-6;
deltaT = FparaC(T2) - FparaC(T1);
```

Usando a função `FparaC` para calcular a diferença de temperatura em graus Celsius.


```

a2 = a1 + alfa*a1*deltaT;
b2 = b1 + alfa*b1*deltaT;
VarArea = a2*b2 - a1*b1;
fprintf('A variacao na area da chapa e %6.5f m2.', VarArea)

```

Calculando o novo comprimento da chapa.

Calculando a nova largura da chapa.

Calculando a variação total na área da chapa.

Executando a rotina na linha do *prompt* da janela Command Window, temos:

```

>> Capitulo6_Exemplo2
A variação na area da chapa é 0.01346m2.

```

6.7 COMPARAÇÃO ENTRE ROTINAS E FUNÇÕES

À primeira vista, alguns estudantes do MATLAB têm dificuldades em compreender quais são exatamente as diferenças entre rotinas e funções, visto que, em muitos problemas, ambas as formas conduzem a uma solução rápida e elegante. As principais diferenças (e semelhanças) entre rotinas e funções podem ser resumidas como segue:

- Ambas são salvas com a extensão .m (motivo pelo qual, às vezes, as denominamos M-files).
- A primeira linha de toda função é a linha de declaração da função.
- As variáveis em uma função são sempre locais. As variáveis de uma rotina são reconhecidas na área de trabalho (Command Window).
- Rotinas podem fazer uso de variáveis declaradas na área de trabalho.
- Rotinas possuem seqüências de comandos (sentenças) do MATLAB.
- Funções podem receber dados de entrada, por meio de argumentos (parâmetros) de entrada, e podem retornar dados para os argumentos de saída.
- Para o caso de funções, é recomendável salvar o arquivo com um nome idêntico ao da função.

6.8 FUNÇÕES inline

Funções são utilizadas para resolver problemas matemáticos simples ou complexos e que requerem programação extensiva, ou são desenhadas para cumprir uma tarefa particular dentro de programas maiores. Nesses casos, quando o valor de uma função matemática precisa ser determinado dentro de um programa, o MATLAB oferece o recurso das funções *inline*. Uma função *inline* é definida como parte da unidade de código de programa que a contém (ou seja, não é um arquivo .m em separado) e pode ser empregada apenas dentro dessa unidade de código. As funções *inline* podem ser declaradas em rotinas, nas declarações de funções, programas escritos no MATLAB (Capítulo 7) ou na janela Command Window.

Todas as funções *inline* são criadas por meio do comando `inline`, seguindo a sintaxe:

```
Nome_funcao = inline('expressao matematica digitada
                     como uma string')
```

- A expressão matemática pode ter uma ou mais variáveis independentes.
- Quaisquer letras, exceto *i* e *j*, podem ser designadas às variáveis independentes na expressão.
- A expressão matemática pode trazer funções nativas ou personalizadas.
- A expressão matemática deve ser escrita de acordo com as dimensões do argumento (i.e., para realizar operações elemento por elemento ou seguindo as regras da álgebra linear).
- As expressões não podem conter variáveis previamente inicializadas.
- Uma vez definida uma função `inline` é possível chamá-la pelo nome, atribuindo-lhe valor(es) ao(s) argumento(s) (veja exemplo).
- Uma função `inline` pode ser utilizada como argumento de outras funções.

Função inline com uma variável independente:

Por exemplo, a função $f(x) = \frac{ex^2}{\sqrt{x^2 + 5}}$ definida na janela Command Window como uma função `inline`, sendo x um escalar:

```
>> FA = inline('exp(x^2)/sqrt(x^2 + 5)')
FA =
    Inline function:
    FA(x) = exp(x^2)/sqrt(x^2 + 5)
>>
```

Se, na declaração da função `FA`, não for adicionado um ponto-e-vírgula no final, o MATLAB esconde a informação de que `FA` é uma função `inline`. Feita a declaração, é possível chamá-la para diferentes valores de x , conforme indicado a seguir:

```
>> FA(2)
ans =
    18.1994
>> z = FA(3)
z =
    2.1656e+003
>>
```

Sendo esperados múltiplos valores simultâneos para x , a declaração da função `FA` deve levar em conta que x é um vetor, matriz ou um arranjo qualquer, isto é, devem ser realizadas operações elemento por elemento com a variável x .

```
>> FA = inline('exp(x.^2)./sqrt(x.^2 + 5)')
FA =
Inline function:
FA(x) = exp(x.^2)./sqrt(x.^2 + 5)
>> FA([1 0.5 2])
ans =
1.1097 0.5604 18.1994
```

Entrando com um vetor no argumento.

Função inline com muitas variáveis independentes:

O comando `inline` para funções com duas ou mais variáveis independentes pode ser escrito utilizando-se a mesma sintaxe anterior ou a sintaxe a seguir:

```
Nome_funcao = inline('Expressao matematica','arg1',
                    'arg2','arg3')
```

Na sintaxe acima, a ordem de entrada dos argumentos da função é definida pelo usuário. Se o formato anterior for utilizado, o MATLAB se encarrega de ordenar as variáveis alfabeticamente*. Por exemplo, a função $f(x,y) = 2x^2 - 4xy + y^2$ escrita como uma função `inline` seria:

```
>> HA = inline('2*x^2-4*x*y+y^2')
HA =
Inline function:
HA(x,y) = 2*x^2-4*x*y+y^2
```

Feita a declaração da função, é possível testá-la na linha do *prompt* para diferentes valores x e y . Por exemplo, `HA(2,3)` resulta em:

```
>> HA(2,3)
ans =
-7
```

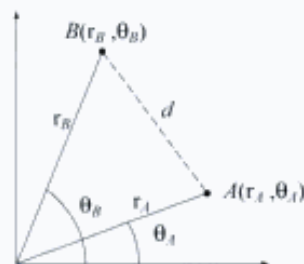
O Problema-exemplo 6-3, a seguir, traz um outro exemplo de função `inline` com muitas variáveis.

Problema-exemplo 6-3: Distância entre dois pontos em coordenadas polares

Escreva uma função `inline` que calcule a distância entre dois pontos no plano, escritos em coordenadas polares. Em seguida, use a função `inline` para calcular a distância entre o ponto $A(2, \pi/6)$ e o ponto $B(5, 3\pi/4)$.

Solução

A forma mais fácil de se determinar a distância entre dois pontos em coordenadas polares é fazendo uso da lei dos co-senos:



* N. de T.: Seguindo o alfabeto inglês.

$$d = \sqrt{r_A^2 + r_B^2 - 2r_A r_B \cos(\theta_A - \theta_B)}$$

Primeiramente, devemos declarar a distância d como uma função `inline` a quatro parâmetros (variáveis). Em seguida, a função pode ser testada no cálculo da distância entre os pontos A e B , sugeridos no enunciado do problema.

```
d = inline('sqrt(rA^2 + rB^2 - 2*rA*rB*cos(tetaB - tetaA))',...
'rA','tetaA','rB','tetaB')
```

A ordem dos argumentos é definida por (rA, tetaA, rB, tetaB).

```
d =
```

Inline function:

```
d(rA,tetaA,rB,tetaB) = sqrt(rA^2 + rB^2 - 2*rA*rB*cos(tetaB - tetaA))
```

```
>> DistA_B = d(2,pi/6,5,3*pi/4)
```

```
DistA_B =
```

```
5.8461
```

Os argumentos (variáveis) são digitados(as) na ordem definida na função.

6.9 O COMANDO `feval`

O comando `feval` testa o valor de uma função para um certo valor ou conjunto de valores do(s) argumento(s). A sintaxe do comando é:

```
Variavel = feval ('Nome_funcao', valor do argumento)
```

É possível atribuir o valor determinado pelo comando `feval` a uma variável ou, se o argumento for digitado sem o operador de atribuição, o MATLAB pode exibir `ans` = e resultado do teste da função. Em suma:

- O nome da função entra como uma *string*.
- A função pode ser nativa ou personalizada.
- Se houver mais de um argumento de entrada, devem ser separados por vírgula.
- Se houver mais de um argumento de saída, as variáveis no lado esquerdo do operador de atribuição devem ser digitadas dentro de colchetes e separadas por vírgulas.

O exemplo a seguir mostra o uso do comando `feval` com funções nativas do MATLAB:

```
>> feval('sqrt',64)
```

```
ans =
```

```
8
```

```
>> x=feval('sin',pi/6)
```

```
x =
```

```
0.5000
```

No exemplo a seguir, o comando `feval` é utilizado na função `emprestimo` criada no início do capítulo (Figura 6-2).

```
>> [M,T]=feval('emprestimo',50000,3.9,10)
M =
    502.22
T =
    60266.47
```

Parcelas mensais.

Pagamento total.

Empréstimo de \$50,000, a uma taxa de 3.9% em 10 anos.

Quando usar o comando `feval`:

Nos exemplos dados, o comando `feval` poderia ter sido preterido para testar a função diretamente, substituindo os valores nos argumentos. Para o último exemplo, digitando-se:

```
>> [M,T]=emprestimo(50000,3.9,10)
M =
    502.22
T =
    60266.47
```

resultaria nas mesmas saídas para `M` e `T`. Parece não haver necessidade de usar o comando `feval` para determinar um ou mais valores de uma função, quando a própria função pode ser chamada para essa finalidade.

O comando `feval` é bastante útil no caso de uma função precisar ser avaliada dentro de outra função, estando a função interna sujeita a variações para os diferentes testes da função principal. Isso é possível porque a *string* que identifica a função interna pode ser importada para a função principal por meio dos argumentos dessa função.

Por exemplo, o MATLAB possui uma função nativa denominada `fzero` que determina o(s) zero(s) de uma função a uma variável independente. A função `fzero` pode ser utilizada para resolver uma vasta gama de funções diferentes, bastando para isso transferir o nome da função a ser resolvida para a função `fzero` através dos parâmetros de entrada. A função `feval` poderia agir no código de programa de `fzero` para determinar os valores da função a ser resolvida. A função `fzero` é explicada em detalhes no Capítulo 10.

6.10 EXEMPLOS DE APLICAÇÕES DO MATLAB

Problema-exemplo 6-4: Dinâmica populacional e decaimento radioativo

Um modelo para o crescimento populacional e/ou decaimento radioativo é dado por:

$$A(t) = A_0 e^{kt}$$

onde $A(t)$ e A_0 se referem às quantidades estudadas em um tempo t e $t = 0$, respectivamente, e k é uma constante característica (única) para cada aplicação.

Escreva uma função que, utilizando o modelo acima, determine a quantidade $A(t)$ para o tempo t , conhecidos A_0 e $A(t_1)$. Sugestão para o nome da função e dos argumentos: $At = \text{expCD}(A0, At1, t1, t)$, sendo At o argumento de saída, correspondendo a $A(t)$, e os argumentos $A0, At1, t1, t$ correspondendo a $A_0, A(t_1), t_1$ e t , respectivamente.

Teste a função na linha do *prompt* da janela Command Window para resolver os seguintes problemas:

- A população do México era de 67 milhões em 1980 e de 79 milhões em 1986. Estime a população no ano 2000.
- A meia-vida de um material radioativo é 5.8 anos. Após 30 anos, quanto restará de uma amostra de 7 gramas desse material?

Solução

Primeiramente, para se utilizar o modelo de crescimento populacional, deve ser determinado o valor da constante k , resolvendo-se a equação para k em termos de $A_0, A(t_1)$ e t_1 :

$$k = \frac{1}{t_1} \ln \frac{A(t_1)}{A_0}$$

Assim, fica determinado o valor de k e o modelo pode ser utilizado para determinar a população em um tempo t qualquer.

Uma função que resolve o problema acima é:

```
function At = expCD(A0,At1,t1,t)
```

Linha de declaração da função.

```
% A funcao expCD determina o crescimento ou decaimento exponencial.
```

```
% Os argumentos (parametros) de entrada sao:
```

```
% A0: quantidade no tempo zero.
```

```
% At1: quantidade no tempo t1.
```

```
% t1: tempo t1.
```

```
% t: tempo t.
```

```
% Os argumentos de saida sao:
```

```
% At: quantidade no tempo t.
```

```
k = log(At1/A0)/t1;
```

Determinação de k .

```
At = A0*exp(k*t);
```

Determinação de $A(t)$. (Atribuição do valor à variável de saída.)

Uma vez salva a função, basta alternar para a janela Command Window e testar os dois casos.

Primeiro caso: a) $A_0 = 67$, $A(t_1) = 79$, $t_1 = 6$ e $t = 20$:


```
>> expCD(67,79,6,20)
```

```
ans =
```

```
116.0332
```

Estimativa da população no ano 2000.

Segundo caso: $b) A_0 = 7, A(t_1) = 3.5$ (tomando t_1 como o tempo de meia-vida, que é o tempo necessário para que o material radioativo decaia à metade da quantidade inicial), $t_1 = 5.8$ e $t = 30$.

```
>> expCD(7,3.5,5.8,30)
```

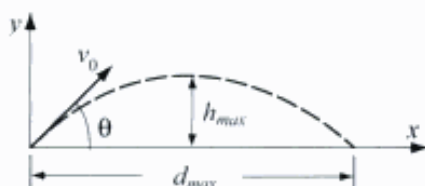
```
ans =
```

```
0.1941
```

A quantidade de material após 30 anos.

Problema-exemplo 6-5: Movimento de um projétil

Crie uma função para determinar a trajetória de um projétil. As variáveis de entrada são a velocidade inicial e o ângulo de lançamento do projétil. Os argumentos de saída são a altura máxima atingida ($h_{\max}$) e a distância de alcance ($d_{\max}$) do projétil. Além disso, a função deve plotar um gráfico da trajetória do projétil. Por fim, teste a função para calcular a trajetória de um projétil lançado com uma velocidade inicial de 230m/s em um ângulo de 39° .



Solução

O movimento de um projétil pode ser analisado com base nas componentes horizontal e vertical da velocidade. A velocidade inicial v_0 é decomposta em suas componentes como segue:

$$v_{0x} = v_0 \cos(\theta) \quad \text{e} \quad v_{0y} = v_0 \sin(\theta)$$

Na direção vertical, a velocidade e a posição do projétil são dados por:

$$v_y = v_{0y} - gt \quad \text{e} \quad y = v_{0y}t - \frac{1}{2}gt^2$$

O tempo para que o projétil atinja a altura máxima ($v_y = 0$) e o valor correspondente da altura são dados por:

$$t_{h_{\max}} = \frac{v_{0y}}{g} \quad \text{e} \quad h_{\max} = \frac{v_{0y}^2}{2g}$$

O tempo total de voo do projétil* é duas vezes o tempo necessário para que ele alcance o ponto vertical mais alto (altura máxima), i.e., $t_{\text{tot}} = 2t_{h_{\max}}$. Na direção horizontal, a velocidade é constante e a posição do projétil é dada por:

$$x = v_{0x}t$$

* N. de T.: Desde que o atrito entre o projétil e as partículas do meio (o ar) seja desprezado.

Sugestão de notação para a função no MATLAB: `[hmax,dmax] = trajetoria(v0,teta)`. A seguir, temos uma função exemplo para o problema:

```
function [hmax,dmax] = trajetoria(v0,teta)
% A funcao trajetoria determina a altura maxima e a distancia de alcance de
% um projatil. Em seguida, plota o grafico da funcao.
% Os argumentos de entrada sao:
% v0: velocidade inicial em (m/s).
% teta: angulo em graus.
% Os argumentos de saida sao:
% hmax: altura maxima do projatil em (m).
% dmax: distancia maxima de alcance do projatil em (m).
% A funcao esboça o grafico da funcao trajetoria.
g = 9.81;
v0x = v0*cos(teta*pi/180);
v0y = v0*sin(teta*pi/180);
thmax = v0y/g;
hmax = v0y^2/(2*g);
ttot = 2*thmax;
dmax = v0x*ttot;
% Criando um grafico da funcao trajetoria.
tplot = linspace(0,ttot,200);
x = v0x*tplot;
y = v0y*tplot - 0.5*g*tplot.^2;
plot(x,y)
xlabel('DISTANCIA (m)')
ylabel('ALTURA (m)')
title('TRAJETORIA DE UM PROJETIL')
```

Linha de declaração da função.

Criando um vetor tempo com 200 elementos.

Calculando as coordenadas x e y do projétil para cada tempo.

Multiplicação elemento por elemento do vetor.

Após salvar a função, teste-a na linha do *prompt* da janela Command Window com os argumentos desejados, ou seja, $v_0 = 230\text{m/s}$ e ângulo 39° .

```
>> [h d] = trajetoria(230,39)
h =
    1.0678e+003
d =
    5.2746e+003
```

Além disso, a função gera o gráfico solicitado na janela Figure Window:



6.11 PROBLEMAS

- Escreva uma função no MATLAB que receba dois parâmetros de entrada, a altura e o peso de uma pessoa, escritos no Sistema Britânico (SB), e retorne outros dois parâmetros de saída, a altura e o peso da pessoa convertidos para o Sistema Internacional de unidades (SI). Utilize a seguinte convenção de nomes para a função e argumentos: `[cm, kg] = SbparaSI(in, lb)`. Teste a função na janela Command Window para:
 - Determinar a altura e o peso no SI de uma pessoa que mede 5 pés e 10 pol e que pesa 175 lb.
 - Determinar a altura e o peso de um objeto que mede 1 pol e pesa 1 lb.
- Declare uma função no MATLAB que realize a seguinte função matemática:

$$y(x) = 0.9x^4 - 12x^2 - 5x$$

A função deve receber a variável x e retornar y . Escreva a função de tal modo que a variável x seja um vetor. Em seguida, utilize-a para resolver:

- a função para $y(-3)$ e $y(5)$.
 - o gráfico da função $y(x)$ quando $-4 \leq x \leq 4$.
- Declare uma função no MATLAB para realizar a seguinte função matemática:

$$r(\theta) = 2(1.1 - \sin^2\theta)$$

A função deve receber a variável θ (em radianos) e retornar r . Escreva a função de tal modo que a variável θ seja um vetor.

- Use a função para encontrar $r(\pi/3)$ e $r(3\pi/2)$.
- Use a função para plotar o gráfico polar de $r(\theta)$ quando $0 \leq \theta \leq 2\pi$.

4. Escreva uma função no MATLAB que calcule o máximo ou o mínimo local da função quadrática escrita na forma: $f(x) = ax^2 + bx + c$. Sugestão para o nome da função e dos argumentos: $[x, y] = \text{maxmin}(a, b, c)$. Perceba que os parâmetros de entrada são as constantes a, b e c ; os parâmetros de saída são as coordenadas x e y do ponto de máximo (ou do mínimo) local.

Teste a função para determinar o ponto máximo (ou o mínimo) das seguintes funções matemáticas:

a) $f(x) = 3x^2 - 18x + 48$

b) $f(x) = -5x^2 + 10x - 3$

5. O saldo P de uma caderneta de poupança, após t anos, feito um investimento inicial P_0 e sendo a taxa de juros anual r (em %), é dado por:

$$P = P_0 \left(1 + \frac{r}{100} \right)^t$$

Escreva uma função no MATLAB para calcular o saldo de uma caderneta de poupança. Utilize a seguinte notação para a função e os parâmetros de entrada e saída: $P = \text{poup}(P_0, r, t)$. Note que a função deve receber como parâmetros o investimento inicial, a taxa de juros e o tempo de aplicação em anos. A saída será o saldo da caderneta de poupança. Em seguida, considere um investimento inicial de \$ 10,000, a uma taxa de juros anual de 6%. Teste a função para determinar o saldo da caderneta após 13 anos de aplicação.

6. Escreva uma função no MATLAB que converta torque em lb.in para N.m. Utilize a seguinte notação para a função: $Nm = \text{lb_in_Nm}(lb_in)$. O parâmetro de entrada é o torque em lb.in e o parâmetro de saída é o torque em N.m. Teste a função para converter 500 lb.in em N.m.
7. Escreva uma função no MATLAB que determine os ângulos internos de um triângulo quando são fornecidos os comprimentos dos lados. Sugestão para o nome da função: $[alf, bet, gam] = \text{triangulo}(a, b, c)$. Teste a função para determinar os ângulos dos triângulos:
- a) $a = 10, b = 15, c = 7$
- b) $a = 6, b = 8, c = 10$
- c) $a = 200, b = 75, c = 250$

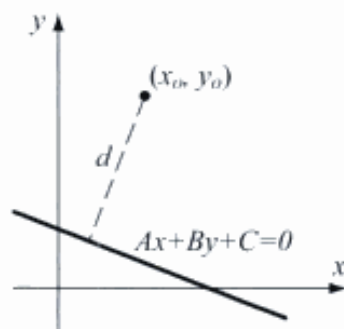
8. Declare uma função no MATLAB que determine o vetor unitário na direção da reta que une dois pontos (A e B) no espaço. Sugestão para o nome da função: $n = \text{vetunit}(A, B)$. A função deve receber dois vetores A e B , cada um com três elementos, correspondendo às coordenadas cartesianas do ponto em questão. A função deve retornar um vetor com três componentes que corres-

ponde ao vetor unitário na direção da reta que liga os pontos A e B . Utilize a função para determinar os seguintes vetores unitários:

- na direção da reta que interliga os pontos $(2, 6, 5)$ e $(-10, 15, 9)$.
- na direção da reta que interliga os pontos $(-10, 15, 9)$ e $(2, 6, 5)$.
- na direção da reta que interliga os pontos $(1, 1, 2)$ e $(2, 1, 1)$.

9. A forma geral da equação de uma reta no plano xy é: $Ax + By + C = 0$. Um ponto nesse plano é definido pelas coordenadas (x_0, y_0) .

Escreva uma função no MATLAB para determinar a distância de um ponto a uma reta no plano xy . Utilize a seguinte notação para a função `d = distPreta(xo, yo, A, B, C)`, sendo os argumentos de entrada as coordenadas do ponto e as três constantes da equação da reta. A variável retornada é a distância. Teste a função para determinar a distância nos seguintes casos:



- ponto: $(2, -4)$, reta: $-2x + 3,5y - 6 = 0$.
- ponto: $(11, 2)$, reta: $y = -2x + 6$.

10. Crie uma função que calcule a nota final em uma disciplina de um grupo de estudantes com base em cinco trabalhos, no desempenho em duas provas intermediárias e uma prova final. As provas intermediárias são avaliadas em uma escala de 0 a 100 e cada uma corresponde a 20% da nota final. A prova final também é avaliada em uma escala de 0 a 100, mas corresponde a 40% da nota final. Os trabalhos são avaliados em uma escala de 0 a 10. Todos os trabalhos juntos correspondem a 20% da nota final.

Sugestão para o nome da função e dos argumentos: `notafinal = notasf(R)`. Os argumentos de entrada devem estar dispostos numa matriz R , onde os elementos em cada linha são as notas de um determinado estudante. As cinco primeiras colunas correspondem às notas dos trabalhos (notas de 0 a 10), as duas próximas colunas devem trazer as notas das provas intermediárias (notas de 0 a 100) e, a última coluna, a nota final (nota de 0 a 100). O parâmetro de saída da função (`notafinal`) é um vetor coluna contendo as notas finais do grupo, cujas linhas representam as notas dos estudantes organizadas na matriz R .

A função pode ser utilizada para calcular as notas finais de qualquer quantidade de estudantes. Para um único estudante, a matriz R reduz-se a um vetor linha.

- Use a linha do *prompt* da janela Command Window para determinar a nota final de um estudante com o seguinte conjunto de notas: 10, 5, 8, 7, 9, 75, 87, 69.

- b) Escreva uma rotina que receba as notas dos estudantes e armazene em uma matriz (as notas de cada estudante entram em uma linha). Em seguida, o programa calcula as notas finais usando a função `notasf`. Teste a rotina na janela Command Window para calcular as notas do seguinte grupo de estudantes:

Estudante A: 7, 9, 5, 8, 10, 90, 70, 85.

Estudante B: 6, 4, 7, 0, 7, 60, 71, 50.

Estudante C: 5, 9, 10, 3, 5, 45, 75, 80.

Estudante D: 8, 8, 7, 7, 9, 82, 81, 88.

11. Quando n resistores são conectados para formar um circuito paralelo, a resistência equivalente R_{eq} (ou resistência total R_T) desse circuito pode ser determinada por:

$$\frac{1}{R_{eq}} = \frac{1}{R_1} + \frac{1}{R_2} + \dots + \frac{1}{R_n}$$

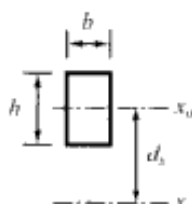
Escreva uma função no MATLAB que determine a R_{eq} . Sugestão para o nome da função e dos argumentos é: `REQ = req(R)`. Os parâmetros de entrada da função devem estar dispostos em um vetor, onde cada entrada desse vetor corresponde à resistência de um resistor, e o parâmetro de saída é R_{eq} . Teste a função para determinar a resistência equivalente quando os seguintes resistores são conectados em paralelo: 50Ω , 75Ω , 300Ω , 60Ω , 500Ω , 180Ω e 200Ω .

12. Crie uma função geradora de um número aleatório inteiro, compreendido entre dois números específicos. Sugestão para o nome da função: `n=randint(a,b)`, onde os argumentos a e b são os parâmetros de entrada (dois números) e n é o número aleatório de saída.

Teste a função na janela Command Window nos seguintes casos:

- a) gerar um número aleatório entre 1 e 49.
b) gerar um número aleatório entre -35 e -2.

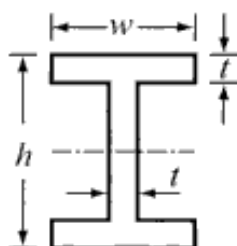
13. O momento de inércia I_{x_0} de um retângulo em relação a um eixo x_0 que passa pelo centróide é: $I_{x_0} = \frac{1}{12}bh^3$. O momento de inércia em relação ao eixo x , paralelo x_0 , é dado por: $I_x = I_{x_0} + A d_x^2$, onde A é a área do retângulo e d_x é a distância normal entre os dois eixos (veja figura).



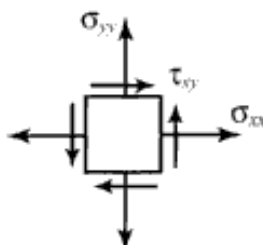
Escreva uma função no MATLAB que determine o momento de inércia de um objeto na forma de um "I" em relação a um eixo que passa pelo centróide (veja figura). Sugestão para o nome da função e dos argumentos: `I = imo(w,h,t)`. Os parâmetros de entrada da função são a largura w , a altura h e a

espessura t , mostrados na figura. (O momento de inércia de um objeto composto pode ser obtido dividindo-se o objeto em várias partes e adicionando-se os momentos de inércia das partes.)

Teste a função para determinar o momento de inércia de um objeto tipo "I" com $w = 200\text{mm}$, $h = 300\text{mm}$ e $t = 22\text{mm}$.



14. A tensão (*stress*), aplicada em um ponto de um objeto bidimensional submetido a uma carga, é definida por três componentes de tensão σ_{xx} , σ_{yy} e τ_{xy} . As tensões normais mínima e máxima (tensões principais) em um ponto ($\sigma_{\min}$ e $\sigma_{\max}$) são calculadas a partir dessas três componentes:



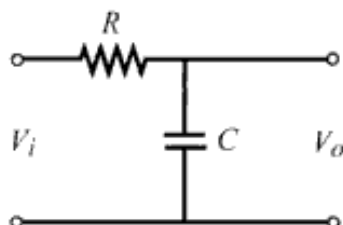
$$\sigma_{\max/\min} = \frac{\sigma_{xx} + \sigma_{yy}}{2} \pm \sqrt{\left(\frac{\sigma_{xx} - \sigma_{yy}}{2}\right)^2 + \tau_{xy}^2}$$

Crie uma função no MATLAB que determine as tensões principais a partir das três componentes de tensão. Sugestão para o nome da função e dos argumentos: `[Smax, Smin] = stressprinc (Sxx, Syy, Sxy)`. Os parâmetros de entrada são as três componentes de tensão e os parâmetros de saída são as tensões principais (máxima e mínima).

Teste a função nos seguintes casos:

- a) $\sigma_{xx} = 150\text{MPa}$, $\sigma_{yy} = -40\text{MPa}$ e $\tau_{xy} = 80\text{MPa}$.
b) $\sigma_{xx} = -12\text{ksi}$, $\sigma_{yy} = -16\text{ksi}$ e $\tau_{xy} = -7\text{ksi}$.

15. Em um filtro passa-baixas RC (filtro que transmite, aproximadamente com a mesma eficiência, sinais de frequências compreendidas entre zero e uma determinada frequência máxima), o módulo da razão entre as tensões de saída e entrada é:



$$RV = \left| \frac{V_o}{V_i} \right| = \frac{1}{\sqrt{1 + (\omega RC)^2}}$$

onde ω é a frequência do sinal de entrada e a razão A_v é denominada ganho de tensão.

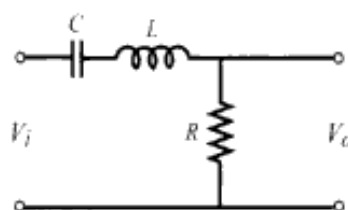
Escreva uma função no MATLAB que determine o módulo de A_v . Sugestão para o nome da função `AV = pbaixas (R, C, w)`. Os argumentos de entrada são: resistência do resistor (R), em ohms (Ω); a capacitância do capacitor (C), em Farads (F), e a frequência angular do sinal de entrada (w), em rad/s. Crie a função de tal forma que w seja um vetor.

Em seguida, escreva uma rotina que utilize a função `pbaixas` para gerar um gráfico do ganho AV em função de ω , $10^{-2} \leq \omega \leq 10^6$ rad/s. Escalone logicamente o eixo horizontal (para ω). Quando a rotina for executada, o usuário deverá ser solicitado a digitar os valores de R e C . Insira título nos eixos do gráfico.

Teste a rotina com $R = 1200\Omega$ e $C = 8\mu F$.

16. Um filtro passa-faixa permite a passagem de frequências compreendidas em uma faixa entre duas frequências limite. Nesse filtro, o ganho de tensão é dado por:

$$RV = \left| \frac{V_o}{V_i} \right| = \frac{\omega RC}{\sqrt{(1 - \omega^2 LC)^2 + (\omega RC)^2}}$$



onde ω é a frequência do sinal de entrada.

Escreva uma função no MATLAB que calcule o ganho A_v . Sugestão para o nome da função `AV = pfaixa(R, C, L, w)`. Os argumentos de entrada são: resistência do resistor (R), em ohm (Ω); a capacitância do capacitor (C), em Farad (F), a indutância do indutor (L), em Henry (H) e a frequência angular do sinal de entrada (w), em rad/s. Crie a função de tal forma que w seja um vetor.

Em seguida, escreva uma rotina que utilize a função `pfaixa` para gerar um gráfico do ganho AV em função de ω , $10^{-2} \leq \omega \leq 10^7$ rad/s. Escalone logicamente o eixo horizontal (para ω). Quando a rotina for executada, o usuário será solicitado a digitar os valores de R , L e C . Insira título nos eixos do gráfico.

Teste a rotina nos seguintes casos:

- a) $R = 1100\Omega$, $C = 9\mu F$ e $L = 7mH$.
b) $R = 500\Omega$, $C = 300\mu F$ e $L = 400mH$.

CAPÍTULO 7

Programando no MATLAB

Programas de computador são uma sequência de comandos devidamente ordenados que visam à resolução de um problema. Em um programa elementar, os comandos são executados um após o outro, na ordem em que são digitados. Todos os programas apresentados nos capítulos sobre rotinas e funções são exemplos de programas elementares. Entretanto, em muitas situações, é frequentemente necessário programas mais sofisticados, onde os comandos não são executados necessariamente na ordem em que aparecem na sequência lógica quando o programa é inicializado com valores diferentes para as variáveis de entrada. Por exemplo, um programa de computador que determine o custo de postagem de pacotes nos Correios e utilize expressões matemáticas diferentes para calcular o custo final, baseadas no peso e no tamanho do pacote, nos itens a serem enviados (o envio de CDs é mais barato do que o de livros, etc.) e o tipo de serviço de transporte disponibilizado (terrestre, marítimo ou aéreo). Em outras situações, pode ser necessário repetir uma sequência de comandos várias vezes dentro de um programa. Por exemplo, um programa que resolve equações numericamente repete uma sequência de cálculos até que o erro na resposta seja menor que alguma medida ou parâmetro especificado.

O MATLAB dispõe de ferramentas que podem ser utilizadas para controlar o fluxo de um programa. As sentenças condicionais (Seção 7.2) e a estrutura `switch...case` (Seção 7.3) tornam possível selecionar comandos ou executar grupos específicos de comandos em diferentes situações. Laços `for` ou `while` (Seção 7.4) proporcionam a repetição sistemática de comandos.

Evidentemente, o redirecionamento do fluxo de um programa requer alguma espécie de estrutura de decisão dentro do programa. O programa deve decidir se executa o próximo comando ou se seleciona um ou mais comandos para continuar em uma linha de programa diferente. O programa toma essas decisões comparando os valores assumidos pelas variáveis. Um modo de fazê-lo é usar operadores lógicos e relacionais, que serão explicados na Seção 7.1.

Para finalizar, deve ser mencionado que funções (Capítulo 6) também podem ser utilizadas na programação. Uma função em si é um subprograma. Quando um programa principal encontra uma linha de comando contendo uma função, essa função é chamada, enquanto o programa principal aguarda o retorno do(s) resultado(s) dessa função. Terminada a execução da função, os cálculos são encerrados, os resultados são retornados para o programa que chamou a função e o programa principal segue para a próxima linha de comando.

7.1 OPERADORES LÓGICOS E RELACIONAIS

O operador relacional compara dois números determinando se o resultado da sentença de comparação é verdadeiro (V) ou falso (F) (p. ex.: $5 < 8 \Rightarrow V$). Se a sentença for verdadeira o valor retornado é 1. Caso contrário, o valor retornado é 0. Um operador lógico examina sentenças verdadeiras/falsas e produz resultados verdadeiro (1) ou falso (0), de acordo com a funcionalidade do operador. Por exemplo, o operador lógico AND resulta 1 (verdadeiro) se, e somente se, todas as sentenças envolvidas na operação forem verdadeiras (1s). Tanto operadores lógicos quanto relacionais podem ser utilizados em expressões matemáticas ou, como será visto adiante, serem combinados a outros comandos para controlar ou tomar decisão sobre o fluxo do programa.

Operadores relacionais:

Os operadores relacionais no MATLAB são:

Operador relacional	Descrição
<	Menor que
>	Maior que
<=	Menor que ou igual a
>=	Maior que ou igual a
= =	Igual a
~ =	Diferente de

Perceba que o operador relacional que testa a igualdade entre dois objetos é representado por dois sinais de igualdade (= =), sem espaço entre eles. Isso porque um único sinal de igualdade representa o operador de atribuição. Os demais operadores duplos (representados por dois caracteres) também não possuem espaços entre os caracteres (<=, >=, ~ =).

- Operadores relacionais são utilizados juntamente com operadores aritméticos dentro de expressões matemáticas. O resultado pode ser utilizado em outras operações matemáticas, no endereçamento de arranjos ou para controlar o fluxo do programa no MATLAB (juntamente com estruturas de tomada de decisão, p. ex.: *if*).
- Quando dois números são comparados, o resultado será 1 (verdadeiro) se, de acordo com o operador relacional, a comparação for verdadeira, e 0 (falso) se a comparação for falsa.

- Se dois escalares estão sendo comparados, o resultado é o escalar 1 ou 0. Caso estejamos comparando arranjos (vetores, matrizes 2D, 3D, etc.), a comparação é feita *elemento por elemento* do arranjo e o resultado é um arranjo lógico de 1s e 0s, cuja dimensão é a mesma dos arranjos originais.
- Se um escalar estiver sendo comparado com um arranjo, o escalar será comparado com cada elemento do arranjo e o resultado é um arranjo lógico de 1s e 0s, de acordo com a posição de cada elemento no arranjo.

Alguns exemplos:

```
>> 5 > 8
ans =
0
>> a=5 < 10
a =
1
>> y=(6 < 10) + (7 > 8) + (5*3 == 60/4)
y =
2
>> b=[15 6 9 4 11 7 14]; c=[8 20 9 2 19 7 10];
>> d=c>b
d =
0 1 1 0 1 1 0
>> b==c
ans =
0 0 1 0 0 1 0
>> b~=c
ans =
1 1 0 1 1 0 1
>> f=b-c>0
```

Verifica se 5 é maior que 8.

Como a comparação é falsa (5 é menor que 8), a resposta é 0.

Verifica se 5 é menor que 10 e atribui a resposta à variável a.

Como a comparação é verdadeira (5 é menor que 10), a resposta 1 é atribuída à variável a.

Igual a 1 porque 6 é menor que 10.

Igual a 0 porque 7 é menor que 8.

Igual a 1 porque $5 \cdot 3$ é igual a $60/4$.

Declara os vetores b e c.

Verifica quais elementos de c são maiores do que ou iguais aos elementos de b.

Atribui 1 onde um elemento de c é maior do que ou igual a um elemento de b.

Verifica quais dos elementos de c são iguais aos elementos de b.

Verifica quais dos elementos de c são diferentes dos elementos de b.

Subtrai c de b e então verifica quais elementos são maiores que zero.

```
f =
    1  0  0  1  0  0  1
>> A = [2 9 4; -3 5 2; 6 7 -1]
```

Declara uma matriz A de dimensão 3 × 3.

```
A =
    2  9  4
   -3  5  2
    6  7 -1
```

```
>> B = A <= 2
```

Verifica quais elementos em A são menores ou iguais a 2. Atribui os resultados à matriz B.

```
B =
    1  0  0
    1  0  1
    0  0  1
```

- Os resultados de uma operação relacional entre vetores formam um vetor de 0s e 1s, denominados vetor lógico, que podem ser utilizados no endereçamento de outros vetores. Quando um vetor lógico é utilizado para tais finalidades, extrai do vetor endereçado os elementos nas posições onde o vetor lógico tem 1s. Por exemplo:

```
>> r = [8 12 9 4 23 19 10]
```

Declara um vetor r.

```
r =
    8  12  9  4  23  19  10
```

```
>> s = r <= 10
```

Verifica quais elementos de r são menores ou iguais a 10.

```
s =
    1  0  1  1  0  0  1
```

O resultado é um vetor lógico s com 1s nas posições onde os elementos de r são menores ou iguais a 10.

```
>> t = r(s)
```

Usa s para endereçar um vetor r e criar o vetor t.

```
t =
    8  9  4  10
```

O vetor t é formado pelos elementos de r nas posições onde s tem 1s.

```
>> w = r(r <= 10)
```

O mesmo pode ser feito em uma única etapa.

```
w =
    8  9  4  10
```

- Os vetores e arranjos numéricos de 0s e 1s não funcionam como vetores e arranjos lógicos de 0s e 1s. Arranjos numéricos não podem ser utilizados para endereçar outros arranjos. Por outro lado, os arranjos lógicos podem ser utilizados em operações aritméticas. Uma vez utilizado em operações aritméticas, o arranjo lógico torna-se numérico.

- Ordem de precedência: em uma expressão matemática que inclua operadores lógicos e aritméticos, as operações aritméticas (+, −, *, /, \) precedem todas as operações relacionais. Os operadores relacionais têm, entre eles, igual precedência e são avaliados da esquerda para a direita. Muitas vezes, parênteses são utilizados para modificar a ordem de precedência. Exemplos:

```
>> 3 + 4 < 16/2
ans =
     1
>> 3 + (4 < 16)/2
ans =
    3.5000
```

As operações + e / são executadas primeiramente.

A resposta é 1 porque $7 < 8$ é verdadeiro.

$4 < 16$ é executado primeiro, o que resulta em 1, porque é verdadeiro.

3.5 é obtido de $3 + 1/2$.

Operadores lógicos:

Os operadores lógicos do MATLAB são:

Operador lógico	Nome	Descrição
& Exemplo: A&B	AND	Age em dois operandos (A, B). Se ambos forem verdadeiros, o resultado será verdadeiro (1). De outro modo, o resultado será falso 0.
 Exemplo: A B	OR	Age em dois operandos (A, B). Se um dos operandos for verdadeiro (em particular os dois), o resultado será verdadeiro (1). De outro modo (ou seja, ambos falsos), o resultado será falso (0).
~ Exemplo: ~A	NOT	Age em um operando (A). Resulta na negação do operando: verdadeiro (1), se o operando for falso, e falso (0), se o operando for verdadeiro.

- Operadores lógicos recebem números como operandos. Qualquer número diferente de zero é verdadeiro e apenas o número zero é falso.
- Operadores lógicos (como os relacionais) podem ser usados juntamente com operadores aritméticos dentro de expressões matemáticas. O resultado pode ser utilizado em outras operações matemáticas, no endereçamento de arranjos ou para controlar o fluxo do programa no MATLAB (juntamente com estruturas de tomada de decisão, p. ex.: if).
- Operadores lógicos (como os relacionais) agem tanto em escalares quanto em arranjos genéricos.
- As operações lógicas AND e OR podem agir em escalares puros, arranjos puros ou numa miscelânea, i.e., entre um escalar e um arranjo. Se ambos os objetos são es-

calares, o resultado será um escalar 0 ou 1. Se ambos são arranjos, eles devem possuir a mesma dimensão e a operação lógica será executada *elemento a elemento*. O resultado será um arranjo lógico de mesma dimensão com 0s e 1s de acordo com a posição que os elementos originais ocupam no *array*. Caso um operando seja um escalar e o outro seja um vetor, a operação lógica é feita entre o escalar e cada um dos elementos no arranjo. O resultado é um arranjo lógico de 0s e 1s, de mesma dimensão do arranjo que entra na operação.

- A operação lógica NOT possui apenas um operando. Quando utilizada em um escalar, resulta no escalar 0 ou 1. Se utilizada em arranjos, cada elemento do arranjo será negado, i.e., os elementos falsos serão tornados verdadeiros e os elementos verdadeiros (diferentes de zero) serão tornados falsos.

Eis alguns exemplos:

```
>> 3&7
```

3 AND 7.

```
ans =
```

```
1
```

Tanto 3 quanto 7 são verdadeiros (diferentes de zero), então o resultado é verdadeiro (1).

```
>> a = 5|0
```

5 OR 0 (atribuído à variável a).

```
a =
```

```
1
```

O valor 1 é atribuído a a, visto que um dos operandos é verdadeiro (diferente de zero).

```
>> ~25
```

NOT 25.

```
ans =
```

```
0
```

O resultado é 0, visto que 25 é verdadeiro (diferente de zero). Negando, resulta em falso.

```
>> t = 25*((12&0) + (~0) + (0|5))
```

Usando operadores lógicos em expressões matemáticas.

```
t =
```

```
50
```

```
>> x = [9 3 0 11 0 15]; y = [2 0 13 -11 0 4];
```

Declarando os vetores x e y.

```
>> x&y
```

```
ans =
```

```
1 0 0 1 0 1
```

O resultado é um vetor com 1s nas posições, onde x e y são verdadeiros simultaneamente (diferentes de zero), e 0s de outro modo.

```
>> z = x|y
```

```
z =
```

```
1 1 1 0 1
```

O resultado é um vetor com 1s, nas posições, onde x ou y são verdadeiros (diferentes de zero), e 0s de outro modo.

```
>> ~(x + y)
```

```
ans =
```

```
0 0 0 1 1 0
```

O resultado é um vetor com 1s, nas posições onde x ou y são verdadeiros (diferentes de zero), e 0s de outro modo.

Ordem de precedência:

Operadores aritméticos, lógicos e relacionais podem ser utilizados simultaneamente em expressões matemáticas. O resultado da expressão depende do modo como eles estão organizados. O MATLAB utiliza a ordem de precedência que segue:

Precedência	Operação
1 (Mais alta)	Parênteses (se existirem parênteses aninhados, os internos têm precedência mais alta)
2	Exponenciação
3	Lógica NOT (~)
4	Multiplicação e divisão
5	Adição e subtração
6	Operadores relacionais (>, <, >=, <=, =, ~=)
7	Lógica AND (&)
8 (Mais baixa)	Lógica OR ()

Se duas ou mais operações possuem a mesma ordem de precedência, a operação mais à esquerda é executada primeiro, depois as demais vão sendo resolvidas uma a uma, até encontrar a operação mais à direita.

Deve ser mencionado aqui que essa ordem foi adotada no MATLAB a partir da versão 6. Versões anteriores do MATLAB têm uma ordem de precedência ligeiramente diferente. Nelas, o operador & não precede |. O usuário deve estar atento a isso. Os problemas de compatibilidade entre versões do MATLAB podem ser resolvidos facilmente com o uso sistemático de parênteses (até mesmo quando eles não são requeridos).

Os exemplos de expressões a seguir ilustram o uso de operadores aritméticos, lógicos e relacionais:

```
>> x = -2; y = 5;
```

Declarando os escalares x e y.

```
>> -5 < x < -1
```

```
ans =
```

```
0
```

Esta desigualdade está matematicamente correta. Porém, o resultado é falso porque o MATLAB executa-a da esquerda para a direita. $-5 < x$ é verdadeiro, logo $1 < -1$ é falso ($=0$).

```
>> -5 < x & x < -1
```

```
ans =
```

```
1
```

O resultado matematicamente correto é obtido usando-se o operador lógico &. As desigualdades são executadas em primeiro lugar. Sendo ambas verdadeiras (1), a resposta é 1.

```
>> ~(y < 7)
```

```
ans =
```

```
0
```

$y < 7$ é executado primeiro (resultando em verdadeiro). Logo -1 é falso ($=0$).

```
>> ~y < 7
```

```
ans =
```

```
1
```

$\sim y$ é executado primeiro, resultando em verdadeiro (1) (visto que y é diferente de zero). Logo $-1 < 7$ é verdadeiro ($=1$).

Solução

Na rotina abaixo, as temperaturas foram colocadas em um vetor. Em seguida, operadores lógicos e relacionais foram usados para analisar os dados.

```
T = [58 73 73 53 50 48 56 73 73 66 69 63 74 82 84 ...  
     91 93 89 91 80 59 69 56 64 63 66 64 74 63 69];
```

```
Tmaior_igual75 = T >= 75;
```

Vetor contendo 1s nas posições onde $T \geq 75$.

```
Ndias_Tmaior_igual75 = sum(Tmaior_igual75)
```

Adiciona todos os 1s no vetor `Tmaior_igual75`.

```
Tentre65_80 = (T >= 65) & (T <= 80);
```

Vetor contendo 1s nas posições onde $T \geq 65$ e $T \leq 80$.

```
Ndias_Tentre65_80 = sum(Tentre65_80)
```

Adiciona todos os 1s no vetor `Tentre65_80`.

```
datasTentre50_60 = find((T >= 50) & (T <= 60))
```

A função `find` retorna os endereços dos elementos em `T` cujos valores estão entre 50 e 60.

A rotina (salva como `Capitulo7_Exemplo1`) executada na linha do *prompt* resulta em:

```
>> Capitulo7_Exemplo1
```

```
Ndias_Tmaior_igual75 =
```

A temperatura esteve acima de 75°F em 7 dias do mês.

```
7
```

```
Ndias_Tentre65_80 =
```

A temperatura esteve entre 65° e 80°F em 12 dias

```
12
```

```
datasTentre50_60 =
```

Os dias em que a temperatura esteve entre 50° e 60°F.

```
1 4 5 7 21 23
```

7.2 SENTENÇAS CONDICIONAIS

As sentenças condicionais são estruturas que permitem ao MATLAB tomar decisões e optar por executar um grupo de comandos que seguem dessa sentença (caso seja verdadeira) ou selecionar um desvio (salto no grupo de comandos), seguindo para a próxima linha de instrução. Em uma sentença condicional é imperativo uma expressão de teste. Se a expressão de teste for verdadeira, o MATLAB executa um grupo de comandos que segue a sentença. Se a expressão for falsa, o programa salta para outro comando ou grupo de comandos. A sintaxe básica de uma sentença condicional é:

```
if expressão condicional consistindo de operadores lógicos e/ou relacionais
```

Exemplos:

```
if a < b
if c >= 5
if a == b
if a ~= 0
if (d<h) & (x>7)
if (x~=13) | (y<0)
```

Todas as variáveis devem ter sido inicializadas.

- As sentenças condicionais podem fazer parte de um programa, rotina ou função.
- Conforme mostrado a seguir, cada sentença `if` deve ser seguida de um comando `end`.

A sentença `if` é usada geralmente em três estruturas: `if-end`, `if-else-end` e `if-elseif-else-end`, que são os objetos de estudo das próximas seções.

7.2.1 A estrutura `if-end`

A estrutura condicional `if-end` tem a estrutura esquemática apresentada na Figura 7-1. A figura mostra como os comandos devem ser digitados no programa, bem como o fluxograma que, simbolicamente, dita o fluxo ou a seqüência segundo o qual os comandos serão executados. Quando um programa é executado, encontra uma sentença `if`. Se a expressão condicional de teste for verdadeira (1), o programa continuará a executar os comandos listados abaixo da sentença `if`, até encontrar o comando `end`. Se a expressão de teste for falsa (0), o programa saltará o grupo de comandos entre as sentenças `if` e `end`, continuando a execução dos comandos listados abaixo de `end`. Portanto, o programa executará a sentença somente quando o resultado do teste for verdadeiro.

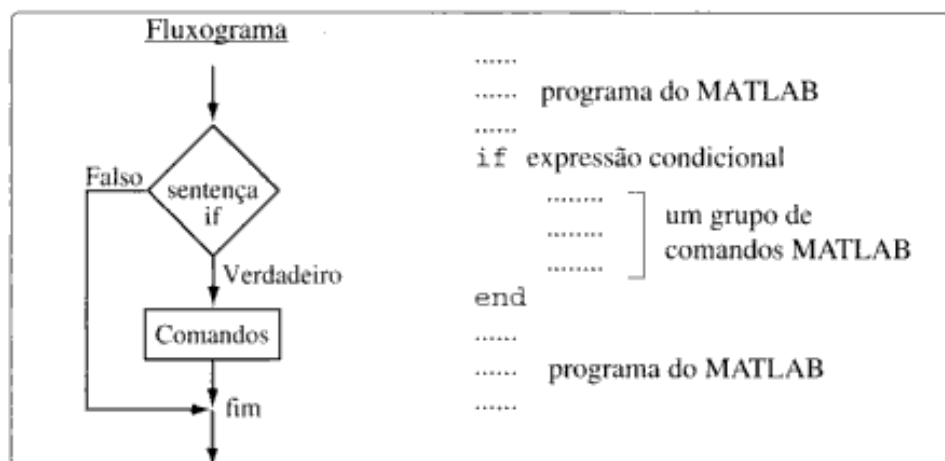


FIGURA 7-1 A estrutura da sentença condicional `if-end`.

7.2.2 A estrutura if-else-end

Esta estrutura executa um grupo de comandos, se o resultado do teste for verdadeiro, ou outro grupo, se o resultado for falso. Observe a Figura 7-2.

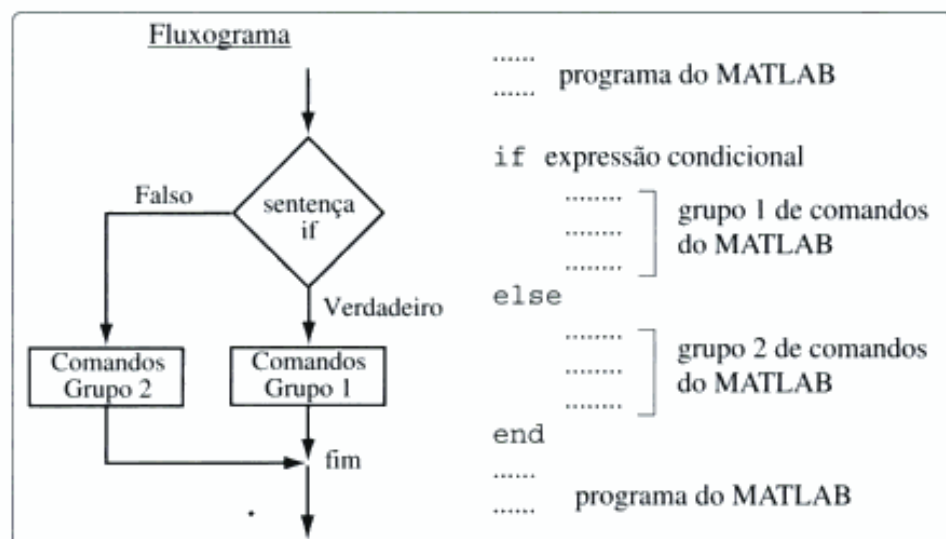


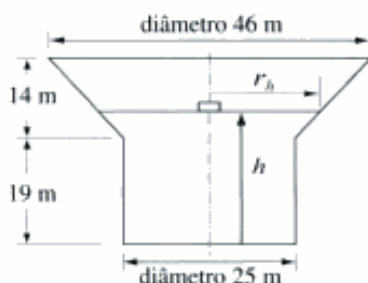
FIGURA 7-2 A estrutura da sentença condicional if-else-end.

A figura mostra como os comandos devem ser digitados e o fluxograma ilustra o fluxo ou a sequência do programa segundo o qual os comandos serão executados. A primeira linha contém uma sentença if com a respectiva expressão de teste. Se o resultado da expressão de teste for verdadeiro, o programa executará os comandos do grupo 1 (entre as sentenças if e else) e saltará para o comando end. Se o resultado do teste for falso, o programa saltará para a sentença else e executará os comandos do grupo 2 (entre as sentenças else e end).

O exemplo a seguir ilustra a estrutura condicional if-else-end.

Problema-exemplo 7-3: Nível de uma caixa d'água

O tanque de uma caixa d'água possui a geometria mostrada na figura (a base é um cilindro e a parte superior é um cone invertido cortado). Dentro do tanque há uma bóia que indica o nível da água. Escreva uma função que determine o volume da água armazenada no tanque, na posição indicada pela bóia (altura h). A função deve receber a variável h (em m) e retornar o volume da água (em m^3).



Solução

Para $0 \leq h \leq 19\text{m}$, o volume do tanque é dado pelo volume de um cilindro de altura $h = 19\text{m}$: $V = (12,5)^2 \pi h$.

Para $19 \leq h \leq 33\text{m}$, o volume do tanque é a adição dos volumes do cilindro com $h = 19\text{m}$ e o volume da água na porção do cone invertido:

$$V = \pi 12.5^2 \cdot 19 + \frac{1}{3} \pi (h - 19)(12.5^2 + 12.5 \cdot r_h + r_h^2)$$

onde $r_h = 12.5 + \frac{10.5}{14}(h - 19)$.

A função, escrita na janela Editor, foi salva como `v = voldagua (h)`.

```
function v = voldagua(h)
% Voldagua determina o volume de agua numa caixa dagua.
% A entrada e o nivel de agua (em m).
% A saida e o volume de agua numa caixa dagua (em m3).
if h <= 19
    v = pi*12.5^2*h;
else
    rh = 12.5 + 10.5*(h-19)/14;
    v = pi*12.5^2*19 + pi*(h-19)*(12.5^2 + 12.5*rh + rh^2)/3;
end
```

Chamando a função na linha do *prompt* (salva como voldagua) para resolver dois casos:

```
>> voldagua(8)
ans =
    3.9270e+003
>> voldagua(25.7)
ans =
    1.4115e+004
```

7.2.3 A estrutura if-elseif-else-end

A estrutura de `if-elseif-else-end` está indicada na Figura 7-3. A figura mostra como os comandos devem ser digitados no programa e ilustra o fluxo e/ou a sequência por meio de um fluxograma segundo o qual os comandos serão executados. Essa estrutura possui duas sentenças de teste (`if` e `elseif`) que tornam possível selecionar um dos três grupos de comandos para a execução. A primeira linha traz a sentença `if` com a expressão de teste. Se o resultado do teste é verdadeiro, o programa executa os comandos do grupo 1 (entre as sentenças `if` e `elseif`) e salta para `end`. Se o resultado do teste da sentença `if` é falso, o programa salta para a sentença `elseif`. Caso a expressão de teste de `elseif` seja verdadeira, o programa executa os comandos do grupo 2 (entre `elseif` e `else`) e salta para `end`. Se a expressão de teste de `elseif` é falsa, o programa salta para `else` e executa os comandos do grupo 3 (entre `else` e `end`).

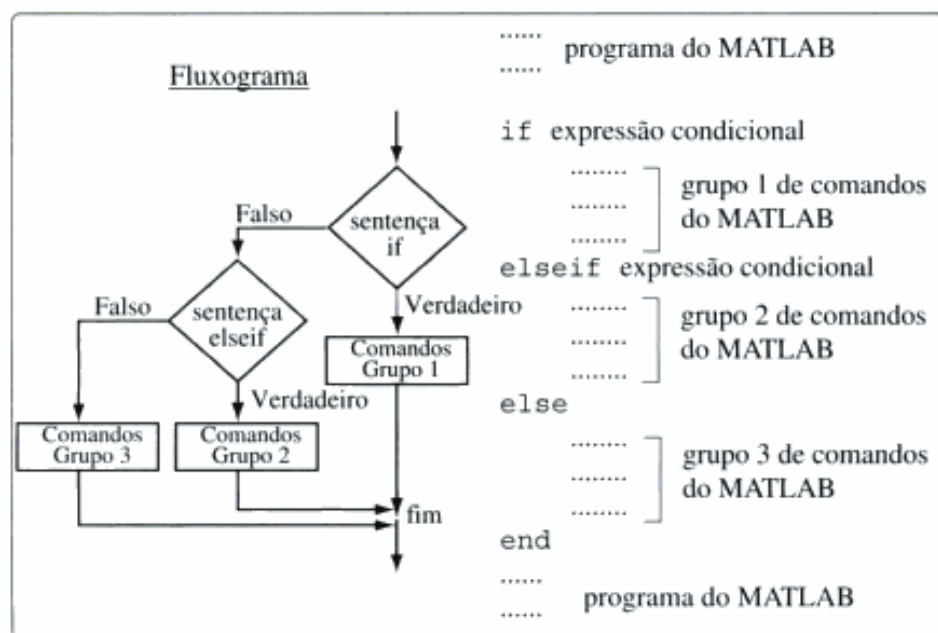


FIGURA 7-3 A estrutura da sentença condicional if-elseif-else.

Deve ser enfatizado que é possível aninhar muitas sentenças `elseif` na estrutura e vários grupos de comandos podem ser adicionados dentro desse ninho. Isso possibilita testar muitas condições num único programa. Além disso, o comando `else` é opcional. Isso significa que, caso sejam aninhadas várias sentenças `elseif` qualquer sentença condicional verdadeira terá os respectivos comandos executados. Caso contrário, não haverá execução de comandos, a menos que haja um `else` na estrutura.

7.3 A SENTENÇA `switch-case`

A sentença `switch-case` é um modo mais elegante de selecionar uma dentre várias opções de um programa. A estrutura da sentença é mostrada na Figura 7-4.

- A primeira linha deve conter o comando `switch`, escrito da seguinte forma:

```
switch valor (opcao) do switch
```

O valor ou opção do `switch` pode ser um número ou uma *string*. É, tipicamente, uma variável inicializada com um número ou uma *string*. Entretanto, é possível introduzir uma expressão matemática, cujas variáveis encontram-se declaradas na opção do `switch`.

- Seguindo na estrutura do comando `switch`, vem um ou muitos comandos `case`. Cada `case` possui um valor característico (numérico ou *string*) no lado direito do comando (`valor1`, `valor2`, etc.) e um grupo de comandos associados abaixo dele.

```

.....   programa do MATLAB
.....

switch expressão de switch
    case valor 1
        .....   ] grupo 1 de comandos
        .....   ]
    case valor 2
        .....   ] grupo 2 de comandos
        .....   ]
    case valor 3
        .....   ] grupo 3 de comandos
        .....   ]
    otherwise
        .....   ] grupo 4 de comandos
        .....   ]
end
.....   programa do MATLAB
.....

```

FIGURA 7-4 A estrutura da sentença switch-case.

- Após o último comando `case`, vem o comando opcional `otherwise`, seguido de um grupo de comandos.
- A última linha sempre deve conter uma sentença `end`.

Como funciona a sentença switch-case?

O valor da expressão `switch` no comando `switch` é comparado com os valores relacionados a cada `case`. Caso seja encontrada uma coincidência, o grupo de comandos relacionados ao `case` é executado. (Somente um grupo de comandos; aquele situado entre dois `cases`, ou entre um `case` e um `otherwise` ou entre `otherwise` e `end`.)

- Se houver opção repetida, apenas a primeira será executada.
- Se não for encontrado o valor do `switch` e a sentença `otherwise` estiver presente, o grupo de comandos entre `otherwise` e `end` será executado.
- Se não for encontrado o valor do `switch` e a sentença `otherwise` não estiver presente, nenhum grupo de comandos será executado.
- Uma sentença `case` pode possuir mais de um valor. Isso pode ser feito digitando-se os valores entre chaves, na forma: `{valor1, valor2, valor3,...}`. Essa forma, não descrita neste livro, é denominada célula de arranjo. O `case` será executado se pelo menos um dos valores entre chaves for encontrado.

NOTA: no MATLAB somente o primeiro case encontrado é executado. Após o grupo de comandos associados ao case, o programa salta para a sentença end. Isso distingue o MATLAB de outras linguagens de programação (por exemplo, o C), nas quais um comando break é necessário na parada de cada case.

Problema-exemplo 7-4: Convertendo unidades de energia

Escreva uma rotina que converta uma quantidade de energia ou trabalho escrita em Joule, ft-lb, cal ou eV nas quantidades de energia equivalentes nas demais unidades especificadas pelo usuário. O programa deve solicitar ao usuário que entre com a quantidade de energia, a unidade atual e a nova unidade requerida. A saída é a quantidade de energia escrita na nova unidade.

Fatores de conversão: $1\text{J} = 0.738\text{ ft-lb} = 0.239\text{ cal} = 6.24 \times 10^{18}\text{ eV}$.

Teste o programa para converter:

- 150J em ft-lb.
- 2.800 cal em Joules.
- 2.7eV em cal.

Solução

O programa solução inclui dois conjuntos de sentenças `switch-case` e um conjunto `if-else-end`. O primeiro conjunto `switch-case` é utilizado para converter a quantidade de energia inicial em Joules. O segundo é utilizado para converter a quantidade em Joules para as novas unidades. A estrutura `if-else-end` é utilizada para gerar uma mensagem de erro, caso as unidades sejam digitadas incorretamente.

```
Eent = input('Entre com o valor da energia(trabalho) a ser convertido: ');
Eent_unid = input('Entre com a unidade atual (J, ft-lb, cal ou eV): ','s');
Esai_unid = input('Selecione a nova unidade de energia (J,ft-lb,cal ou eV): ','s');
erro = 0;
switch Eent_unid
case 'J'
    EJ = Eent;
case 'ft-lb'
    EJ = Eent/0.738;
case 'cal'
    EJ = Eent/0.239;
case 'eV'
    EJ = Eent/6.24e18;
otherwise
    erro = 1;
end
switch Esai_unid
```

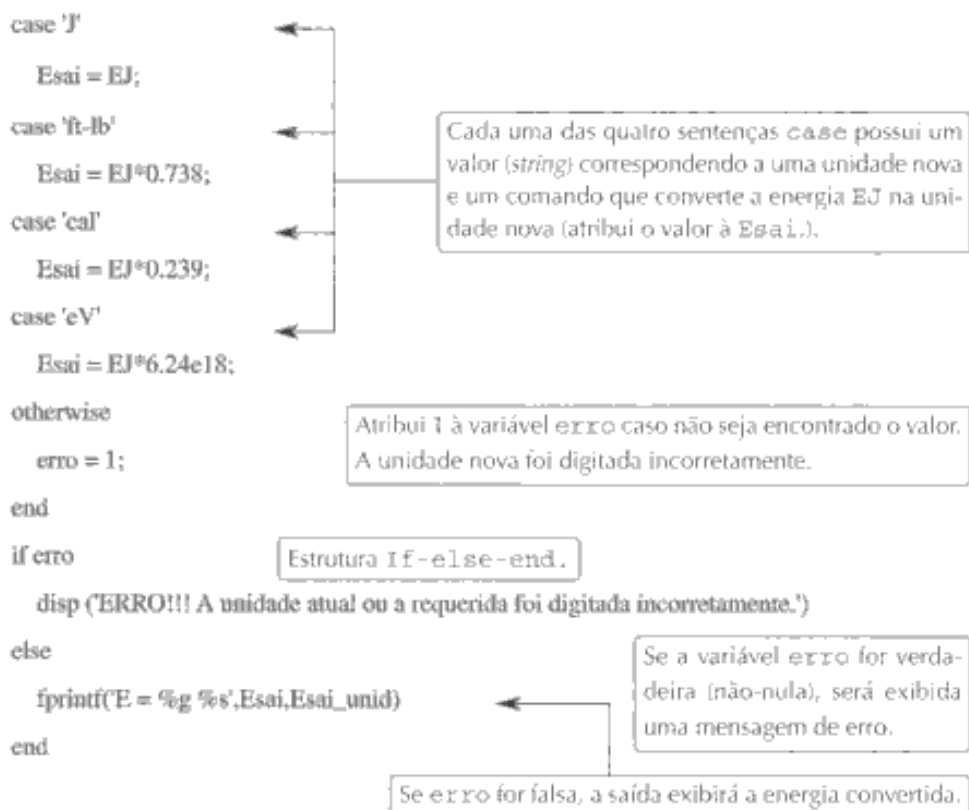
Atribui 0 à variável erro.

Primeira sentença switch. O valor do switch é uma string com as unidades iniciais.

Cada uma das quatro sentenças case possui um valor (string) correspondendo a uma unidade inicial e um comando que converte a energia Eent, escrita numa unidade qualquer, para Joule (atribui o valor à EJ).

Atribui 1 à variável erro, caso não seja encontrado o valor do case. A unidade inicial foi digitada incorretamente.

Segunda sentença switch. O valor do switch é uma string com as unidades novas.



A título de exemplo, a rotina (salva como `Capitulo7_Exemplo4`) é testada na janela Command Window para resolver a letra *b* do problema.

```

>> Capitulo7_Exemplo4
Entre com o valor da energia(trabalho) a ser convertido: 2800
Entre com a unidade atual (J, ft-lb,cal ou eV): cal
Selecione a nova unidade de energia (J,ft-lb,cal ou eV): J
E = 11715.5 J

```

7.4 LAÇOS (LOOPS)

O laço é outro método de alterar o fluxo de um programa. Em um laço, a execução de um ou um grupo de comandos é repetida diversas vezes, sucessivamente. Cada seqüência de execução (repetição) do laço é denominada passo. A cada passo, ao menos uma variável é modificada dentro do laço. O MATLAB traz dois tipos de laço: *for-end* e *while-end*. No laço *for-end* (Seção 7.4.1), o número de repetições é conhecido desde o início do laço. No laço *while-end* (Seção 7.4.2), o número de repetições não é conhecido de antemão e o laço é repetido até que uma condição específica de parada seja satisfeita. Ambos os tipos de laço podem ser finalizados a qualquer momento, por meio do comando *break* (veja a Seção 7.6).

- O valor da variável índice do laço (k) não é mostrado automaticamente. É possível exibir o valor de k a cada passo (às vezes, útil no trabalho de *debugging* – depuração – do programa), digitando k como um dos comandos do laço.
- Terminado o laço, a variável índice (k) sai com o último valor atribuído a ela.

Um exemplo simples de um laço `for-end` (escrito como uma rotina) é:

```
for k=1:3:10
    x=k^2
end
```

Executando a rotina, o laço é repetido quatro vezes. O valor de k nas quatro repetições é $k = 1, 4, 7$ e 10 , e os respectivos valores atribuídos à variável x são: $x = 1, 16, 49$ e 100 . Como não foi digitado um ponto-e-vírgula após a segunda linha, os valores de x são exibidos, um a um, na janela Command Window. Executando a rotina, os resultados de x na janela Command Window são:

```
>>x =
     1
x =
    16
x =
    49
x =
   100
```

Problema-exemplo 7-5: Soma de séries

- Construa um laço `for-end` em uma rotina para determinar a soma dos n primeiros termos da série: $\sum_{k=1}^n \frac{(-1)^k k}{2^k}$. Execute a rotina para $n = 4$ e $n = 20$.
- A função $\sin(x)$ pode ser escrita em termos da série de Taylor como:

$$\sin x = \sum_{k=0}^{\infty} \frac{(-1)^k x^{2k+1}}{(2k+1)!}$$

Escreva uma função que calcule $\sin(x)$ usando a série de Taylor. Sugestão para o nome da função e argumentos: $y = \text{Tsin}(x, n)$. Os argumentos de entrada são o ângulo x , em graus, e o número de termos da série n . Teste a função para calcular $\sin(150^\circ)$, usando 3 e 7 termos.

Solução

- Uma rotina elaborada para calcular a soma dos n primeiros termos da série é mostrada a seguir:

```

n = input('Entre com o numero de termos da serie:');
S = 0;
for k = 1:n
    S = S+(-1)^k*k/2^k;
end
fprintf('A soma da serie e: %f',S)

```

Estabelecendo o resultado inicial da soma em zero.

Laço for-end

A cada repetição é determinado um elemento da série e o resultado é adicionado à soma do laço anterior.

O somatório é realizado dentro do laço. A cada passo um termo da série é calculado, sendo automaticamente adicionado à soma dos termos precedentes na série. Resta-nos executar o arquivo (salvo como `Capitulo7_Exemplo5`) duas vezes na janela Command Window:

```

>> Capitulo7_Exemplo5
Entre com o numero de termos da serie:4
A soma da serie e: -0.125000
>> Capitulo7_Exemplo5
Entre com o numero de termos da serie:20
A soma da serie e: -0.222216

```

b) Uma declaração de função que encontra $\sin(x)$ adicionando n termos da fórmula de Taylor é mostrada a seguir:

```

function y =Tsin(x,n)
% Tsin(x) determina o sin(x) com base na formula de Taylor.
% Argumentos de entrada:
% x (angulo em graus) e n (numero de termos da serie).
xr = x*pi/180;
y = 0;
for k = 0:n-1
    y = y+(-1)^k*xr^(2*k+1)/factorial(2*k+1);
end

```

Convertendo o ângulo de graus para radianos.

Laço for-end

O primeiro elemento da série corresponde a $k = 0$ e o último passo acontece para $k = n - 1$, de modo a adicionar n termos da série. Utilizando a função duas vezes na janela Command Window para calcular o $\sin(150^\circ)$, com 3 e 7 termos na série, respectivamente, temos:

Executando a rotina na janela Command Window:

```
>> Capitulo7_Exemplo6
V =
    10    17   -27     8     0    -7    24    30    40    -6    12     4    -8    16
```

7.4.2 Laços while-end

O comando `while-end` é útil quando se deseja realizar um laço mas o número de repetições não é conhecido de antemão. Sendo assim, nesse laço, o formato do comando não contém um campo especificando o número de repetições (antes de iniciá-lo). Em vez disso, o processo continua sendo executado até que uma certa condição seja satisfeita. A estrutura do comando `while-end` é mostrada na Figura 7-6.

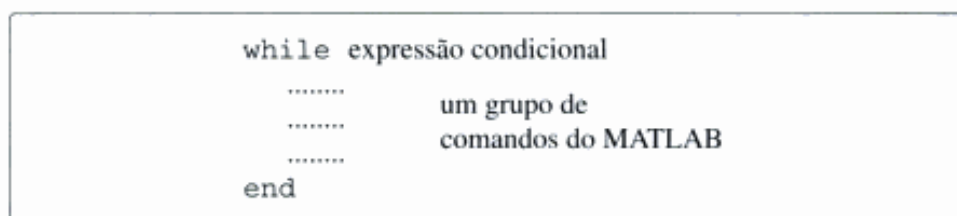


FIGURA 7-6 A estrutura do laço `while-end`.

A primeira linha traz a sentença `while`, incluindo a sentença condicional. Ao encontrar essa linha, o programa verifica a expressão condicional. Sendo falsa (0), o MATLAB salta para a sentença `end` e continua na primeira linha abaixo. Sendo verdadeira (1), o MATLAB executa o grupo de comandos compreendidos entre `while` e `end`. Depois disso, o MATLAB salta para a sentença `while` e verifica a expressão condicional. O laço é realizado enquanto a expressão condicional for verdadeira, abandonando o laço apenas quando o resultado do teste for falso.

Para um laço `while-end` ser executado corretamente:

- A expressão condicional no comando `while` deve incluir ao menos uma variável.
- As variáveis na expressão condicional devem ter sido inicializadas com os valores corretos quando o MATLAB encontrar, na primeira vez, o comando `while`.
- Ao menos uma variável na expressão condicional deve ser modificada dentro do laço, ou seja, entre os comandos `while` e `end`. De outro modo, o programa entrará em um laço infinito.

Um exemplo simples de laço `while-end` é mostrado no programa a seguir. Nesse programa, a variável `x`, inicializada com o valor 1, é dobrada a cada passo, enquanto o valor for menor ou igual a 15.


```
x=1
while x<=15
    x=2*x
end
```

Declarando $x = 1$.O próximo comando é executado somente se $x \leq 15$.

O resultado na janela Command Window é:

```
x =
    1
x =
    2
x =
    4
x =
    8
x =
   16
```

Valor inicial de x .O valor de x é dobrado a cada repetição.Quando $x = 16$, a expressão condicional no comando `while` torna-se falso e o laço termina.

Observação:

Ao escrever um programa que utilize um laço `while-end`, o programador deve certificar-se de que a variável ou variáveis presente(s) na expressão condicional, e que deve ser modificada dentro do laço, receberá um valor que tornará falsa a expressão condicional do comando `while`. De outro modo, conforme dito anteriormente, o laço continuará indefinidamente (laço infinito). No exemplo acima, se a expressão condicional for modificada para $x \geq 0.5$, o laço nunca terminará. Essas situações podem ser evitadas contando-se o número de repetições antes de parar o laço. Isso pode ser feito adicionando-se o número máximo admissível de repetições ou utilizando um comando `break` (Seção 7.6).

Mesmo os melhores programadores cometem erros e um *loop* infinito pode ocorrer por descuido do programador. Caso isso aconteça, o usuário pode parar a execução do programa pressionando, simultaneamente, as teclas **Ctrl+C** ou **Ctrl+Break**.

Problema-exemplo 7-7: Representação em série de potência de uma função (série de Taylor)

A função $f(x) = e^x$ pode ser representada em série de Taylor por: $e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$.

Escreva um rotina que determine e^x usando a representação em série de Taylor. O programa deve calcular e^x adicionando-se os termos da série e parar quando o valor absoluto da última parcela adicionada tornar-se menor que 0.0001. Use um laço `while-end`, mas limite o número de repetições a 30. Se na trigésima repetição o valor absoluto do termo

número de repetições depende do valor da variável x . Testando a rotina (salva como Capítulo7_Exemplo7) na linha do *prompt* da janela Command Window para e^2 , e^{-4} e e^{21} :

```
>> Capítulo7_Exemplo7
Entre com o valor de x: 2
exp(2.000000) = 7.389046
O numero de termos utilizado foi: 12
O n-esimo termo e: 0.000051
>> Capítulo7_Exemplo7
Entre com o valor de x: -4
exp(-4.000000) = 0.018307
O numero de termos utilizado foi: 18
O n-esimo termo e: -0.000048
>> Capítulo7_Exemplo7
Entre com o valor de x: 21
Sao necessarios mais de 30 termos para a representacao em serie de Taylor com a precisao re-
querida.
else
    fprintf('exp(%f)=%f,x,S)
    fprintf('\nO numero de termos utilizado foi: %i',n)
    fprintf('\nO n-esimo termo e:%f,an)
end
```

Calculando exp(2).

Foram utilizados 12 termos.

O valor do n-ésimo termo.

Calculando exp(-4).

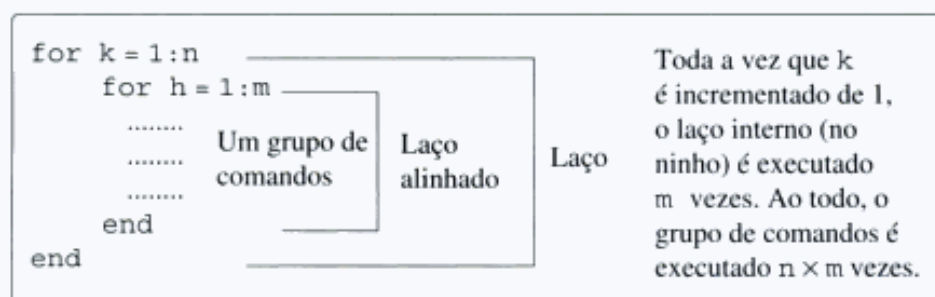
Foram utilizados 18 termos.

Tentando calcular exp(21).

7.5 LAÇOS ANINHADOS E NINHOS DE SENTENÇAS CONDICIONAIS

Laços e sentenças condicionais podem ser aninhadas entre si. Isso significa que um *loop* ou uma sentença condicional pode ser inicializada (e terminada) dentro de outro *loop* ou sentença condicional. Não há limites para o número de laços e sentenças condicionais que podem ser aninhadas. Entretanto, deve ser salientado que toda estrutura *if*, *case*, *for* e *while* deve terminar em um comando *end* correspondente. A Figura 7-7 ilustra a estrutura de um ninho de *for-end* dentro de outro laço *for-end*.

Nos laços mostrados na figura, se, por exemplo, $n = 3$ e $m = 4$, então o primeiro $k = 1$ e o laço aninhado são executados quatro vezes, para $h = 1, 2, 3$ e 4 . Em seguida, $k = 2$ e o laço aninhado são executados novamente outras quatro vezes. Por último, $k = 3$ e outra sequência de quatro laços internos são executados. Toda vez que laços aninhados são digitados, o MATLAB automaticamente promove a indentação do laço mais interno, relativamente ao(s) laço(s) externo(s). O Problema-exemplo 7-8 demonstra o uso de ninhos de laços e sentenças condicionais.

FIGURA 7-7 Estrutura de ninhos de *loops*.**Problema-exemplo 7-8: Criando uma matriz com um loop**

Escreva uma rotina que crie uma matriz $m \times n$ cujos elementos tenham os seguintes valores: os valores dos elementos da primeira linha seguem a numeração das colunas. Os valores dos elementos da primeira coluna seguem a numeração das linhas. Os demais elementos são iguais à soma dos elementos imediatamente acima e à esquerda do elemento em questão. Quando executado, o programa deve solicitar ao usuário entrar com os valores de m e n .

Solução

O programa a seguir possui dois laços (um *ninho*) e uma sentença condicional aninhada *if-elseif-else-end*. Os elementos na matriz são inicializados um a um, linha por linha. As variáveis índice k e h endereçam as linhas e as colunas, respectivamente, do primeiro e segundo *loop*.

```

m = input('Entre com o numero de linhas da matriz: ');
n = input('Entre com o numero de colunas da matriz: ');
A = [];
for k = 1:n
    for h = 1:m
        if k == 1
            A(k,h) = h;
        elseif h == 1
            A(k,h) = k;
        else
            A(k,h) = A(k,h-1) + A(k-1,h);
        end
    end
end
A

```

Annotations for the code:

- `A = [];`: Declara a matriz A e não a inicializa.
- `for k = 1:n`: Início do primeiro *loop for-end*.
- `for h = 1:m`: Início do segundo *loop for-end*.
- `if k == 1`: Início da sentença condicional.
- `A(k,h) = h;`: Atribui valores aos elementos da primeira linha.
- `elseif h == 1`: Atribui valores aos elementos da segunda linha.
- `else`: Atribui valores aos outros elementos.
- `A(k,h) = A(k,h-1) + A(k-1,h);`: Atribui valores aos outros elementos.
- `end` (after if): end da sentença *if*.
- `end` (after for h): end do *loop for-end* interno.
- `end` (after for k): end do *loop for-end* externo.

Executando a rotina (salva como `Capitulo7_Exemplo8`) na linha do *prompt* da janela Command Window é gerada uma matriz 4×5 .

```
>>> Capitulo7_Exemplo8
Entre com o numero de linhas: 4
Entre com o numero de colunas: 5
A =
    1    2    3    4    5
    2    4    7   11   16
    3    7   14   25   41
    4   11   25   50   91
```

7.6 OS COMANDOS `break` E `continue`

O comando `break`:

- Quando inserido em um laço, `for` ou `while`, provoca a saída imediata do laço. Assim, quando o MATLAB encontra um comando `break` dentro do laço, o programa salta para o comando `end` desse laço e segue para o próximo comando imediatamente abaixo, saindo incondicionalmente do laço.
- Inserido dentro de um *loop* aninhado, finaliza somente o *loop* que contém o `break`.
- Aparecendo na estrutura de uma rotina ou função, provoca o término imediato da execução das mesmas, a partir do ponto onde está situado.
- Geralmente, aparece dentro de sentenças condicionais. Caso uma certa condição de teste seja satisfeita, é um modo rápido e seguro de provocar a parada na execução do processo de *looping*. Por exemplo, quando o número de laços exceder um valor predeterminado ou acontecer um erro em algum procedimento numérico. Nesses casos, o comando `break` provoca uma saída imediata, para que dados incorretos (inconsistentes com o esperado) sejam propagados pela função.

O comando `continue`:

- É utilizado dentro de um laço, `for` ou `while`, de maneira a parar o passo atual e iniciar o próximo passo no processo de *looping*.
- Geralmente, o comando `continue` entra como parte de uma sentença condicional. Quando o MATLAB encontra o comando `continue`, deixa de executar o restante do laço atual, saltando para o comando `end` no final do laço, para então iniciar um novo passo.

7.7 EXEMPLOS DE APLICAÇÕES DO MATLAB

Problema-exemplo 7-9: Saques em uma conta de poupança

Um aposentado depositou \$300,000 em uma caderneta de poupança que paga uma taxa de juros anual de 5%. Ele planeja sacar o dinheiro da conta uma vez a cada ano, come-

quando com um saque de \$25,000 no primeiro ano e incrementando a quantia anual de acordo com a inflação. Por exemplo, se a inflação anual for 3%, ele sacará \$25,750 após o segundo ano. Sabendo disso, determine a quantidade de saques que ele poderá fazer na conta (em número de anos), considerando uma inflação anual constante de 2%. Construa um gráfico que mostre os saques anuais em função do saldo da conta ao longo dos anos.

Solução

O problema será resolvido usando um *loop*; no caso, *while*, porque o número de repetições não é conhecido de antemão. A cada passo, a quantia a ser sacada e o novo saldo da conta devem ser calculados. O processo de *looping* continua até que o saldo da conta seja menor ou igual à quantidade a ser sacada. O programa a seguir resolve o problema usando uma rotina. Nesse programa, as variáveis *ano*, *saque* e *saldo* são vetores que representam o número de anos, a quantia sacada anualmente e o saldo anual da conta, respectivamente.

A rotina (salva como *Capitulo7_Exemplo9*) executada na janela *Command Window*:

```

juros = 0.05; inf = 0.02;

clear saque saldo ano

ano(1) = 0;
saque(1) = 0;
saldo(1) = 0;

prosaque = 25000;
saldocor = 300000*(1+juros);
n = 2;

while saldocor >= prosaque
    ano(n) = n-1;
    saque(n) = prosaque;
    saldo(n) = saldocor-saque(n);
    saldocor = saldo(n)*(1+juros);
    prosaque = saque(n)*(1+inf);
    n = n+1;
end

fprintf('Os saques serao efetuados durante %.f anos.',ano(n-1))
bar(ano,[saldo' saque'],2.0)

```

Primeiro elemento é o ano 0.

Quantia sacada no ano 0.

Saldo da conta no ano 0.

Quantia a ser sacada no ano 1.

Saldo da conta no final do ano 1.

while verifica se o saldo corrigido é maior que o próximo saque.

Quantia a ser sacada no ano $n - 1$.

Saldo da conta no ano $n - 1$, pós-saque.

Saldo corrigido no final de um ano.

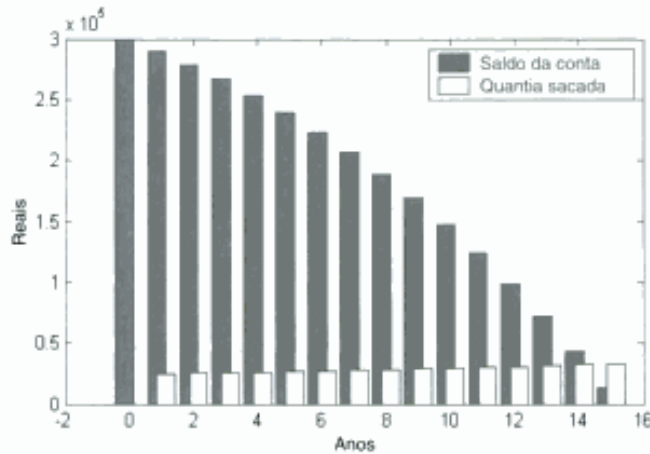
Quantia a ser sacada no final do próximo ano.

```
>> Capitulo7_Exemplo9
```

Os saques serão efetuados durante 15 anos.

A última quantia sacada será de R\$32986.97.

O programa também gera a figura a seguir (os rótulos dos eixos e a legenda foram adicionados manualmente na janela Figure, menu Insert + X Label, Insert + Y Label e Insert + Legend):



Problema-exemplo 7-10: Gerando números para aposta na loteria

Em uma certa loteria, o apostador deve selecionar alguns números dentre uma lista de números predeterminados. Escreva uma função que gere uma lista de n números inteiros distribuídos uniformemente entre os números a e b . Todos os números selecionados na lista devem ser diferentes entre si. Teste a função para gerar:

- uma lista de 6 números escolhidos entre os números 1 e 49.
- uma lista de 8 números escolhidos entre os números 60 e 75.
- uma lista de 9 números escolhidos entre os números -15 e 15.

Solução

A função personalizada mostrada a seguir utiliza o uso da função `rand`, nativa do MATLAB (veja a Seção 3.7). Para assegurar que todos os números sejam diferentes entre si, os números são escolhidos individualmente. Cada número escolhido pela função `rand` é comparado com os demais. Caso haja uma correspondência, o número é descartado e um novo número é escolhido.

O `loop while` verifica o número escolhido, comparando-o com os demais no vetor `x`. Se for encontrada alguma correspondência, continua a escolher números até que uma nova entrada, diferente das demais, seja detectada.

Parte 2: o movimento do foguete desde o instante em que o motor pára até que o pára-quedas seja aberto. Nessa parte, o foguete move-se com uma desaceleração constante $-g$. A velocidade e a altura do foguete em função do tempo são dadas, respectivamente, por:

$$v(t) = v_1 - g(t - t_1) \quad \text{e} \quad h(t) = h_1 + v_1(t - t_1) - \frac{1}{2}g(t - t_1)^2$$

O processo de *looping* continua até que a velocidade final do foguete atinja -20m/s (negativa, porque o foguete move-se para baixo). A altura e o tempo são denotados por h_2 e t_2 .

Parte 3: o movimento a partir da abertura do pára-quedas até que o foguete atinja o solo. O foguete move-se a uma velocidade constante (aceleração zero). A altura em função do tempo é dada por: $h(t) = h_2 - v_{\text{para-quedas}}(t - t_2)$, onde $v_{\text{para-quedas}}$ é a velocidade constante após a abertura do pára-quedas. O processo de *looping* continua enquanto a altura é maior que zero.

Uma rotina que desenvolve esses cálculos é mostrada a seguir:

```
m = 0.05; g=9.81; tmotor = 0.15; Forca=16; vpara_quedas=-20; Dt=0.01;
clear t v h
n=1;
t(n) = 0; v(n) = 0; h(n) = 0;
% Parte 1
a1 = (Forca - m*g)/m;
while t(n)<tmotor & n < 50000
    n = n + 1;
    t(n) = t(n-1) + Dt;
    v(n) = a1*t(n);
    h(n) = 0.5*a1*t(n)^2;
end
v1 = v(n); h1 = h(n); t1 = t(n);
% Parte 2
while v(n) >= vpara_quedas & n < 50000
    n = n + 1;
    t(n) = t(n-1) + Dt;
    v(n) = v1 - g*(t(n)-t1);
    h(n) = h1 + v1*(t(n)-t1) - 0.5*g*(t(n)-t1)^2;
end
v2 = v(n); h2 = h(n); t2 = t(n);
% Parte 3
while h(n) > 0 & n < 50000
    n = n + 1;
    t(n) = t(n-1) + Dt;
```

Primeiro loop while.

Segundo loop while.

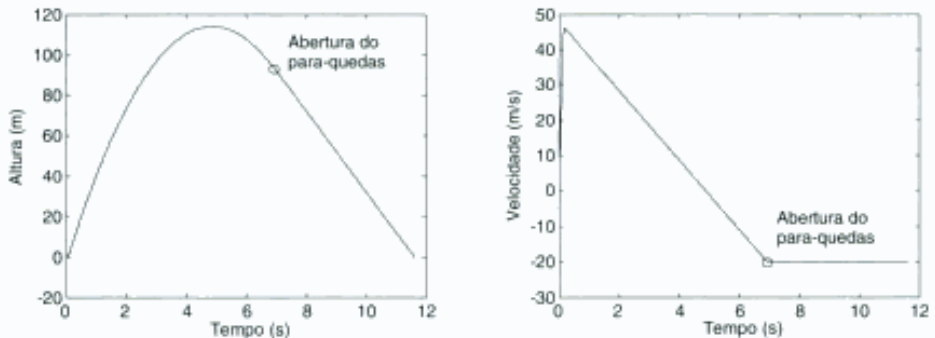
Terceiro loop while.

```

v(n) = vpara_quedas;
h(n) = h2 + vpara_quedas*(t(n)-t2);
end
subplot(1,2,1)
plot(t,h,t2,h2,'o')
subplot(1,2,2)
plot(t,v,t2,v2,'o')

```

A precisão dos resultados depende do valor do incremento Δt . Um incremento de 0.01 apresenta bons resultados. A expressão condicional no comando `while` inclui uma condição para n (se n for maior que 50.000, o laço é terminado). Isso é uma precaução para evitar um *loop* infinito, caso haja um erro em alguma sentença dentro do *loop*. Os gráficos gerados pela rotina (salva como `Capitulo7_Exemplo11`) são mostrados a seguir (os títulos e as escalas adotadas para os eixos foram adicionados manualmente):



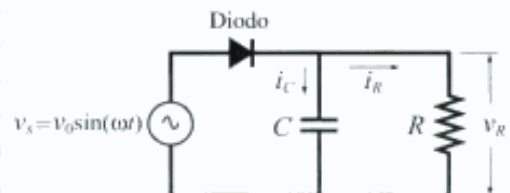
Observação:

O problema pode ser resolvido e programado de diversas formas diferentes. A solução dada aqui é bastante particular. Por exemplo, em vez de usar laços `while`, os instantes de tempo para o pára-quedas ser aberto e o foguete atingir o solo podem ser calculados primeiro e laços `for-end` podem ser utilizados no lugar dos laços `while`. Se os tempos são determinados em primeiro lugar é possível usar cálculos elemento a elemento do vetor, em vez de laços.

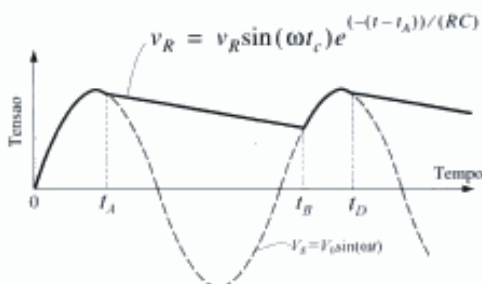
Problema-exemplo 7-12: Conversor CA/CC (circuito retificador)

O circuito retificador de meia-onda a diodo é um circuito eletrônico que, em síntese, converte uma tensão CA em uma tensão CC. Um circuito retificador de meia-onda é constituído por uma fonte CA, um diodo, um capacitor e uma carga (no caso, um resistor), mostrado na figura a seguir.

A tensão da fonte varia de acordo com $v_s = v_m \sin(\omega t)$, onde $\omega = 2\pi f$ e f é a frequência da fonte CA.



A operação do circuito pode ser ilustrada nas formas de onda, onde a linha tracejada corresponde à forma de onda da fonte CA e a linha cheia mostra a queda de tensão através do resistor. No primeiro ciclo, o diodo está ligado (conduzindo corrente) de $t = 0$ a $t = t_A$. Durante o intervalo de tempo que o diodo está desligado (polarizado reversamente), a potência consumida pela carga é toda fornecida pelo capacitor em descarga (de $t = t_A$ a $t = t_B$). Em $t = t_B$, o diodo é ligado novamente (polarizado diretamente) e passa a conduzir corrente, alimentando a carga e repondo as perdas do capacitor até $t = t_D$. O ciclo será repetido enquanto a fonte de tensão estiver ligada. Nessa análise simplificada, consideraremos que o diodo é ideal e que o capacitor não possui nenhuma carga inicial (em $t = 0$). Quando o diodo está ligado, a queda de tensão e a corrente através do resistor são dadas por:



$$v_R = v_0 \sin(\omega t) \quad \text{e} \quad i_R = v_0 \sin(\omega t) / R$$

A corrente no capacitor é:

$$i_C = \omega C v_0 \cos(\omega t)$$

Quando o diodo está desligado, a queda de tensão através da carga é dada por:

$$v_R = v_0 \sin(\omega t_A) e^{-(t-t_A)/(RC)}$$

Os instantes em que o diodo passa ao estado desligado (t_A , t_D , etc.) são calculados a partir da imposição de que $i_R = -i_C$. O diodo chaveia para o estado ligado quando a fonte se torna maior ou igual à tensão no resistor (o instante t_B na figura).

Escreva uma rotina no MATLAB que construa as formas de onda no resistor v_R e na fonte v_s , em função do tempo para $0 \leq t \leq 70\text{ms}$. A resistência da carga é $1.8\text{K}\Omega$, a fonte de tensão possui uma amplitude $v_0 = 12\text{V}$ e $f = 60\text{Hz}$. Estude o efeito da capacitância do capacitor sobre a tensão da carga testando o programa para dois valores de capacitância, $C = 45\mu\text{F}$ e $C = 10\mu\text{F}$.

Solução

A seguir, temos uma rotina que resolve o problema. O programa foi dividido em duas partes: a primeira calcula a tensão v_R quando diodo está no estado ligado e a segunda calcula a tensão v_R quando o diodo está desligado. O comando `switch` é utilizado para comutar entre as duas partes do programa. Os cálculos começam com o diodo no estado ligado (variável `estado = 'on'`) e, quando $i_R - i_C \leq 0$, o valor da variável `estado = 'off'`, comutando para o grupo de comandos que determinam v_R para esse estado. Esses cálculos se repetem até que $v_s \geq v_R$, quando o programa retorna às equações válidas para o diodo no estado ligado.

```
V0 = 12; R = 1800; f = 60;
```

```
Tf = 70e-3; w = 2*pi*f;
```



```

clear t VR Vs
t = 0:0.05e-3:Tf;
n = length(t);
estado = 'on';
for i = 1:n
    Vs(i) = V0*sin(w*t(i));
    switch estado
        case 'on'
            VR(i) = Vs(i);
            iR = Vs(i)/R;
            iC = w*C*V0*cos(w*t(i));
            sumI = iR+iC;
            if sumI <= 0
                estado = 'off';
                tA = t(i);
            end
        case 'off'
            VR(i) = V0*sin(w*tA)*exp(-(t(i)-tA)/(R*C));
            if Vs(i) >= VR(i)
                estado = 'on';
            end
        end
    end
end
plot(t, Vs, ':', t, VR, 'k', 'linewidth', 1)
xlabel('Tempo(s)'); ylabel('Tensao (V)')

```

Atribui 'on' à variável estado.

Determina a tensão da fonte num tempo $t(i)$.

Diodo ligado.

Verifica se $i_R - i_L \leq 0$.

Caso seja, atribui 'off' à variável estado.

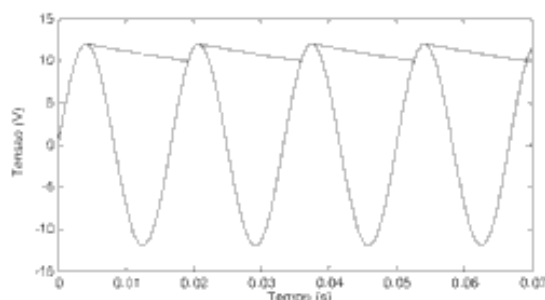
Atribui o valor de $t(i)$ a t_A .

Diodo desligado.

Verifica se $v_i \geq v_R$.

Caso seja, atribui 'on' à variável estado.

Testando a rotina (salva como Capitulo7_Exemplo12) na janela Command Window:


 $C = 45 \mu\text{F}$

6. As temperaturas máximas diárias (em °F) de Nova York (NY) e Anchorage (Alasca), durante o mês de janeiro de 2001, são mostradas nos vetores abaixo (dados da U.S. National Oceanic and Atmospheric Administration).

$\begin{aligned}
 \text{TNY} &= [31 \ 26 \ 30 \ 33 \ 33 \ 39 \ 41 \ 41 \ 34 \ 33 \ 45 \ 42 \ 36 \ 39 \ 37 \ 45 \ 43 \ 36 \ 41 \ 37 \ 32 \ 32 \ 35 \\
 &\quad 42 \ 38 \ 33 \ 40 \ 37 \ 36 \ 51 \ 50]
 \end{aligned}$

$\begin{aligned}
 \text{TANC} &= [37 \ 24 \ 28 \ 25 \ 21 \ 28 \ 46 \ 37 \ 36 \ 20 \ 24 \ 31 \ 34 \ 40 \ 43 \ 36 \ 34 \ 41 \ 42 \ 35 \ 38 \ 36 \\
 &\quad 35 \ 33 \ 42 \ 42 \ 37 \ 26 \ 20 \ 25 \ 31]
 \end{aligned}$

Escreva uma rotina para responder às seguintes questões:

- Calcule a temperatura média mensal em cada cidade.
 - Em quantos dias do mês de janeiro as temperaturas estiveram abaixo da média em cada cidade?
 - Em quantos e em quais dias do mês a temperatura em Anchorage esteve maior que a temperatura em Nova York?
 - Em quantos e em quais dias do mês as temperaturas foram as mesmas em ambas as cidades?
 - Em quantos e em quais dias do mês as temperaturas em ambas as cidades estiveram acima do ponto de congelamento da água (acima de 32°F)?
7. Use o MATLAB de duas formas diferentes, descritas abaixo, para plotar a função:

$$f(x) = \begin{cases} 4^{x+2} & \text{para } -6 \leq x \leq -2 \\ x^2 & \text{para } -2 \leq x \leq -2.5 \\ (x+6.5)^{1/3} & \text{para } -2.5 \leq x \leq -6 \end{cases}$$

- Escrevendo uma rotina composta de sentenças condicionais e laços.
 - Declarando uma função para $f(x)$ e, então, usando-a em uma rotina para construir o gráfico.
8. Escreva uma rotina que determine as raízes reais da equação quadrática $ax^2 + bx + c = 0$. Salve a função como `raizquad`. Quando o arquivo for executado, o usuário deverá entrar com os valores das constantes a , b e c . Para calcular as raízes da equação, o programa deve primeiro calcular o discriminante D dado por:

$$D = b^2 - 4ac$$

Se $D > 0$, o programa deve exibir a mensagem: "A equação possui duas raízes reais.", e as raízes devem ser mostradas na próxima linha.

Se o $D = 0$, o programa deve exibir a mensagem: "A equação possui duas raízes iguais.", e a raiz deve ser mostrada na próxima linha.

Se $D < 0$, o programa deve exibir a mensagem: "A equação não possui raízes reais."

Teste a rotina na janela Command Window três vezes para obter a solução das seguintes equações:

a) $2x^2 + 8x - 3 = 0$

b) $15x^2 + 10x + 5 = 0$

c) $18x^2 + 12x + 2 = 0$

9. Use laços para criar uma matriz 4×7 , onde o valor de cada elemento é a soma dos índices da matriz (número da linha e da coluna do elemento). Por exemplo, o valor elemento $A(2,5)$ é 7.
10. Use laços e sentenças condicionais para criar uma matriz 5×8 , onde o valor de cada elemento é igual à raiz quadrada da soma dos índices do elemento, a não ser que o elemento esteja em uma linha ou coluna onde os índices são números pares. Nesses casos, o valor de um elemento é igual à soma dos quadrados dos índices. (Os índices de um elemento em uma matriz são o número da linha e da coluna do elemento.)
11. Escreva um programa (usando um laço) que determine a soma dos m primeiros termos da série:

$$\sum_{n=0}^m (-1)^n \frac{1}{2n+1} \quad (n = 0, 1, 2, \dots, m)$$

Rode o programa com $m = 10$ e $m = 500$. Compare o resultado encontrado com $\pi/4$. Essa série é denominada série de Leibniz e converge a $\pi/4$.

12. Um vetor é dado por: $x = [15 \ -6 \ 0 \ 8 \ -2 \ 5 \ 4 \ -10 \ 0.5 \ 3]$. Utilizando sentenças condicionais e laços, escreva uma rotina para determinar a soma dos elementos positivos do vetor.
13. Escreva uma rotina para encontrar o menor número inteiro ímpar divisível por 3, cuja terceira potência é maior que 4000. Utilize apenas um laço no programa. O laço deve iniciar em 1 e parar quando o número for encontrado. Em seguida, o programa deve imprimir uma mensagem: "O número requerido é:" e imprime o número.
14. Escreva uma função que ordene os elementos de um vetor do maior para o menor. Utilize $y = \text{ordenadecres}(x)$ para o nome da função e dos argumentos. A função deve receber um vetor x de qualquer tamanho e retornar um vetor y , contendo os elementos de x organizados em ordem decrescente. Não use a função `sort` do MATLAB. Teste sua função em um vetor com 14 elementos distribuídos aleatoriamente entre -30 e 30. Use a função `rand` do MATLAB para gerar o vetor inicial.
15. Escreva uma função que ordene os elementos de uma matriz. Utilize $B = \text{sortmatriz}(A)$ para o nome da função e dos argumentos, onde A é uma

matriz de qualquer dimensão e B é uma matriz com mesma dimensão de A , cujos elementos estão organizados em ordem crescente, linha após linha, onde o elemento $(1,1)$ é o menor e o elemento (m,n) é o maior.

Teste sua função em uma matriz 4×7 cujos elementos sejam números inteiros distribuídos aleatoriamente entre -30 e 30 . Utilize a função `rand` do MATLAB para gerar a matriz inicial.

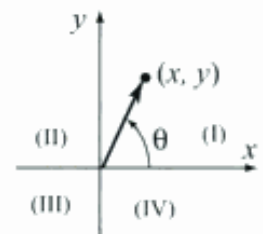
16. Escreva uma rotina que calcule o custo de postagem de um pacote, de acordo com a tabela de preços:

Tipo de serviço	Peso 0 – 2 kg	Peso 2 – 10 kg	Peso 10 – 50 kg
Terrestre	\$1.50	\$1.50 + \$0.50 por quilo (ou fração) que exceder o limite de 2 kg.	\$5.50 + \$0.30 por quilo (ou fração) que exceder o limite de 10 kg.
Aéreo	\$3.00	\$3.00 + \$0.90 por quilo (ou fração) que exceder o limite de 2 kg.	\$10.20 + \$0.60 por quilo (ou fração) que exceder o limite de 10 kg.
Expresso	\$18.00	\$18.00 + \$6.00 por quilo (ou fração) que exceder o limite de 2 kg.	O serviço expresso não está disponível para pacotes acima de 10 kg.

O programa solicita ao usuário que entre com o peso e o tipo de serviço. Então, o programa mostra o custo de postagem. Se um peso maior que 50 kg for digitado para o serviço terrestre ou aéreo é exibida a mensagem “O serviço terrestre ou aéreo não está disponível para pacotes cujos pesos excedam 50 kg”. Se um pacote pesando mais do que 10 kg for digitado para o serviço expresso, será exibida a mensagem “O serviço expresso não está disponível para pacotes acima de 10 kg”. Rode o programa e entre com pacotes pesando 0.5; 6.3; 20; e 50.4 kg para serviços terrestres e aéreos, e 2; 8.1; 13 kg para o serviço expresso.

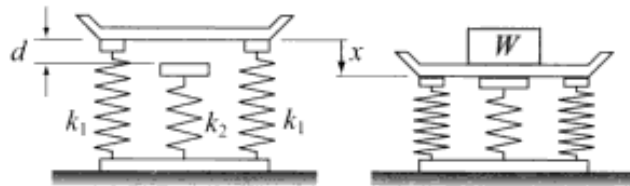
17. Um vetor x é dado por: $x = [1:50]$. Escreva uma rotina que remova todos os elementos do vetor divisíveis por 3, 4 ou 5. No final, apresente o novo vetor.

18. Escreva uma função que determine as coordenadas polares de um ponto escrito em coordenadas cartesianas. Utilize `[teta raio] = Cartesia-no_Polar (x,y)` para o nome da função e dos argumentos. Os argumentos de entradas são as coordenadas x e y do ponto em coordenadas cartesianas e os argumentos de saída são o ângulo θ e a distância radial do ponto à origem. O ângulo θ é dado em graus e é medido relativamente ao eixo x , tal que ele cresça positivamente na direção dos quadrantes



molas de fora suportam o peso da bandeja. Se os objetos tiverem um peso suficiente, a bandeja toca na terceira bandeja do meio.

$k_1 = 800 \text{ N/m}$, $k_2 = 1700 \text{ N/m}$ e $d = 20 \text{ mm}$.



Escreva uma função que calcule o peso W do objeto para um dado deslocamento x da bandeja. Utilize $W = \text{balanca}(\times)$ como nome da função e dos argumentos.

- Utilizando a função na janela Command Window, determine o peso de dois objetos para deslocamentos da bandeja de 1.5 e 3.1 cm.
- Escreva uma rotina que gere os gráficos do peso em função dos deslocamentos para $0 \leq x \leq 4 \text{ cm}$.

8.1 POLINÔMIOS

Polinômios são formas algébricas racionais inteiras. Dito de uma outra maneira, é uma soma finita de monômios da forma:

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + \dots + a_1 x + a_0$$

Os coeficientes $a_n, a_{n-1}, \dots, a_1, a_0$ são números reais* e n é um número inteiro não-negativo que revela o grau ou a ordem do polinômio. Exemplos de polinômios:

$$\begin{array}{ll} f(x) = 5x^5 + 6x^2 + 7x + 3 & \text{Polinômio de grau 5} \\ f(x) = 2x^2 - 4x + 10 & \text{Polinômio de grau 2} \\ f(x) = 11x - 5 & \text{Polinômio de grau 1} \end{array}$$

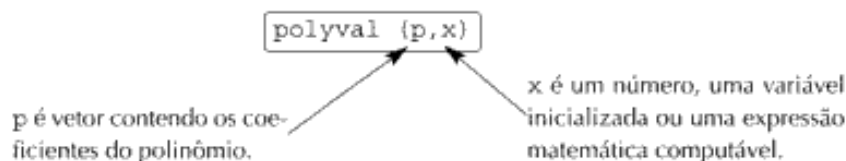
Uma função constante é um polinômio de grau zero. Ex.: $f(x) = 6$.

No MATLAB, polinômios são representados por um vetor linha onde os elementos são os coeficientes $a_n, a_{n-1}, \dots, a_1, a_0$. O primeiro elemento do vetor é o coeficiente do termo de potência mais elevada na variável x . O vetor deve incluir todos os coeficientes, inclusive os coeficientes iguais a zero. Por exemplo:

Polinômio	Representação no MATLAB
$8x + 5$	$p = [8 \ 5]$
$2x^2 - 4x + 10$	$d = [2 \ -4 \ 10]$
$6x^2 - 150$ (forma no MATLAB: $6x^2 + 0x - 150$)	$h = [6 \ 0 \ -150]$
$5x^5 + 6x^2 - 7x$ (forma no MATLAB: $5x^5 + 0x^4 + 0x^3 + 6x^2 - 7x + 0$)	$c = [5 \ 0 \ 0 \ 6 \ -7 \ 0]$

8.1.1 Valor numérico de um polinômio

O valor numérico de um polinômio em um ponto $x = \alpha$ é o resultado da substituição de x por α , efetuando-se as operações pertinentes. No MATLAB, a função `polyval` tem esse propósito. A sintaxe de `polyval` é:



Problema-exemplo 8-1: Determinando valores numéricos de polinômios no MATLAB

Considere o polinômio: $f(x) = x^5 - 12.1x^4 + 40.59x^3 - 17.015x^2 - 71.95x + 35.88$

- Determine $f(9)$.
- Esboce o polinômio para $-1.5 \leq x \leq 6.7$.

* N. de T.: Os polinômios mais gerais possuem coeficientes complexos, ou seja, $a_i \in \mathbb{C} = [0, 1, 2, \dots, n]$.

Solução

O problema é totalmente solucionável na janela Command Window.

- a) Os coeficientes do polinômio são atribuídos ao vetor p . Em seguida, a função `polyval` é utilizada para calcular o valor do polinômio em $x = 9$.

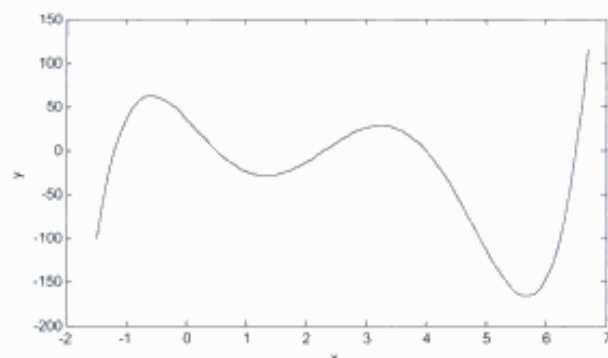
```
>> p = [1 -12.1 40.59 -17.015 -71.59 35.88];
>> polyval(p,9)
ans =
7.2644e+003
```

- b) Para esboçar o polinômio, um vetor auxiliar qualquer (por exemplo, x) é definido inicialmente contendo elementos distribuídos no intervalo -1.5 a 6.7 . Em seguida, um outro vetor (y) é criado para armazenar os valores do polinômio calculado para cada elemento de x . Finalmente, é gerado um gráfico de y versus x .

```
>> x = -1.5:0.1:6.7;
>> y = polyval(p,x);
>> plot(x,y)
```

Calculando o valor numérico do polinômio para cada elemento do vetor x .

O gráfico gerado pelo MATLAB é (os eixos foram rotulados manualmente na janela Plot Editor):



8.1.2 Raízes de um polinômio

As raízes de um polinômio são os valores dos argumentos x ($x = \alpha_i$, $i = 0, 1, \dots, n$; onde n é grau do polinômio) para os quais $P(\alpha_i) = 0$. Por exemplo, as raízes do polinômio $f(x) = x^2 - 2x - 3$ são os valores de x tais que $x^2 - 2x - 3 = 0$, ou seja, $x = -1$ e $x = 3$.

O MATLAB possui uma função nativa, chamada `roots`, que determina a raiz (ou raízes) de um polinômio. A forma funcional de `roots` é:



Por exemplo, as raízes do polinômio do Problema-exemplo 8-1 podem ser determinadas por:

```
>> p = [1 -12.1 40.59 -17.015 -71.59 35.88];
```

```
>> r = roots(p)
```

```
r =
```

```
6.5000
```

```
4.0000
```

```
2.3000
```

```
-1.2000
```

```
0.5000
```

Conhecendo-se as raízes de um polinômio, podemos escrevê-lo na forma fatorada:

$$f(x) = (x + 1.2)(x - 0.5)(x - 2.3)(x - 4)(x - 6.5)$$

O comando `roots` é bastante útil na determinação das raízes de uma equação quadrática. Por exemplo, para determinar as raízes de $f(x) = 4x^2 - 10x - 8$, digite:

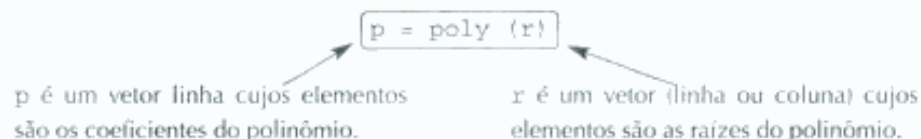
```
>> roots ([4 10 -8])
```

```
ans =
```

```
-3.1375
```

```
0.6375
```

Quando as raízes de um polinômio são conhecidas, o comando `poly` possibilita inverter o processo, ou seja, determinar os coeficientes do polinômio. A forma da função `poly` é:



Por exemplo, os coeficientes do polinômio do Problema-exemplo 8-1 podem ser obtidos a partir das raízes do polinômio:

```
>> r = [6.5 4 2.3 -1.2 0.5];
```

```
>> p = poly(r)
```

```
p =
```

```
1.0000 -12.1000 40.5900 -17.0150 -71.9500 35.8800
```

8.1.3 Adição, Multiplicação e Divisão de Polinômios

Adição:

A adição ou subtração de polinômios é feita adicionando-se ou subtraindo-se os coeficientes dos termos de mesma potência. Se dois polinômios não têm o mesmo grau, isto é, os vetores correspondentes não têm a mesma dimensão, o vetor mais curto deve ser modificado, recebendo zeros, de modo a possibilitar a adição dos vetores no MATLAB. Por exemplo, os polinômios:

$$f_1(x) = 3x^6 + 15x^5 - 10x^3 - 3x^2 + 15x - 40 \quad \text{e} \quad f_2(x) = 3x^3 - 2x - 6$$

Quando adicionados, resultam em:

```
>> p1 = [3 15 0 -10 -3 15 -40];
```

```
>> p2 = [3 0 -2 -6];
```

```
>> p = p1 + [0 0 0 p2]
```

```
p =
```

```
3 15 0 -7 -3 13 -46
```

Três zeros foram adicionados ao vetor p2 de maneira a ajustá-lo ao grau de p1, visto que o grau de p1 é 6 e de p2 é 3.

Multiplicação:

A multiplicação de dois polinômios é feita tomando-se a distributividade em relação à soma entre os coeficientes dos polinômios a serem multiplicados. No MATLAB, há uma função nativa, denominada `conv`, criada especialmente para isso:

```
c = conv (a, b)
```

`c` é um vetor cujos elementos são os coeficientes do polinômio resultante da multiplicação.

`a` e `b` são dois vetores cujos elementos representam os coeficientes dos dois polinômios a serem multiplicados.

- Não é necessário que os dois polinômios possuam o mesmo grau.
- A multiplicação de três ou mais polinômios é feita por meio do uso repetido da função `conv`.

Por exemplo, a multiplicação dos polinômios $f_1(x)$ e $f_2(x)$ acima resulta:

```
>> pm = conv(p1,p2)
```

```
pm =
```

```
9 45 -6 -78 -99 65 -54 -12 -10 240
```

O que significa que a resposta é:

$$9x^9 + 45x^8 - 6x^7 - 78x^6 - 99x^5 + 65x^4 - 54x^3 - 12x^2 - 10x + 240$$

Divisão:

Matematicamente, um polinômio pode ser dividido por outro de diversas formas: método de Descartes ou método dos coeficientes a determinar, divisão por um binômio, divisões sucessivas, etc. O MATLAB possui a função nativa `deconv` cuja funcionalidade gera o resultado da divisão entre polinômios. O formato geral da função `deconv` é:

A Figura 8-2 ilustra como polinômios de diferentes ordens se ajustam a um conjunto de sete pontos, dados por: (0.9; 0.9), (1.5; 1.5), (3; 2.5), (4; 5.1), (6; 4.5), (8; 4.9) e (9.5; 6.3). Os pontos foram ajustados através da função `polyfit` com polinômios de ordem 1 a 6. Cada gráfico na Figura 8-2 traz o mesmo conjunto de pontos, destacados por círculos, e uma curva que se ajusta aos pontos com um polinômio de ordem específica. Está claro que o polinômio correspondendo a $n = 1$ é uma reta e com $n = 2$ é uma linha ligeiramente curva. À medida que o grau do polinômio cresce a curva aproxima-se mais do conjunto de pontos. Quando $n = 6$, um a menos que o número de pontos, a curva toca todos os pontos. Porém, em alguns pontos, a curva tende a desviar-se significativamente da inclinação sugerida pelos dados.

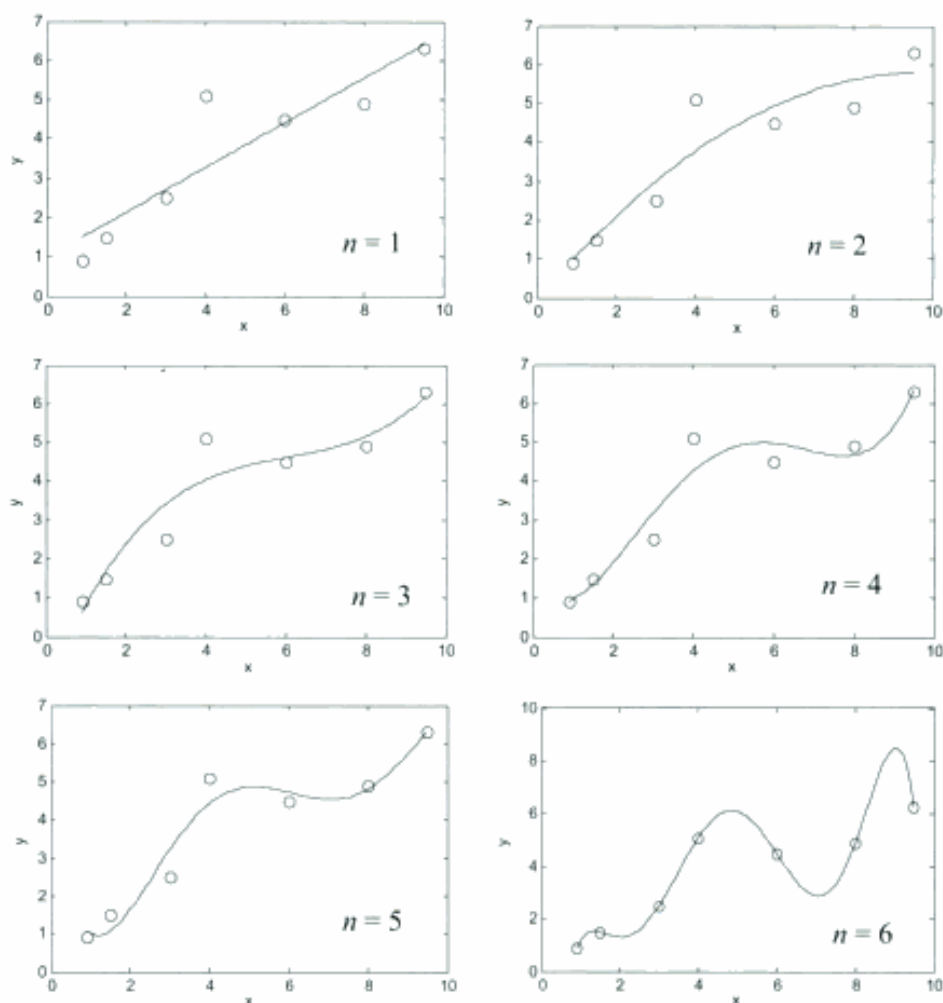


FIGURA 8-2 Ajustando dados com polinômios de ordem diferente.

Uma rotina utilizada na geração dos gráficos da Figura 8-2, para o polinômio com $n = 3$, é mostrada a seguir. Perceba que foi criado um novo vetor `xp`, incrementado de 0.1 entre os limites, de modo a esboçar o polinômio. Este vetor foi utilizado juntamente com a função `polyval` para criar um vetor `yp` contendo os valores do polinômio para cada elemento de `xp`.

$$\ln(y) = mx + \ln(b) \text{ ou } \log(y) = mx + \log(b) \quad (\text{exponencial ou logaritmo na base 10})$$

$$\frac{1}{y} = mx + b \quad (\text{hiperbólica})$$

Todas essas equações exibem relações funcionais lineares entre $\log(y)$ e $\log(x)$ para a potência; $\ln(y)$ ou $\log(y)$ e x para a função exponencial; y e $\ln(x)$ ou $\log(x)$ para a função logarítmica e entre $1/y$ e x para a função hiperbólica. Logo, a função `polyfit(x, y, 1)` aplica-se na determinação das constantes m e b para o melhor ajuste se, em vez de x e y , os seguintes argumentos forem utilizados:

Função		Forma da função <code>polyfit</code>
potência:	$y = bx^m$	<code>p = polyfit(log(x), log(y), 1)</code>
exponencial:	$y = be^{mx}$ ou	<code>p = polyfit(x, log(y), 1)</code> ou
	$y = b10^{mx}$	<code>p = polyfit(x, log10(y), 1)</code>
logarítmica	$y = m\ln(x) + b$ ou	<code>p = polyfit(log(x), y, 1)</code> ou
	$y = m\log(x) + b$	<code>p = polyfit(log10(x), y, 1)</code>
hiperbólica	$y = \frac{1}{mx + b}$	<code>p = polyfit(x, 1./y, 1)</code>

A função `polyfit` age nos argumentos e os resultados são atribuídos a `p`, que é um vetor com dois elementos. O primeiro elemento `p(1)` representa a constante m e o segundo elemento `p(2)` representa a constante b para os termos $\log(x)$ e $\ln(x)$, por exemplo, das funções exponencial e logarítmica.

Para um dado conjunto de pontos é possível prever com alguma exatidão quais funções proporcionam o melhor ajuste. Basta plotar os pontos em um gráfico para diferentes combinações de escalas dos eixos (linear e logarítmica). Se o conjunto de pontos em uma combinação qualquer de escalas aparecer em linha reta, a função correspondente que proporciona o melhor ajuste pode ser uma das listadas abaixo:

Eixo x	Eixo y	Função
linear	linear	linear: $y = mx + b$
logarítmica	logarítmica	potência: $y = bx^m$
linear	logarítmica	exponencial: $y = be^{mx}$ ou $y = b10^{mx}$
logarítmica	linear	logarítmica: $y = m\ln(x) + b$ ou $y = m\log(x) + b$
linear	linear (plot $1/y$)	hiperbólica: $y = \frac{1}{mx + b}$

Outras considerações quanto à escolha de uma função de ajuste:

- Exponenciais não passam pela origem.
- Exponenciais só podem ser utilizadas para o ajuste se todos os pontos tiverem as coordenadas y positivas ou y negativas.

- Funções logarítmicas não podem aproximar de um ponto em $x = 0$ ou valores negativos de x .
- Em uma potência, $y = 0$ quando $x = 0$.
- A equação da hipérbole não está definida para $y = 0$.
- O exemplo a seguir ilustra o processo de ajuste de um conjunto de pontos por uma função.

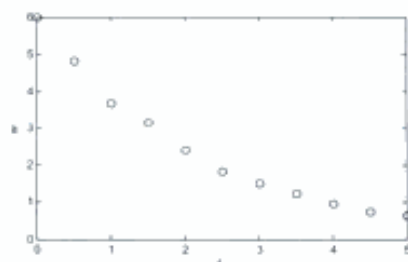
Problema-exemplo 8-2: Ajustando uma equação a um conjunto de ponto

A tabela a seguir mostra um conjunto de pontos medidos em um certo experimento. Determine uma função $w = f(t)$, onde t e w são as variáveis independente e dependente, respectivamente, que melhor se ajustam aos dados da tabela.

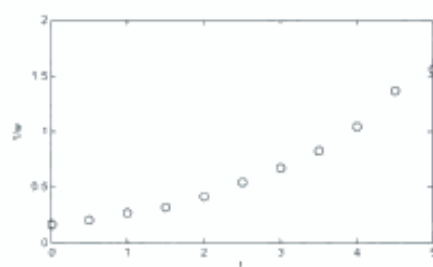
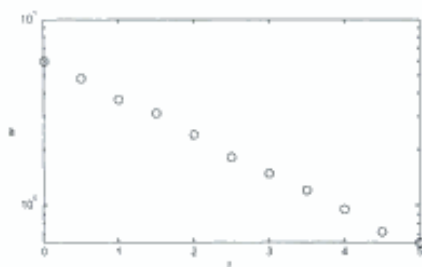
t	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0	4.5	5.0
w	6.00	4.83	3.70	3.15	2.41	1.83	1.49	1.21	0.96	0.73	0.64

Solução

O gráfico é plotado, a princípio, escolhendo-se ambas as escalas lineares. A figura ao lado indica que uma reta não será o melhor ajuste dos pontos, visto que esses não parecem formar um padrão linear. Observando as funções logarítmicas e a lei de potência, a exclusão é imediata porque a função logarítmica tem problemas em $t = 0$ e, para a potência, em $t = 0$, $w \neq 0$.



Para verificar se as demais funções (exponencial e hiperbólica) produzem um ajuste melhor, foram construídos dois outros gráficos adicionais, mostrados abaixo. O gráfico à esquerda possui uma escala monolog (logarítmica no eixo vertical e linear no horizontal). No gráfico à direita, ambos os eixos foram escalonados linearmente e a quantidade $1/w$ foi plotada no eixo vertical.



Na figura da esquerda, os pontos aparecem ao longo de uma linha reta. Isso sugere que uma função exponencial na forma $y = be^{mx}$ produz o melhor ajuste do conjunto de pontos.

Uma rotina para determinar as constantes b e m e gerar o gráfico da função e do conjunto de pontos é mostrada a seguir:

```
t = 0:0.5:5;
```

Declarando os vetores t e w com as coordenadas dos pontos.

```
w = [6 4.83 3.7 3.15 2.41 1.83 1.49 1.21 0.96 0.73 0.64];
```

```
p = polyfit(t, log(w), 1);
```

Usando a função `polyfit` com os argumentos t e $\log(w)$.

```
m = p(1)
```

```
b = exp(p(2))
```

Determina os coeficientes b e m .

```
tm = 0:0.1:5;
```

Criando vetor auxiliar tm a ser utilizado na construção do polinômio.

```
wm = b*exp(m*tm);
```

Calculando o valor da função para cada elemento de tm .

```
plot(t,w,'o',tm,wm)
```

Plotando os dados juntamente com a função de ajuste.

Executando-se a rotina (salva como `Capitulo8_Exemplo2`) na linha do *prompt* da janela Command Window, os valores das constantes de ajuste m e b são exibidos na tela.

```
>> Capitulo8_Exemplo2
```

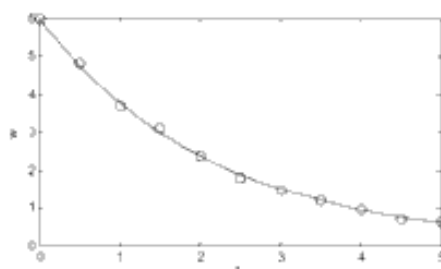
```
m =
```

```
    -0.4580
```

```
b =
```

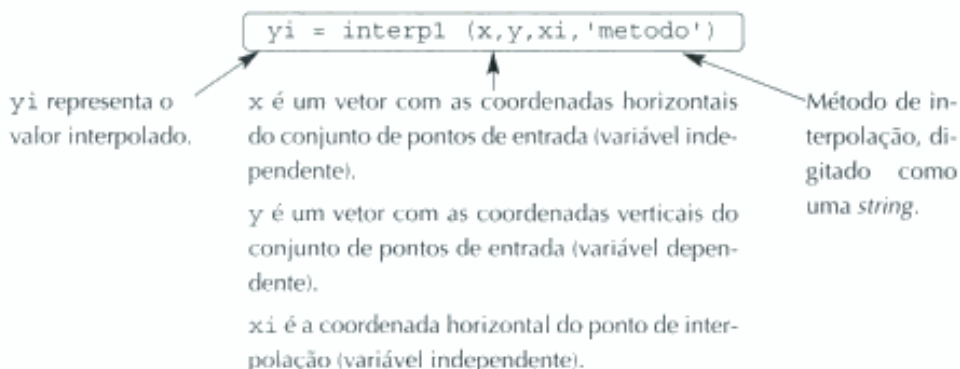
```
    5.9889
```

O gráfico gerado pela rotina mostra o conjunto de pontos e a função (os títulos dos eixos foram adicionados no **menu** Edit + Axes Properties):



Cabe ressaltar que, embora as funções exponencial, potência, logarítmica e hiperbólica tenham sido discutidas nesta seção, muitas outras funções podem, juntamente com a função nativa `polyfit`, ser escritas de maneira a solucionar um problema de ajuste de curvas. No Problema-exemplo 8-7, a função é ajustada por um polinômio de terceira ordem por meio da função `polyfit`.

A função nativa do MATLAB que trata de interpolação 1-D é a função `interp1` (o último caractere é o número um), cuja forma geral é:



- vetor x deve ser monotônico, i.e., elementos devem estar na ordem crescente ou decrescente.
- x_i pode ser um escalar (interpolação de um ponto) ou um vetor (interpolação de muitos pontos). y_i será um número ou um vetor, dependendo dos valores interpolados.
- O MATLAB pode fazer interpolação usando um dos muitos métodos nativos disponíveis. Alguns deles são:

'nearest'	retorna o valor do ponto na vizinhança do ponto interpolado.
'linear'	usa interpolação <i>spline</i> linear.
'spline'	usa interpolação <i>spline</i> cúbica.
'pchip'	utiliza interpolação por partes, cúbica hermitiana, também denominada 'cubic'.

- Quando os métodos 'nearest' e 'linear' são utilizados, os valores de x_i devem pertencer ao domínio de x . Caso sejam usados os métodos 'spline' ou 'pchip', x_i admite valores fora do domínio de x para que a função `interp1` realize a extrapolação.
- O método 'spline' produz grandes erros quando aplicado a um conjunto de pontos não-uniforme, como alguns pontos mais próximos que outros.
- A especificação do método é opcional. Caso não seja especificado, o padrão é 'linear'.

Problema-exemplo 8-3: Interpolação

A tabela a seguir mostra um conjunto de pontos obtidos da função $f(x) = 1.5 \cos(2x)$. Utilize os métodos de interpolação linear, *spline* e *pchip* para calcular o valor de y entre os pontos. Construa gráficos para cada método de interpolação. Mostre nos gráficos os pontos, a curva de ajuste e a curva correspondente ao método de interpolação.

x	0	1	2	3	4	5
y	1.0	-0.6242	-1.4707	3.2406	-0.7366	-6.3717

Solução

A rotina (salva como Capitulo8_Exemplo3) desenvolve os cálculos necessários às diversas interpolações solicitadas.

```
x = 0:1.0:5.0;
```

Declarando vetores x e y com as coordenadas dos pontos.

```
y = [1.0 -0.6242 -1.4707 3.2406 -0.7366 -6.3717];
```

```
xi = 0:0.1:5;
```

Declarando um vetor xi com os pontos de interpolação.

```
yilin = interp1(x,y,xi,'linear');
```

Determina os pontos y das interpolações linear, spline e pchip.

```
ysispl = interp1(x,y,xi,'spline');
```

```
yipch = interp1(x,y,xi,'pchip');
```

```
yfun = 1.5.^xi.*cos(2*xi);
```

Determina os pontos y da função.

```
subplot(1,3,1)
```

```
plot(x,y,'o',xi,yfun,xi,yilin,'--');
```

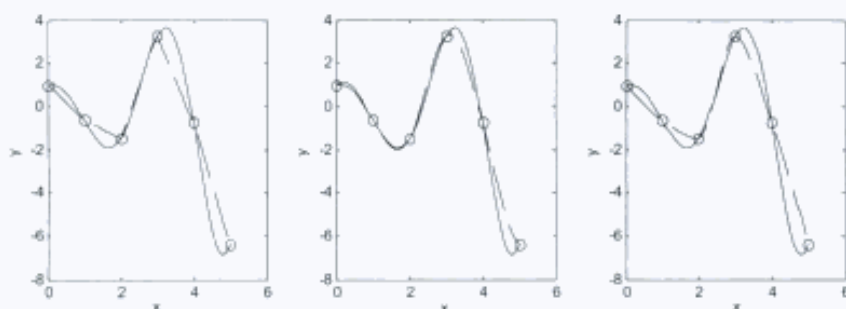
```
subplot(1,3,2)
```

```
plot(x,y,'o',xi,yfun,xi,ysispl,'--');
```

```
subplot(1,3,3)
```

```
plot(x,y,'o',xi,yfun,xi,yipch,'--');
```

Os três gráficos gerados na rotina são mostrados a seguir (os títulos eixos e informações adicionais foram colocadas usando o **menu** Plot Editor). Os pontos estão destacados com círculos, as curvas de interpolação foram plotadas com linhas tracejadas e a função é mostrada em linha cheia. As figuras estão identificadas como linear, spline e pchip.



Coefficients and norm of residuals: mostra os resultados numéricos para um ajuste polinomial selecionado no menu **Fit**. Esse campo traz os coeficientes do polinômio escolhido e as normas dos resíduos. Os resultados podem ser salvos clicando no botão **Save to workspace**.

Find $Y = f(x)$: proporciona um modo rápido de obter interpolação ou extrapolação dos dados numéricos para valores específicos da variável independente (x). Basta digitar um valor da variável x na caixa e clicar sobre o botão **Evaluate**. Se a caixa **Plot evaluated result** estiver selecionada, o ponto escolhido será exibido em destaque no gráfico.

A título de exemplo, a interface Basic Fitting é utilizada para tratar os dados do Problema-exemplo 8-3. As opções selecionadas na janela Basic Fitting são mostradas na Figura 8-3 e os resultados correspondentes às seleções aparecem na Figura 8-4.

A janela Figure Window mostra um conjunto de pontos, uma interpolação (*spline*), dois polinômios (linear e cúbico), com as equações correspondentes, e um ponto específico $x = 1.5$ requerido na caixa **Find $Y = f(x)$** da interface Basic Fitting. A figura também traz um gráfico dos resíduos gerados pelos ajustes polinomiais e as respectivas normas.

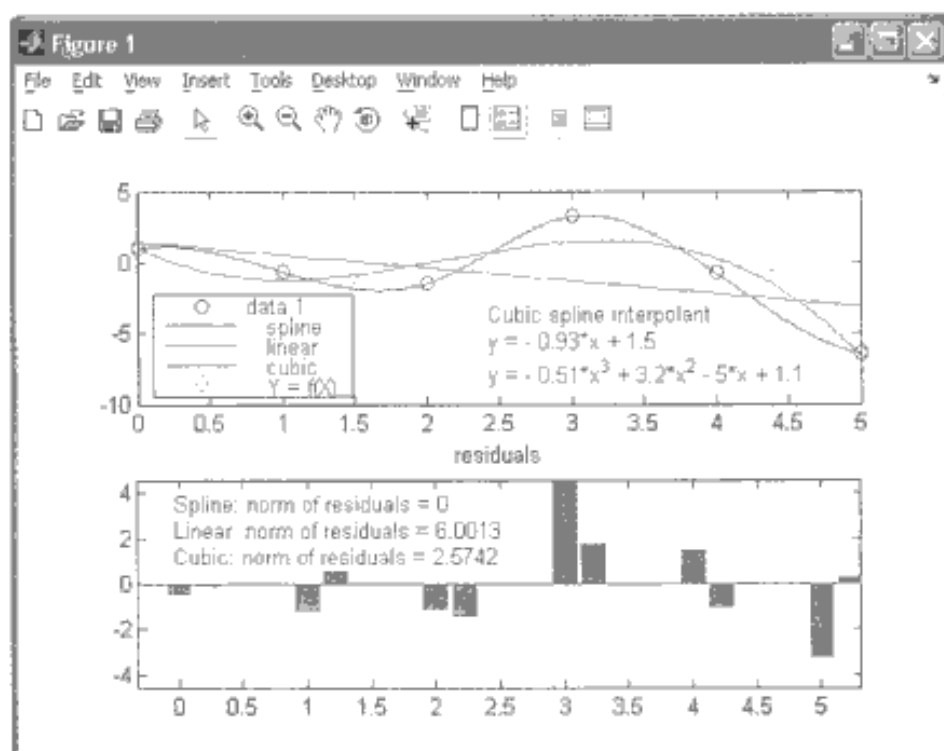
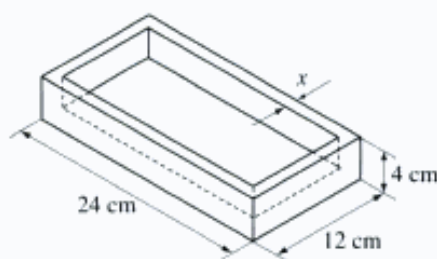


FIGURA 8-4 Um conjunto de pontos modificado por meio da interface Basic Fitting.

8.5 *EXEMPLOS DE APLICAÇÕES DO MATLAB

Problema-exemplo 8-4: Determinação da espessura das paredes de uma caixa

As dimensões externas de uma caixa retangular de alumínio (fundo e as quatro laterais) são 24 cm $\times$ 12 cm $\times$ 4 cm. A espessura da parte do fundo e dos lados aparece, na figura, denotada por x . Determine uma expressão que relacione o peso da caixa com a espessura das paredes x . Determine a espessura x para uma caixa cujo peso é 6.8 kg. O peso específico do alumínio é de 2.8 g/cm³.

**Solução**

O volume da caixa de alumínio V_A é calculado a partir do peso P da caixa por:

$$V_A = \frac{P}{\gamma}$$

onde γ é o peso específico. O volume da caixa de alumínio é determinado por meio das dimensões da caixa:

$$V_A = 24 \cdot 12 \cdot 4 - (24 - 2x)(12 - 2x)(4 - x)$$

onde o volume interno da caixa foi subtraído do volume externo. Essa equação pode ser reescrita convenientemente como:

$$(24 - 2x)(12 - 2x)(4 - x) + V_A - (24 \cdot 12 \cdot 4) = 0$$

que se trata, reconhecidamente, de um polinômio de terceira ordem. Uma das raízes desse polinômio é a solução desejada para x . Uma rotina que determina o polinômio e encontra as raízes é mostrada a seguir:

```
P = 6800; gama = 2.8;
```

```
VAI = P/gama;
```

```
a = [-2 61.0];
```

```
b = [-2 30.5];
```

```
c = [-1 10.2];
```

```
Vint = conv(c,conv(a,b));
```

```
polyeq = [0 0 0 (VAI - 61.0*30.5*10.2)] + Vint
```

```
x = roots(polyeq)
```

Atribui valores a P e gama.

Calcula o volume da caixa de alumínio.

Atribui o polinômio $61 - 2x$ à variável a.

Atribui o polinômio $30.5 - 2x$ à variável b.

Atribui o polinômio $10.2 - x$ à variável c.

Multiplica os três polinômios a, b e c.

Adiciona

$V_A - 61 \cdot 30.5 \cdot 10.2$ a
Vint.

Perceba que na penúltima linha a dimensão dos vetores que representam os polinômios ($\text{VAL} = 61 \times 30,5 \times 10,2$) e Vin foram igualadas para possibilitar a adição dos mesmos (Vin é um polinômio de terceira ordem). Quando a rotina (salva como `Capitulo8_Exemplo4`) é executada, são mostrados os coeficientes do polinômio e o valor de x :

```
>> Capitulo8_Exemplo4
```

```
polyeq =
```

```
-4.0000 223.8000 -3727.1000 2428.6000
```

O polinômio é:

$$-4x^3 + 223,8x^2 - 3727,1x + 2428,6 = 0.$$

```
x =
```

```
27.6355 +11.4248i
```

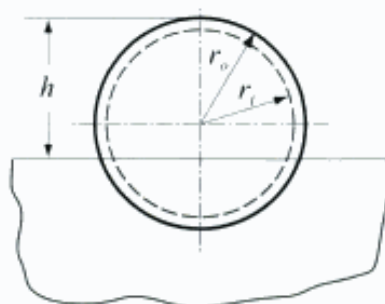
```
27.6355 -11.4248i
```

```
0.6789
```

O polinômio possui uma única raiz real, fisicamente admissível: $x = 0,6789\text{cm}$. Essa é a espessura da parede da caixa de alumínio.

Problema-exemplo 8-5: Cálculo da altura emersa de uma bóia

Uma esfera de alumínio é utilizada como bóia sinalizadora. A esfera possui raio externo igual a 60cm e espessura da capa de alumínio de 12mm. A densidade do alumínio é $\rho_{Al} = 2.690\text{kg/m}^3$. A bóia foi jogada ao mar, onde a densidade da água vale 1.030kg/m^3 . Determine a altura h entre o topo da bóia e a superfície da água.



Solução

De acordo com Arquimedes, o empuxo em um objeto imerso em um fluido, sujeito à ação da gravidade, representa a força que age para cima com módulo igual ao peso do volume do fluido deslocado pelo objeto e cujo ponto de aplicação é o centro de gravidade do mesmo. Desse modo, a esfera de alumínio atingirá uma profundidade correspondente à coluna de água deslocada pela calota esférica imersa na água.

O peso da esfera é dado por:

$$P_{esf} = \rho_{Al} V_{Al} g = \frac{4}{3} \pi (r_o^3 - r_i^3) g \rho_{Al}$$

sendo V_{Al} o volume da esfera de alumínio, r_o e r_i os raios da esfera externo e interno, respectivamente, e g a aceleração da gravidade.

O peso do volume de água deslocado pela calota esférica submersa na água é dado por:

$$P_{\text{água}} = \rho_{\text{água}} V_{\text{água}} g = \frac{1}{3} \pi (2r_o - h)^2 (r_o + h) g \rho_{\text{água}}$$

Exigindo-se que esses pesos sejam iguais, resulta a equação:

$$h^3 - 3r_0h^2 + 4r_0^3 - 4\frac{\rho_{Al}}{\rho_{\text{água}}}(r_0^3 - r_i^3) = 0$$

A última equação é um polinômio de terceira ordem para h . Das raízes do polinômio podemos chegar à resposta do problema.

Uma rotina no MATLAB é elaborada escrevendo-se os polinômios e usando a função `roots` para determinar os valores de h . A seguir, temos essa implementação:

```
ro = 0.60; ri = 0.588;
rhoalum = 2690; rhoagua=1030;
a0 = 4*ro^3 - 4*rhoalum*(ro^3-ri^3)/rhoagua;
p = [1 -3*ro 0 a0];
h = roots(p)
```

Declara as variáveis raio externo e interno, respectivamente.

Declara as densidades do alumínio e da água.

Declara o coeficiente a_0 .

Declara um vetor cujos elementos são os coeficientes do polinômio.

Determina as raízes do polinômio.

Rodando a rotina na janela Command Window (salva como Capitulo8_Exemplo5), são geradas três raízes, possíveis respostas ao problema, porque o polinômio é de terceira ordem. A única resposta fisicamente admissível é a segunda, i.e., aquela para a qual $h = 0.9029\text{m}$.

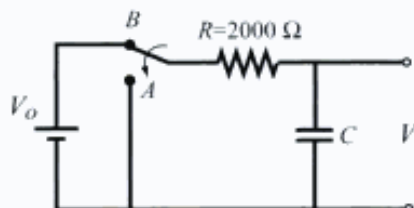
```
>> Capitulo8_Exemplo5
h =
    1.4542
    0.9029
   -0.5570
```

O polinômio possui três raízes. A única resposta fisicamente aceitável é aquela para a qual $h = 0.9029\text{m}$.

Problema-exemplo 8-6: Determinação da capacitância de um capacitor

Um capacitor de capacitância desconhecida foi conectado em um circuito contendo uma fonte de tensão, uma chave bipolar e um resistor.

Primeiramente, a chave é posicionada em B e o capacitor é carregado. Em seguida, a chave é comutada para a posição A e o capacitor descarrega-se através do resistor. Durante a descarga, a tensão sobre os terminais do capacitor é medida em intervalos regulares de 1s , em um período de 10s . A tabela a seguir apresenta os resultados das medidas. Construa



um gráfico da tensão sobre o capacitor em função do tempo e determine a capacitância do mesmo, ajustando o conjunto de pontos com uma curva exponencial.

t (s)	1	2	3	4	5	6	7	8	9	10
V (V)	9.40	7.31	5.15	3.55	2.81	2.04	1.26	0.97	0.74	0.58

Solução

A tensão sobre o capacitor, em função do tempo, durante a descarga é dada por:

$$V = V_0 e^{(-t)/(RC)}$$

sendo V_0 a tensão inicial do capacitor, R a resistência do resistor e C a capacitância do capacitor. De acordo com a explicação da Seção 8.2.2, a função exponencial pode ser linearizada para $\ln(V)$ e t como:

$$\ln(V) = \frac{-1}{RC}t + \ln(V_0)$$

A equação anterior está na forma-padrão $y = mx + b$ e pode ser ajustada ao conjunto de pontos da tabela, utilizando a função `polyfit(x, y, 1)`, com t e $\ln(V)$ sendo, respectivamente, as variáveis independente (x) e dependente (y). Os coeficientes m e b determinados por meio da função `polyfit` são utilizados para encontrar o valor de C e V_0 :

$$C = \frac{-t}{Rm} \quad \text{e} \quad V_0 = e^b$$

A rotina a seguir determina a melhor função de ajuste exponencial aos pontos da tabela, determina C e V_0 e constrói os gráficos dos pontos e da função de ajuste.

```
R = 2000;
```

Declara R.

```
t = 1:10;
```

Declara os vetores t e v com os dados da tabela.

```
v = [9.40 7.31 5.15 3.55 2.81 2.04 1.26 0.97 0.74 0.58];
```

```
p = polyfit(t,log(v),1);
```

Usa a função `polyfit` com t e $\log(V)$ como argumentos.

```
C = -1/(R*p(1))
```

Determina C a partir de $p(1)$, que representa m na equação.

```
V0 = exp(p(2))
```

Determina V_0 a partir de $p(2)$, que representa b na equação.

```
tplot = 0:0.1:10;
```

```
vplot = V0*exp(-tplot./(R*C));
```

Cria os vetores auxiliares `tplot` (do tempo) e `vplot` (da tensão) para a construção do gráfico da função.

Executando a rotina (salva como `Capitulo8_Exemplo6`) na janela Command Window, são impressos na tela os valores de C e V_0 , como mostra a caixa a seguir:

```
>> Capitulo8_Exemplo6
```

```
C =
```

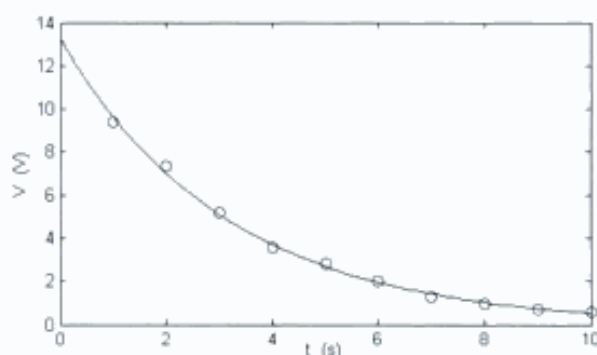
```
0.0016
```

```
V0 =
```

```
13.2796
```

A capacitância do capacitor vale $1600\mu\text{F}$.

Além disso, a rotina gera o seguinte gráfico (os eixos foram rotulados através do menu Plot Editor):



Problema-exemplo 8-7: Dependência térmica da viscosidade

A viscosidade (μ) é a propriedade de resistência que todo fluido real apresenta ao movimento relativo de qualquer de suas partes. Para a maioria dos fluidos, a viscosidade é altamente sensível à variação de temperatura. Abaixo é fornecida uma tabela de viscosidade do óleo SAE 10W a diferentes temperaturas (Dados extraídos de B.R. Munson, D.F. Young e T.H.Okiishi, “Fundamental of Fluid Mechanics”, 4th Edition John Wiley and Sons, 2002). Determine uma equação que se ajusta aos dados:

$T (^{\circ}\text{C})$	-20	0	20	40	60	80	100	120
$\mu(\text{Ns/m}^2)$ ($\times 10^{-3}$)	4	0.38	0.095	0.032	0.015	0.0078	0.0045	0.0032

Solução

Primeiramente é necessário plotar os pontos da viscosidade (μ) em função da temperatura (T – escala absoluta) para avaliar o tipo de equação que permitirá o melhor ajuste dos dados. Uma boa tentativa é adotar uma escala linear para T e logarítmica para μ . Feita a escolha, os pontos marcados nesse gráfico claramente não estão distribuídos ao longo de uma linha reta. Significa que uma função exponencial simples (tipo $y = be^{mx}$) não propor-

ciona o melhor ajuste. Como os pontos na figura aparentam situar-se ao longo de uma linha curva, uma função que possivelmente se ajusta aos dados é:

$$\ln(\mu) = a_2 T^2 + a_1 T + a_0$$

Para plotá-la, é possível lançar mão da função `polyfit(x, y, 2)`, polinômio de segundo grau, nativa do MATLAB, onde a variável independente é T e a variável dependente é $\ln(\mu)$. A equação anterior pode ser resolvida para μ e resultar na viscosidade em função da temperatura:

$$\mu = e^{(a_2 T^2 + a_1 T + a_0)} = e^{a_0} e^{a_1 T} e^{a_2 T^2}$$

A rotina a seguir determina a melhor função de ajuste dos dados e exibe um gráfico mostrando os dados e a função.

```
T = [-20:20:120];
mu = [4 0,38 0,095 0,032 0,015 0,0078 0,0045 0,0032];
TK = T + 273;
p = polyfit(TK,log(mu),2)
Tplot = 273 + [-20:120];
muplot = exp(p(1)*Tplot.^2 + p(2)*Tplot + p(3));
semilogy(TK,mu,'o',Tplot,muplot)
```

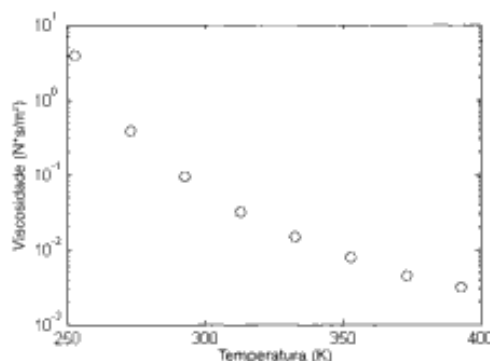
Executando a rotina (salva como `Capitulo8_Exemplo7`), os coeficiente a_0 , a_1 e a_2 são determinados através da função `polyfit` e mostrados na janela Command Window (veja abaixo) como sendo os três elementos de `p`.

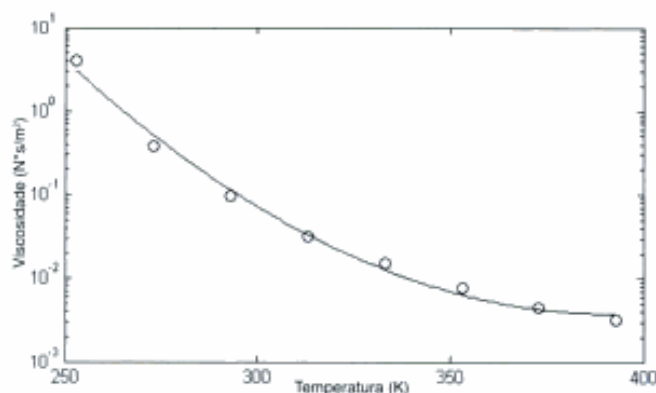
```
>> Capitulo8_Exemplo7
p =
0.0003 -0.2685 47.1673
```

Sendo assim, a viscosidade do óleo em função da temperatura é:

$$\mu = e^{(0.0003T^2 - 0.2685T + 47.1673)} = e^{47.1673} e^{-0.2685T} e^{0.0003T^2}$$

O gráfico mostrando os pontos correspondentes aos dados tabelados e à curva de ajuste é mostrado a seguir (os rótulos dos eixos foram adicionados no Plot Editor).

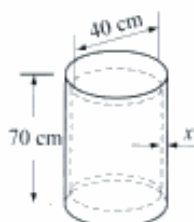




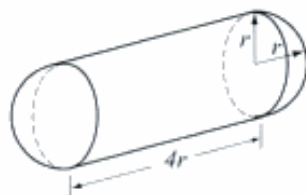
8.6 PROBLEMAS

1. Plote o polinômio $y = 1.5x^4 - 5x^2 + x + 2$ no domínio $-2 \leq x \leq 2$. Crie primeiramente um vetor x . Em seguida, utilize a função `polyval` para determinar y e, por fim, empregue a função `plot`.
2. Divida o polinômio $15x^5 + 35x^4 - 37x^3 - 19x^2 + 41x - 15$ pelo polinômio $5x^3 - 4x + 3$.
3. Divida o polinômio $4x^4 + 6x^3 - 2x^2 - 5x + 3$ pelo polinômio $x^2 + 4x + 2$.

4. Um tanque de combustível de aço inoxidável possui o diâmetro externo de 40 cm e comprimento de 70 cm. As espessuras laterais e da base do cilindro valem x . Determine o valor de x considerando que a massa do tanque é 18 kg. A densidade do aço inoxidável é 7.920 kg/m^3 .



5. Um tanque de gás possui o formato de um cilindro com as extremidades hemisféricas. O raio do cilindro é r e o comprimento é $4r$. Determine r considerando que o volume do tanque é 0.85 m^3 .



6. Escreva uma função que adicione ou subtraia dois polinômios de qualquer ordem. Defina a função como `p = polyadd(p1, p2, oper)`. Os dois primeiros argumentos `p1` e `p2` são vetores cujos elementos são os coeficientes dos dois polinômios. (Caso os polinômios não sejam de mesma ordem, a função deve adicionar zeros ao menor deles). O terceiro argumento (`oper`) é uma *string* que determina se a operação será 'soma' ou 'sub', para adicionar ou subtrair polinômios, respectivamente. O argumento de saída será o polinômio desejado.

Teste a função, adicionando e subtraindo os seguintes polinômios:

$$f_1(x) = x^5 - 7x^4 + 11x^3 - 4x^2 - 5x - 2 \quad \text{e} \quad f_2(x) = 9x^2 - 10x + 6$$

- Determine a função exponencial que melhor se ajusta ao conjunto de pontos. Utilize a função para estimar a população em 1955.
- Ajuste o conjunto de pontos utilizando uma equação quadrática (polinômio de segunda ordem). Utilize a função para estimar a população em 1955.
- Interpole *spline* e linearmente esse conjunto de pontos. Utilize essas funções para estimar a população em 1955.

Em cada uma das letras, construa gráficos do conjunto de pontos, identificando-os com círculos, e das curvas de ajuste e interpolações. Perceba que na letra *c* devem ser feitas duas interpolações. A população real da China em 1955 era 614.4 milhões.

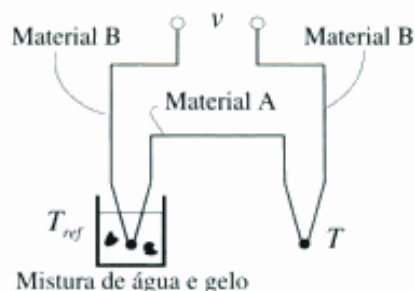
11. As densidades-padrão do ar D (média dos dados coletados) medidas em diferentes alturas (h) em relação ao nível do mar, são mostradas na tabela a seguir:

$h(\text{km})$	0	3	6	9	12	15
$D (\text{kg/m}^3)$	1.2	0.91	0.66	0.47	0.31	0.19
$h(\text{km})$	18	21	24	27	30	33
$D (\text{kg/m}^3)$	0.12	0.075	0.046	0.029	0.018	0.011

- Construa quatro gráficos para o conjunto de pontos (densidade em função da altura). (1) Ambos os eixos com escalas lineares; (2) escala logarítmica para h e linear para D ; (3) escala linear para h e logarítmica para D ; (4) ambas as escalas logarítmicas. Observando os quatro gráficos, escolha uma função (linear, uma potência, exponencial ou logarítmica) que melhor se ajuste ao conjunto de pontos e determine os coeficientes da função.
 - Plote a função, juntamente com o conjunto de pontos, utilizando escalas lineares.
12. Escreva uma função que ajuste certo conjunto de pontos a partir de uma potência da forma: $y = bx^m$. Salve a função como `[b,m] = powerfit(x,y)`, onde os argumentos x e y são vetores contendo as coordenadas dos pontos e os argumentos b e m são constantes da equação de ajuste. Teste a função no conjunto de pontos abaixo. Construa um gráfico que mostre, simultaneamente, os dados e a função.

x	0.5	1.9	3.3	4.7	6.1	7.5
y	0.8	10.1	25.7	59.2	105	122

13. O termopar é um dispositivo sensor bastante utilizado na medição de temperatura. É um sistema constituído por um par de ligas de metais diferentes e no qual se estabelece uma força eletromotriz quando as duas ligas estão em temperaturas diferentes. Para se medir a temperatura, as duas ligas são conectadas formando o circuito mostrado ao lado. Uma junção termopar é colocada em uma temperatura de referência T_{ref} (p. ex., em mistura de água + gelo) e a outra é utilizada para fazer a medição de temperatura T propriamente dita. A tensão v é gerada pela diferença de temperatura entre as junções termopar. Essa tensão v pode ser escrita em função da diferença de temperaturas como segue:



$$v = K_x (T - T_{ref})$$

onde a constante K_x depende essencialmente dos dois materiais utilizados na construção do termopar.

A tabela a seguir apresenta os resultados de um experimento destinado a determinar a constante K_x de um determinado termopar. Usando esses dados, determine K_x ajustando uma curva a esse conjunto de pontos.

$T (^{\circ}\text{C})$	25	100	200	300	400	500	600	700
$v \text{ (mV)}$	1.11	4.03	8.16	12.62	16.54	20.90	23.70	29.15

14. O limite de escoamento σ_v de muitos metais depende do tamanho das partículas. Para esses metais, a relação entre o limite de escoamento e o diâmetro médio das partículas pode ser modelada pela equação de Hall-Petch:

$$\sigma_v = \sigma_0 + k d^{\left(\frac{-1}{2}\right)}$$

A tabela a seguir apresenta algumas medições do diâmetro médio das partículas e o limite de escoamento:

$d(\text{mm})$	0.005	0.009	0.016	0.025	0.040	0.062	0.085	0.110
$\sigma_v \text{ (MPa)}$	205	150	135	97	89	80	70	67

- Usando ajuste de curvas determine as constantes σ_0 e k na equação de Hall-Petch para esse material. Utilizando as constantes, determine o limite de escoamento do material, considerando que o tamanho das partículas é 0,05mm. Construa um gráfico que destaque os dados da tabela com círculos e a curva da equação de Hall-Petch com uma linha cheia.
- Utilize interpolação linear para determinar o limite de escoamento do material cujas partículas têm tamanho de 0,05mm. Construa um gráfico que destaque os dados com círculos e a interpolação linear com uma linha cheia.

que os dados com círculos e a curva da equação de Suzerainty com uma linha sólida.

O ajuste pode ser realizado facilmente reescrevendo a equação na forma:

$$\frac{T^{3/2}}{\mu} = \frac{1}{C}T + \frac{S}{C}$$

e utilizando um polinômio de primeira ordem.

CAPÍTULO 9

Gráficos Tridimensionais

Os gráficos tridimensionais (3-D) são ferramentas poderosas quando se deseja estudar o comportamento de funções multivariáveis. O MATLAB suporta várias funções para tratar e exibir gráficos 3-D. Estão incluídos nesse conjunto de funções curvas no espaço, superfícies, gráficos em rede ou em malha, etc. De posse do gráfico, é possível formatá-lo e animá-lo introduzindo efeitos especiais e muitos outros recursos. Muitas características dos gráficos 3-D são abordadas neste capítulo; outras opções não tratadas aqui podem ser pesquisadas na janela Help Window, opção **Plotting and Data Visualization**.

De qualquer modo, este capítulo é a continuação do Capítulo 5, que tratou especificamente dos gráficos 2-D. Os gráficos 3-D são apresentados em separado neste capítulo propositalmente e porque nem todos os usuários do MATLAB precisam deles. Além disso, para usuários iniciantes do MATLAB, é uma prática fortemente recomendada iniciar pelos gráficos 2-D, estudar os Capítulos 6–8, e então partir para os gráficos 3-D. Presume-se, neste capítulo, que o usuário já esteja familiarizado com os gráficos 2-D.

9.1 CURVAS NO ESPAÇO

A idéia central das curvas no espaço é generalizar curvas no plano, permitindo que sejam introduzidas novas variantes de estudo*, como curvatura, vetor normal e binormal, torção, etc. Basicamente, a maioria dos gráficos 3-D é criada pelo comando `plot3`, cuja sintaxe é bastante similar à do comando `plot`:

* N. de T.: Embora seja possível introduzir tais objetos no plano, é no espaço o habitat natural dessas estruturas matemáticas.

```
plot3 (x, y, z, 'especificadores de linha',
      'Propriedade', Valor_propriedade)
```

x , y e z são os vetores cujas entradas são as coordenadas dos pontos.

(Opcional) Propriedades adicionais usadas para definir a espessura da linha, o tamanho do marcador e da borda, e a cor de preenchimento.

(Opcional) Especificadores de fim o tipo e a cor das linhas e marcadores.

- Os três vetores com as coordenadas dos pontos devem ter a mesma dimensão, ou seja, o mesmo número de elementos.
- Os especificadores de linha, propriedades e o valor da propriedade são exatamente aqueles tratados na Seção 5.1 para gráficos 2-D.

Por exemplo, se as coordenadas x , y e z são fornecidas através de equações paramétricas em t :

$$\begin{aligned}x &= \sqrt{t} \sin(2t) \\ y &= \sqrt{t} \cos(2t) \\ z &= 0.5t\end{aligned}$$

Uma curva pode ser gerada por esse conjunto de equações paramétricas no intervalo $0 \leq t \leq 6\pi$. A rotina a seguir:

```
t = 0:0.1:6*pi;
x = sqrt(t).*sin(2*t);
y = sqrt(t).*cos(2*t);
z = 0.5*t;
plot3(x,y,z,'k','linewidth',1)
grid on
xlabel('x');ylabel('y');zlabel('z');
```

gera a curva mostrada na Figura 9-1:

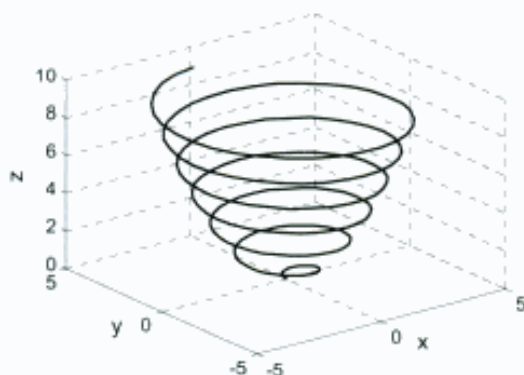


FIGURA 9-1 Um gráfico da função no intervalo $0 \leq t \leq 6\pi$.

9.2 MALHAS E SUPERFÍCIES

As malhas e as superfícies são duas formas de representar graficamente funções da forma $z = f(x, y)$, onde x e y são as variáveis independentes e z é a variável dependente. A equação para z indica que, dado um certo domínio, o valor de z fica completamente estabelecido para quaisquer combinações de x e y . A criação de malhas e superfícies é feita em três etapas. A primeira é a escolha de um *grid* no plano xy que cubra inteiramente o domínio da função, a segunda é o cálculo propriamente dito do valor de z para cada ponto do *grid* e a terceira é a criação do gráfico em si. Abaixo, as etapas são explicadas detalhadamente.

Criando um grid no plano xy:

O *grid* é um conjunto de pontos do domínio de validade ou de definição da função no plano xy . A densidade do *grid*, número de pontos usados para a definição do domínio, fica inteiramente a critério do usuário. A Figura 9-2 mostra, por exemplo, uma escolha de *grid* no domínio $-1 \leq x \leq 3$ e $1 \leq y \leq 4$.

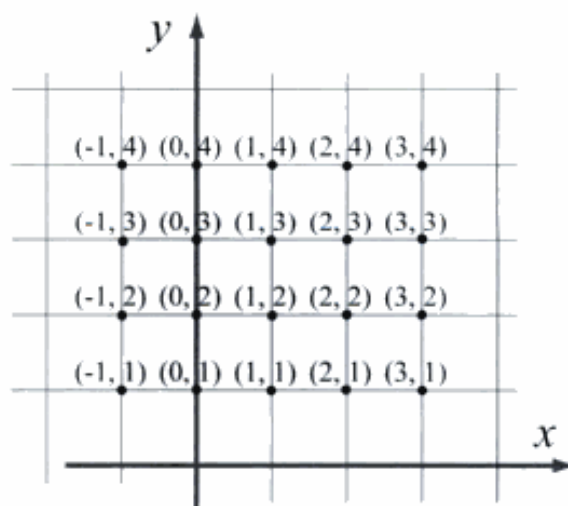


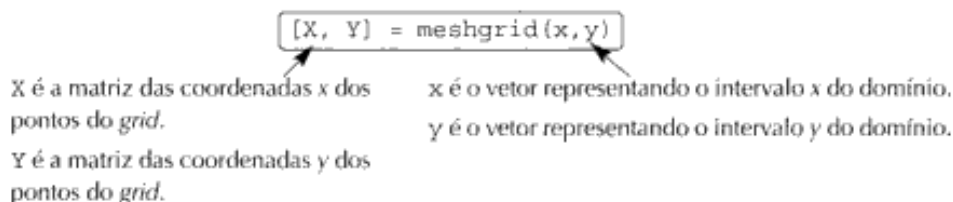
FIGURA 9-2 Um *grid* no plano x - y para o domínio $-1 \leq x \leq 3$ e $1 \leq y \leq 4$.

Nesse *grid*, a distância entre os pontos é unitária. Pontos do *grid* são definidos convenientemente através das matrizes de malhas, por exemplo, X e Y . As matrizes X e Y possuem as coordenadas x e y de todos os pontos do domínio. Para o *grid* acima:

$$X = \begin{bmatrix} -1 & 0 & 1 & 2 & 3 \\ -1 & 0 & 1 & 2 & 3 \\ -1 & 0 & 1 & 2 & 3 \\ -1 & 0 & 1 & 2 & 3 \end{bmatrix} \quad \text{e} \quad Y = \begin{bmatrix} 4 & 4 & 4 & 4 & 4 \\ 3 & 3 & 3 & 3 & 3 \\ 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

A matriz X é feita de linhas idênticas, ou seja, cada linha tem as mesmas coordenadas x . De modo equivalente, a matriz Y tem todas as colunas iguais. Nesse caso, as coordenadas y de todos os pontos são as mesmas.

A função `meshgrid`, nativa do MATLAB, é utilizada freqüentemente para desenvolver a primeira etapa do processo, i.e., criar as matrizes X e Y . A sintaxe da função `meshgrid` é:



Nos vetores x e y o primeiro e último elementos representam os limites (fronteiras) do domínio. A densidade do *grid* é determinada através do número de elementos nos vetores. Por exemplo, as matrizes das malhas X e Y da Figura 9-2 podem ser criadas através do comando `meshgrid` da seguinte maneira:

```
>> x = -1:3;
>> y = 1:4;
>> [X,Y] = meshgrid(x,y)
X =
    -1     0     1     2     3
    -1     0     1     2     3
    -1     0     1     2     3
    -1     0     1     2     3
Y =
     1     1     1     1     1
     2     2     2     2     2
     3     3     3     3     3
     4     4     4     4     4
```

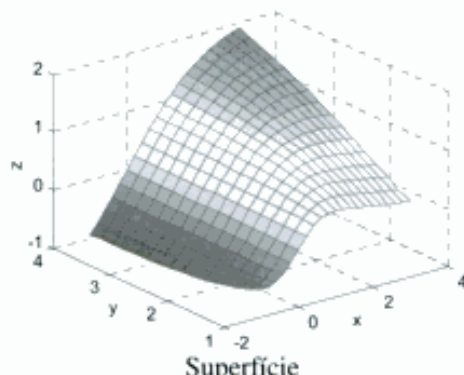
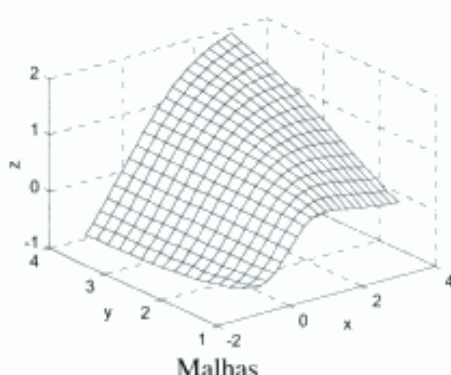
Uma vez criadas as duas matrizes formadoras do *grid*, é possível passar para a próxima etapa do processo: determinar o valor da coordenada z para cada ponto do *grid*.

Determinando o valor de z para cada ponto do *grid*:

Em geral, o valor de z é uma função explícita das coordenadas x e y . Como essas coordenadas são definidas mediante o uso de vetores, o valor da variável z é calculado utilizando-se operações elemento por elemento. Quando as variáveis x e y são matrizes, necessariamente de mesma dimensão, o cálculo da variável dependente ficará armazenado em uma matriz de dimensão igual à dimensão das matrizes originais. Em síntese, o valor de z para cada elemento do arranjo é determinado a partir dos elementos x e y correspondentes. Por exemplo, se a expressão para z for:

$$z = \frac{xy^2}{x^2 + y^2}$$

do *mouse* e escolha a opção *Property Editor Window*. Então, modifique as cores a bel-prazer na lista *Mesh Properties*). No comando `colormap`, *C* é um vetor numérico de três elementos, representando o padrão RGB (Red-Green-Blue). Cada elemento tem intensidade variando entre 0 (mínima) e 1 (máxima). A seguir, estão listadas algumas cores típicas:



$C = [0 \ 0 \ 0]$ (Preto)

$C = [1 \ 0 \ 0]$ (Vermelho)

$C = [0 \ 1 \ 0]$ (Verde)

$C = [0 \ 0 \ 1]$ (Azul)

$C = [1 \ 1 \ 0]$ (Amarelo)

$C = [1 \ 0 \ 1]$ (Magenta)

$C = [0.5 \ 0.5 \ 0.5]$ (Cinza)

- Ao executar os comandos `mesh` e `surf`, o *grid* é mostrado automaticamente. Caso seja necessário desligá-lo, basta digitar o comando `grid off` na linha do *prompt* da janela *Command Window*.
- Os gráficos podem ser “encaixotados” por meio do comando `box on` na janela *Command Window*.
- Os comandos `mesh` e `surf` podem ser utilizados na forma `mesh(Z)` e `surf(z)`. Nesses casos, os valores de *Z* são plotados em função dos endereços (entradas) da matriz. As linhas são representadas no eixo *x* e as colunas no eixo *y*.

Vários outros comandos gráficos possuem funcionalidades similares aos comandos `mesh` e `surf`. A Tabela 9-1 mostra um sumário dos comandos para criação de gráficos em malhas e superfícies do MATLAB. Todos os exemplos na tabela referem-se à função $z = 1.8 \cdot 1.5 \sqrt{x^2 + y^2} \sin(x) \cos(0.5y)$, cobrindo o domínio $-3 \leq x \leq 3$ e $-3 \leq y \leq 3$.

TABELA 9-1 Gráficos em malhas e superfícies

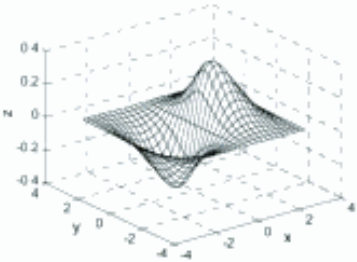
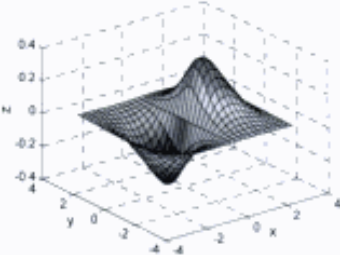
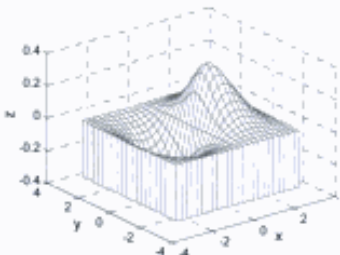
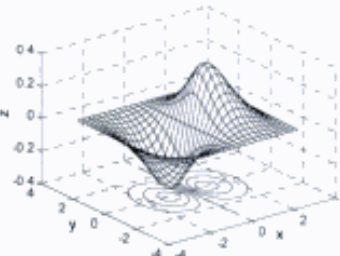
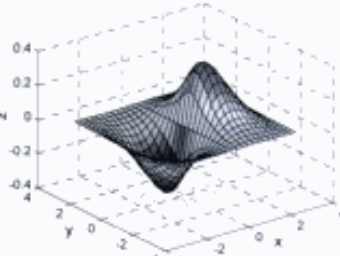
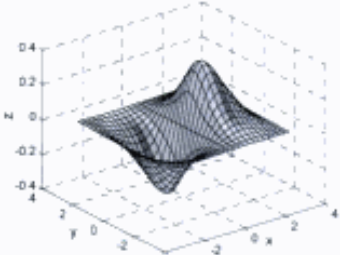
Tipo de gráfico	Exemplo	Rotina
Malhas Sintaxe: <code>mesh(X,Y,Z)</code>		<pre> x = -3:0.25:3; y = -3:0.25:3; [X,Y] = meshgrid(x,y); Z = 1.8.^(-1.5*sqrt(X.^2+... Y.^2)).*cos(0.5*Y).*sin(X); mesh(X,Y,Z) xlabel('x'); ylabel('y'); zlabel('z');</pre>

TABELA 9-1 Gráficos em malhas e superfícies

(continuação)

Tipo de gráfico	Exemplo	Rotina
<u>Superfície</u> Sintaxe: <code>surf(X,Y,Z)</code>		<pre>x = -3:0.25:3; y = -3:0.25:3; [X,Y] = meshgrid(x,y); Z = 1.8.^(-1.5*sqrt(X.^2 + ... Y.^2)).*cos(0.5*Y).*sin(X); surf(X,Y,Z) xlabel('x'); ylabel('y'); zlabel('z');</pre>
<u>Malhas e persianas verticais</u> (desenha as projeções ortogonais [diretrizes] das bordas do gráfico em malhas, formando uma espécie de persiana vertical). Sintaxe: <code>meshz(X,Y,Z)</code>		<pre>x = -3:0.25:3; y = -3:0.25:3; [X,Y] = meshgrid(x,y); Z = 1.8.^(-1.5*sqrt(X.^2 + ... Y.^2)).*cos(0.5*Y).*sin(X); meshz(X,Y,Z) xlabel('x'); ylabel('y'); zlabel('z');</pre>
<u>Malhas e curvas de nível</u> (desenha as curvas de nível do gráfico em malhas no plano xy). Sintaxe: <code>meshc(X,Y,Z)</code>		<pre>x = -3:0.25:3; y = -3:0.25:3; [X,Y] = meshgrid(x,y); Z = 1.8.^(-1.5*sqrt(X.^2 + ... Y.^2)).*cos(0.5*Y).*sin(X); meshc(X,Y,Z) xlabel('x'); ylabel('y'); zlabel('z');</pre>
<u>Superfície e curvas de nível</u> (desenha as curvas de nível da superfície no plano xy). Sintaxe: <code>surfc(X,Y,Z)</code>		<pre>x = -3:0.25:3; y = -3:0.25:3; [X,Y] = meshgrid(x,y); Z = 1.8.^(-1.5*sqrt(X.^2 + ... Y.^2)).*cos(0.5*Y).*sin(X); surfc(X,Y,Z) xlabel('x'); ylabel('y'); zlabel('z');</pre>
<u>Superfície em degrade*</u> (gradiente de cores) Sintaxe: <code>surf1(X,Y,Z)</code>		<pre>x = -3:0.25:3; y = -3:0.25:3; [X,Y] = meshgrid(x,y); Z = 1.8.^(-1.5*sqrt(X.^2 + ... Y.^2)).*cos(0.5*Y).*sin(X); surf1(X,Y,Z) xlabel('x'); ylabel('y'); zlabel('z');</pre>

* N. de T.: Ou *dégradé*, como sugere o Dicionário Aurélio.

TABELA 9-2 Gráficos 3-D especiais

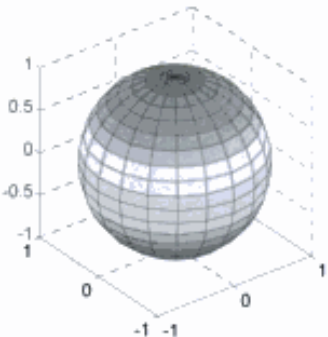
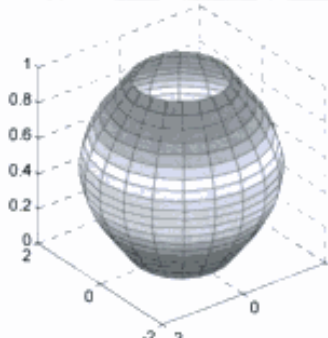
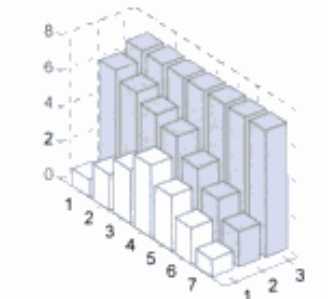
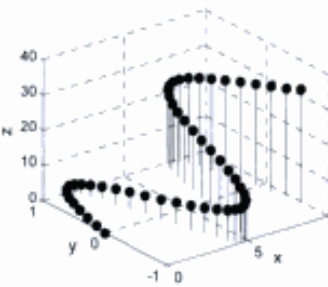
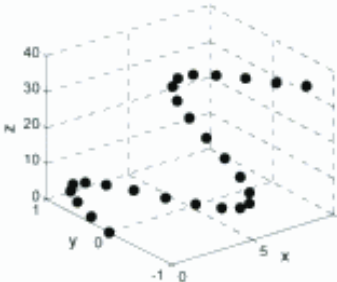
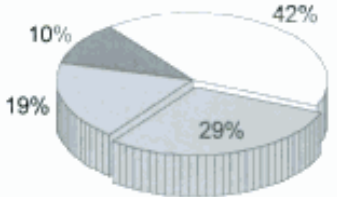
Tipo de gráfico	Exemplo	Rotina
<u>Esfera</u> Sintaxe: <code>sphere</code> Retorna as coordenadas x, y e z de uma esfera unitária composta de 20 setores esféricos (ou seja, 20 faces) Sintaxe: <code>sphere(n)</code> Retorna as coordenadas x, y e z de uma esfera unitária composta de n setores esféricos (ou seja, n faces)		<code>sphere</code> ou: <code>[X,Y,Z] = sphere(20);</code> <code>surf(X,Y,Z)</code>
<u>Cilindro</u> Sintaxe: <code>[X,Y,Z] = cylinder(r)</code> Retorna as coordenadas x, y e z de um cilindro com um perfil r.		<code>t = linspace(0,pi,20);</code> <code>r = 1 + sin(t);</code> <code>[X,Y,Z] = cylinder(r);</code> <code>surf(X,Y,Z)</code> <code>axis square</code>
<u>Barras 3-D</u> Sintaxe: <code>bar3(Y)</code> Cada elemento em Y é desenhado como uma barra. As colunas são alinhadas.		<code>Y = [1 6.5 7; 2 6 7; 3 5.5 7; 4 5 7; 3 4 7; 2 3 7; 1 2 7];</code> <code>bar3(Y)</code>
<u>Hastes 3-D</u> (Desenha uma sequência de pontos e as respectivas projeções ortogonais, as hastes, sobre o plano xy) Sintaxe: <code>stem3(X,Y,Z)</code>		<code>t = 0:0.4:10;</code> <code>x = t;</code> <code>y = sin(x);</code> <code>z = t.^1.5;</code> <code>stem3(x,y,z,'fill')</code> <code>grid on</code> <code>xlabel('x'); ylabel('y');</code> <code>zlabel('z');</code>

TABELA 9-2 Gráficos 3-D especiais

(continuação)

Tipo de gráfico	Exemplo	Rotina
<u>Dispersão 3-D</u> Sintaxe: <code>scatter3(X,Y,Z)</code>		<pre>(X,Y,Z) t = 0:0.4:10; x = t; y = sin(x); z = t.^1.5; scatter3(x,y,z,'filled') grid on colormap([0.1 0.1 0.1]) xlabel('x'); ylabel('y'); zlabel('z');</pre>
<u>Pizza 3-D</u> Sintaxe: <code>pie3(X,explode)</code>		<pre>X = [5 9 14 20]; explode = [0 0 1 0]; pie3(X,explode)</pre> <code>explode</code> é um vetor (com a mesma dimensão de <code>X</code>) de 0s e 1s. A entrada em 1 desloca do centro a fatia correspondente.

Os exemplos mostrados nas tabela são algumas das diversas opções disponíveis em cada tipo de gráfico. Mais detalhes podem ser obtidos na janela Help Window, ou no *prompt* da janela Command Window, digitando `help nome_do_comando`.

9.4 O COMANDO `view`

O comando `view` controla a direção observação do gráfico. O controle é feito através da especificação em termos dos ângulos de azimute e de elevação (veja a Figura 9-3) ou definindo um ponto no espaço de onde o gráfico é observado. Para configurar o ângulo de visão, utilize as seguintes sintaxes do comando `view`:

`view(az,el)` ou `view([az,el])`

- `az` é o ângulo de azimute (em graus), ou seja, é a distância angular, medida sobre o plano xy , a partir do eixo y negativo, definido como positivo no sentido anti-horário.
- `el` é o ângulo de elevação (em graus) do plano xy . Os valores positivos correspondem aos ângulos de abertura tomados na direção do eixo z positivo.
- Os ângulos *default* são `az = -37.5°` e `el = 30°`.

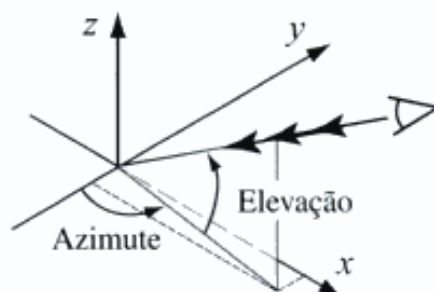


FIGURA 9-3 Ângulos elevação e azimute.

Por exemplo, a superfície da Tabela 9-1 foi plotada novamente na Figura 9-4 com os ângulos de observação $az = 20^\circ$ e $el = 35^\circ$.

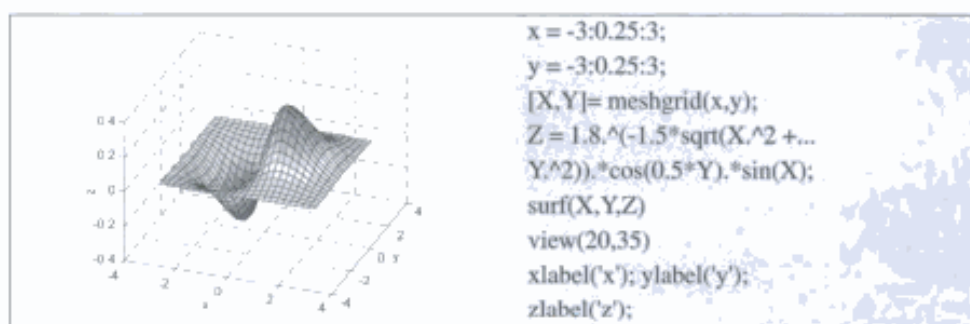


FIGURA 9-4 Uma superfície da função $z = 1.8 \cdot (-1.5 \sqrt{x^2 + y^2}) \cdot \sin(x) \cos(0.5y)$, vista a partir dos ângulos de $az = 20^\circ$ e $el = 35^\circ$.

- Escolhendo-se convenientemente os ângulos de azimute e de elevação, o comando `view` permite plotar as projeções do gráfico 3-D sobre vários planos. A seguir, temos os três casos mais importantes:

Plano de projeção	Valor az	Valor el
x-y (vista superior)	0	90
x-z (vista lateral)	0	0
y-z (vista lateral)	90	0

Um exemplo de vista no plano xy (superior) é mostrado logo adiante. A figura destacada é a função da Figura 9-1.

Exemplos de projeções sobre os planos xz e yz são mostrados nas Figuras 9-6 e 9-7, respectivamente. As figuras mostram as projeções do gráfico em malhas plotado na Tabela 9-1.

- Existem ainda outras duas opções *default*:
`view(2)` Seleciona a projeção do gráfico no plano xy , isto é, $az = 0^\circ$ e $el = 90^\circ$.
`view(3)` Seleciona a vista-padrão 3-D do gráfico no plano xy , isto é, $az = -37.5^\circ$ e $el = 30^\circ$.

- Finalmente, a direção de observação também pode ser especificada a partir de um ponto no espaço de onde o gráfico é visualizado. Nesse caso, a sintaxe a ser utilizada no comando `view` é `view([x, y, z])`, onde x , y e z são as coordenadas do ponto de observação. A direção é determinada pela direção do ponto especificado, relativamente à origem do sistema de coordenadas, e independe da distância ponto-origem. Isso significa que observar do ponto $[6,6,6]$ produz o mesmo efeito que observar do ponto $[10,10,10]$. A tomada superior é feita através do vetor $[0,0,1]$. A vista lateral sobre o plano xz na direção negativa do eixo y é escolhida pelo vetor $[0,-1,0]$, e assim por diante.

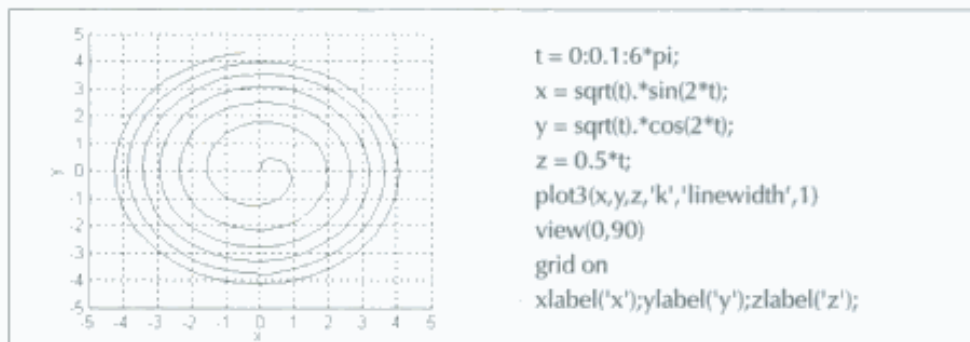


FIGURA 9-5 Vista superior do gráfico da função $x = \sqrt{t} \sin(2t)$, $y = \sqrt{t} \cos(2t)$, $z = 0.5t$ para $0 \leq t \leq 6\pi$.

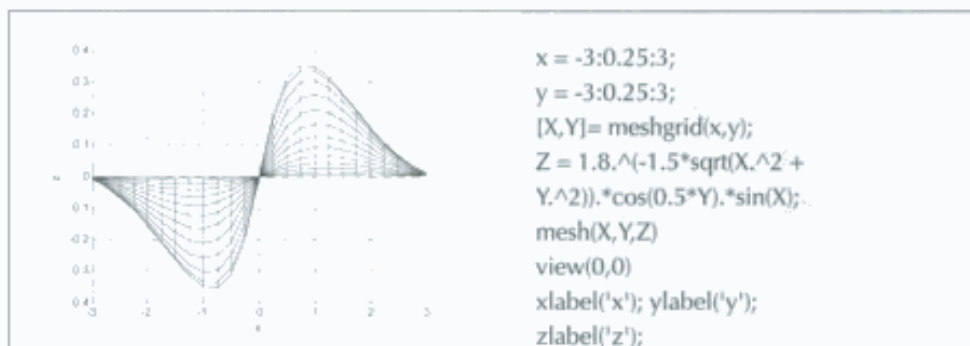


FIGURA 9-6 Projeções no plano xz do gráfico da função $z = 1.8^{-1.5\sqrt{x^2+y^2}} \sin(x) \cos(0.5y)$.

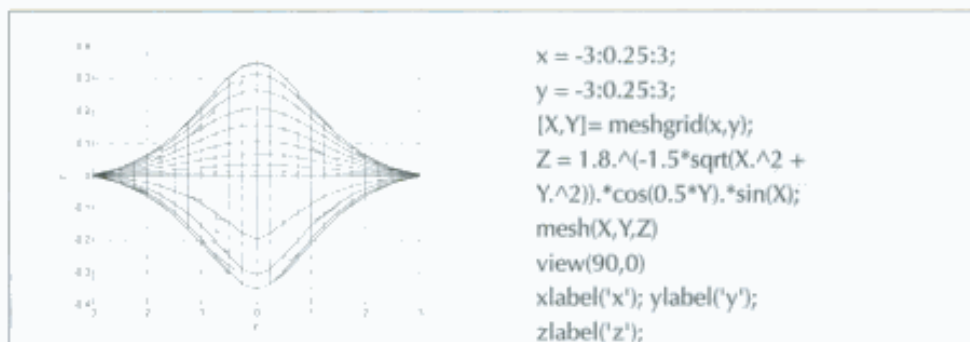


FIGURA 9-7 Projeções no plano yz do gráfico da função $z = 1.8^{-1.5\sqrt{x^2+y^2}} \sin(x) \cos(0.5y)$.

```

v0 = 250; g = 9.81; teta = 65;
x0 = 3000; vx = -30;
v0z = v0*sin(teta*pi/180);
v0y = v0*cos(teta*pi/180);
t = 2*v0z/g;
tplot = linspace(0,t,100);
z = v0z*tplot - 0.5*g*tplot.^2;
y = v0y*tplot;
x = x0 + vx*tplot;
xsem_vento(1:length(y)) = x0;
plot3(x,y,z,'x',xsem_vento,y,z,'-')
grid on
axis([0 6000 0 6000 0 2500])
xlabel('x (m)'); ylabel('y (m)'); zlabel('z (m)')

```

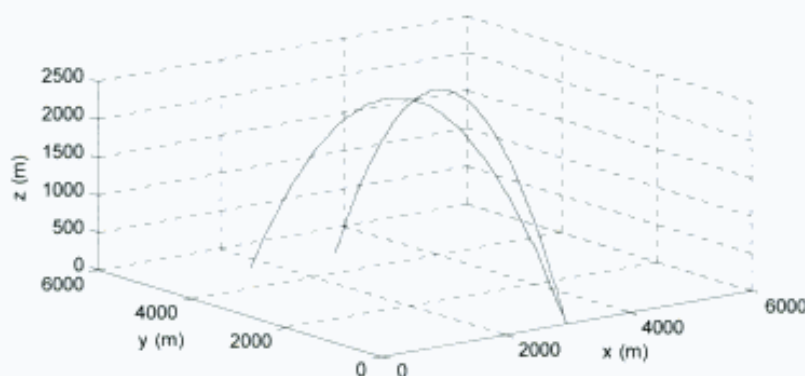
Declarando um vetor tempo com 100 elementos.

Determinando as coordenadas x, y e z do projétil em cada instante de tempo.

Coordenada x constante (sem a ação do vento).

Plotando as duas trajetórias no espaço.

As trajetórias geradas pela rotina (salva como `Capitulo9_Exemplo1`) são mostradas abaixo:



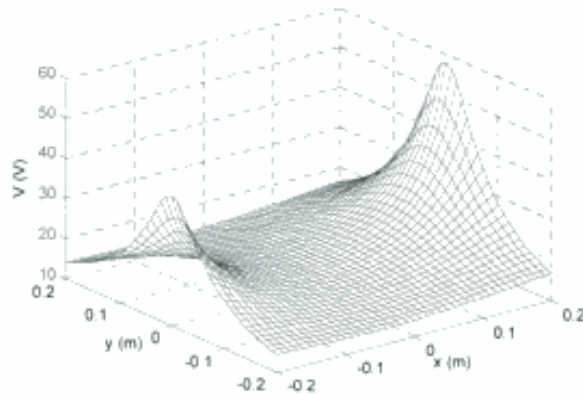
Problema-exemplo 9-2: Potencial elétrico de duas cargas pontuais

O potencial elétrico V produzido por uma partícula carregada é dado por:

$$V = \frac{1}{4\pi\epsilon_0} \frac{q}{r}$$

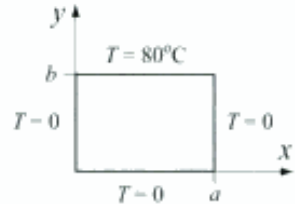
onde $\epsilon_0 = 8.8541878 \times 10^{-12} \frac{\text{C}}{\text{N} \cdot \text{m}^2}$ é a permissividade do vácuo, q é o módulo da carga elétrica em Coulombs e r é a distância radial da partícula (em metros) onde se deseja conhecer o potencial. O potencial elétrico de duas ou mais partículas carregadas é calculado

Executando a rotina na linha do *prompt* da janela Command Window:



Problema-exemplo 9-3: Condução de calor em uma placa retangular

Três lados de uma placa retangular ($a = 5\text{m}$, $b = 4\text{m}$) são mantidos à temperatura $T = 0^\circ\text{C}$, e o outro lado é mantido à temperatura $T_1 = 80^\circ\text{C}$ (veja a figura ao lado). Determine e plote a distribuição de temperatura $T(x, y)$ da placa.



Solução

A distribuição de temperatura $T(x, y)$ em uma placa retangular pode ser determinada resolvendo-se a equação do calor bidimensional (a duas variáveis x, y). Para as condições de contorno fornecidas no problema, $T(x, y)$ pode ser expressa analiticamente em termos da série de Fourier (veja Erwin Kreyszig, "Advanced Engineering Mathematics", John Wiley and Sons, 1993):

$$T(x, y) = \frac{4T_1}{\pi} \sum_{n=1}^{\infty} \frac{\sin\left[(2n-1)\frac{\pi x}{a}\right] \sinh\left[(2n-1)\frac{\pi y}{a}\right]}{(2n-1) \sinh\left[(2n-1)\frac{\pi b}{a}\right]}$$

A rotina a seguir resolve este problema em três etapas:

- Primeiramente, cria um *grid* X, Y no domínio $0 \leq x \leq a$ e $0 \leq y \leq b$. O comprimento a da placa foi dividido em 20 partes e o comprimento b em 16 partes.
- Em seguida, determina a temperatura em cada ponto da malha. Os cálculos são feitos ponto a ponto usando um laço duplo. Para cada ponto, a temperatura é determinada e são adicionados os k termos da série de Fourier.
- Por último, são construídas as superfícies para T (para k termos da série de Fourier).

9.6 PROBLEMAS

1. A posição de uma partícula pode ser escrita de acordo com as equações paramétricas em t por:

$$x = (2 + 4 \cos(t)) \cos(t)$$

$$y = (2 + 4 \cos(t)) \sin(t)$$

$$z = t^2$$

Plote um gráfico da posição da partícula para $0 \leq t \leq 20$.

2. Uma escada em espiral pode ser modelada pelo seguinte conjunto de equações paramétricas:

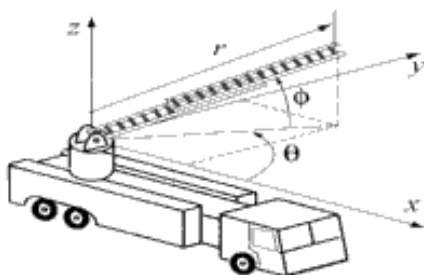
$$x = R \cos\left(2\pi n \frac{t}{h}\right)$$

$$y = R \sin\left(2\pi n \frac{t}{h}\right)$$

$$z = \frac{t}{h}$$

onde R é o raio da escada, h é a altura entre os andares e n é o número de revoluções que a escada faz em cada andar. Uma construção possui 2 andares com $h = 3$ m. Construa gráficos de duas possíveis escadas: uma com $R = 1.5$ m e $n = 3$, e outra com $R = 4$ m e $n = 2$. Represente os dois gráficos na mesma figura.

3. A escada de um caminhão do Corpo de Bombeiros pode ser erguida verticalmente (ângulo ϕ), girada em relação ao eixo z (ângulo θ) e alongada (distância r). Inicialmente, a escada repousa sobre o caminhão, $\phi = 0$, $\theta = 0$ e $r = 8$ m. À medida que a escada é movida para uma nova posição, os ângulos ϕ e θ aumentam a uma taxa de 5 graus/s e 8 graus/s, respectivamente, e a escada é alongada a uma taxa de 0.6 m/s. Determine e plote a posição da extremidade (ponta) da escada para 10 segundos de operação.



4. Construa a superfície 3-D e as órbitas da função $z = -\frac{x^2}{4} - \frac{y^2}{4}$ no domínio $-4 \leq x \leq 4$ e $-4 \leq y \leq 4$.
5. Construa a superfície 3-D e as respectivas curvas de nível (na mesma figura) da função $z = (y + 3)^2 + 1.5x^2 - x^2y$ no domínio $-3 \leq x \leq 3$ e $-3 \leq y \leq 3$.

6. A equação do gás ideal relaciona pressão, temperatura e volume de um gás através da equação:

$$P = \frac{nRT}{V}$$

onde P é a pressão em Pa, n é o número de moles, $R = 8.31 \text{ J/mol-K}$ é a constante universal dos gases, T é a temperatura em Kelvin e V é o volume em m^3 .

Faça um gráfico 3-D que mostre a dependência da pressão (eixo z) com o volume (eixo x) e a temperatura (eixo y) de um mol de gás. Os domínios para o volume e a temperatura são: $0.5 \times 10^{-3} \leq V \leq 2.0 \times 10^{-3} \text{ m}^3$ e $273 \leq T \leq 473 \text{ K}$.

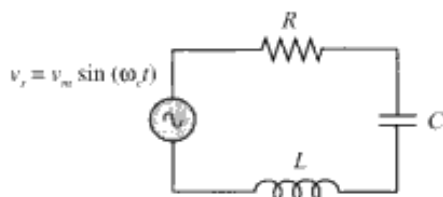
7. As moléculas de um gás encerrado em um recipiente movem-se aleatoriamente em diferentes velocidades. A lei de distribuição de velocidades de Maxwell dá a distribuição de probabilidade $P(v)$ em função da temperatura e da velocidade das moléculas:

$$P(v) = 4\pi \left(\frac{M}{2\pi RT} \right)^{3/2} v^2 e^{(-Mv^2)/(2RT)}$$

onde M é a massa molar do gás em kg/mol , $R = 8.31 \text{ J/mol-K}$ é a constante universal dos gases, T é a temperatura em Kelvin e v é a velocidade das moléculas em m/s .

Plote um gráfico 3-D de $P(v)$ em função de v e T para $0 \leq v \leq 1000 \text{ m/s}$ e $70 \leq T \leq 320 \text{ K}$ para o gás oxigênio (massa molar 0.032 kg/mol).

8. Um circuito RLC é excitado através de uma fonte de tensão alternada senoidal (veja a figura). A fonte de tensão v_s varia no tempo de acordo com $v_s = v_m \sin(\omega_e t)$, onde $\omega_e = 2\pi f_e$, com f_e sendo a frequência de excitação da fonte. Passado o regime transitório, a amplitude da corrente I no circuito é dada por:



$$I = \frac{v_m}{\sqrt{R^2 + (\omega_e L - 1/(\omega_e C))^2}}$$

onde R é a resistência do resistor em Ohm (Ω), C é a capacitância do capacitor em Farad (F) e L é a indutância do indutor em Henry (H). Para o circuito da figura, $C = 15 \mu\text{F}$, $L = 240 \text{ mH}$ e $v_m = 24 \text{ V}$.

- Construa um gráfico 3-D de I (eixo z) em função de ω_e (eixo x) e R (eixo y) no domínio $60 \leq f \leq 110 \text{ Hz}$ e $10 \leq R \leq 40 \Omega$.
- Construa um outro gráfico que seja a projeção sobre o plano xz do gráfico da letra a . Estime, a partir desse gráfico, o valor da frequência natural ou de ressonância do circuito, isto é a frequência para a qual I

CAPÍTULO 10

Aplicações em Cálculo Numérico

Nas ciências e nas engenharias, os métodos numéricos são utilizados frequentemente na solução de problemas matemáticos onde é difícil, ou até mesmo impossível, obter soluções exatas para as equações. O MATLAB é uma ferramenta essencialmente de Cálculo Numérico. Por isso, há uma diversidade de funções para solução numérica de problemas matemáticos. Este capítulo explica como utilizar algumas dessas importantes funções. Deve ser ressaltado que o propósito deste livro é mostrar aos usuários como utilizar o MATLAB e, por isso, a teoria do Cálculo Numérico não será abordada aqui. Entretanto, algumas informações gerais sobre Cálculo Numérico serão fornecidas, mas o estudante pode, e deve, consultar bibliografias específicas sobre o assunto se desejar mais detalhes sobre as técnicas e os métodos numéricos de solução.

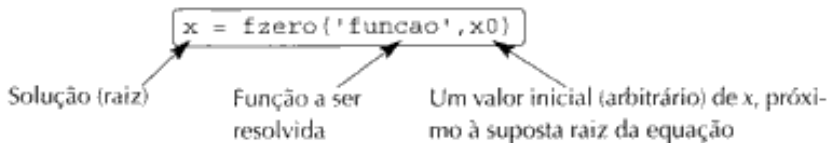
Os seguintes tópicos são abordados neste capítulo: solução de uma equação a uma variável; máximo e mínimo de uma função; integração numérica e resolução de equações diferenciais ordinárias (EDOs) de primeira ordem.

10.1 RESOLVENDO UMA EQUAÇÃO A UMA VARIÁVEL

A equação a uma variável sempre pode ser escrita na forma $f(x) = 0$. A solução dessa equação é o valor de x onde a função cruza (ou toca) o eixo x , isto é, onde a função vale zero. Uma solução ou raiz exata é o valor de x para o qual a função vale exatamente zero. Se a solução exata não existir, ou for difícil de ser determinada, será necessário utilizar algum método numérico, de modo a encontrar o valor de x que mais se aproxima do valor exato. O melhor valor é aquele cujo erro da aproximação torna-se desprezível em relação a algum parâmetro (medida) previamente adotado. Uma boa maneira de se realizar esses cálculos é utilizar algum método iterativo, onde a cada iteração o computador determina um valor para x que mais se aproxima da solução exata. O processo de iteração termina

quando a diferença em x entre as duas últimas iterações tornar-se menor que alguma medida. Em geral, uma função pode ter nenhuma, uma, muitas ou até mesmo um número infinito de soluções.

No MATLAB, o zero de uma função é calculado pela função nativa `fzero`, cuja sintaxe é:



Outros detalhes a respeito dos argumentos de `fzero`:

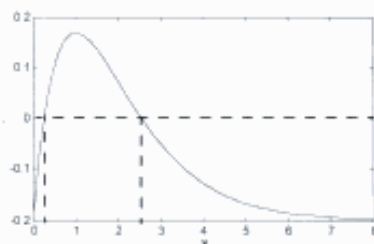
- x é um escalar.
- 'função' é a expressão que se deseja resolver. Existem três maneiras de introduzi-la no argumento:
 1. O modo mais simples é digitá-la como uma expressão matemática na forma de *string*.
 2. É possível que a expressão a ser resolvida envolva uma função personalizada pelo usuário (veja o Capítulo 6). Nesses casos, basta digitar o nome da função entre aspas simples.
 3. A função pode ser criada como uma função *inline* (veja a Seção 6.8) e o nome dela entrar, outra vez, como uma *string*.
- A função deve ser escrita na forma padrão, i.e., $f(x) = 0$. Por exemplo, se a função a ser resolvida for $x e^{-x} = 0.2$, será necessário reescrevê-la na forma $f(x) = x e^{-x} - 0.2 = 0$. Então, para entrar com essa função no comando `fzero` basta digitar: `'x*exp(-x) - 0.2'`.
- Quando uma função é digitada como uma *string*, as variáveis dessa função não devem ser previamente inicializadas. Por exemplo, se a função a ser resolvida for $f(x) = x e^{-x} - 0.2$ não é possível declarar `b = 0.2` e, então, digitar `'x*exp(-x) - b'`.
- x_0 pode ser um escalar ou um vetor de dois elementos. Caso x_0 seja introduzido como um escalar, o valor atribuído deve estar próximo do ponto onde a função cruza o eixo x . Se x_0 for introduzido como um vetor, os dois elementos do vetor devem ser pontos em cada lado da solução esperada, tal que $f(x_0(1))$ tenha sinal diferente de $f(x_0(2))$. Quando a função exibe mais de uma solução, cada solução pode ser determinada separadamente, através do uso sistemático da função `fzero` e os diversos valores a serem digitados para x_0 devem ser escolhidos próximos de cada solução particular.
- A melhor maneira de encontrar, aproximadamente, onde a função possui uma solução é construindo o gráfico da função. Nas engenharias e nas ciências exatas, em geral as aplicações envolvem uma escolha (estimativa) do domínio da solução. Além disso, quando uma função exibe mais de uma solução, somente uma ou poucas soluções costumam ter significado físico.

Problema-exemplo 10-1: Resolvendo uma equação não-linear

Determine a solução da equação $x e^{-x} = 0.2$.

Solução

Inicialmente, é necessário reescrever a equação na forma $f(x) = 0$. Daí, $f(x) = x e^{-x} - 0.2 = 0$. O gráfico da equação no intervalo $0 \leq x \leq 8$ é mostrado ao lado. Perceba que existem duas raízes para a equação: uma no intervalo $0 \leq x \leq 1$ e outra no intervalo $2 \leq x \leq 3$. O gráfico foi obtido pelo comando `fplot` na linha do *prompt* da janela Command Window:



```
>> fplot('x*exp(-x)-0.2',[0 8])
```

As raízes da função são encontradas usando o comando `fzero` duas vezes; uma tomando o valor de `x0` entre 0 e 1 (p. ex.: 0.7) e outra adotando o valor de `x0` entre 2 e 3 (p. ex.: 2.8):

```
>> x1 = fzero('x*exp(-x)-0.2',0.7)
```

```
x1 =
```

```
0.2592
```

A primeira raiz é 0.2592.

```
>> x1 = fzero('x*exp(-x)-0.2',2.8)
```

```
x1 =
```

```
2.5426
```

A segunda raiz é 2.5426.

Comentários adicionais:

- `fzero` determina os zeros de uma função somente onde a função cruza o eixo x . `fzero` não encontra um zero onde a função apenas toca, mas não cruza, o eixo x .
- Caso uma solução não possa ser determinada, NaN é atribuído a x .
- O comando `fzero` possui outras opções (veja o Help Window do comando). As duas opções mais importantes são:

`[x fval] = fzero('funcao', x0)`: atribui o valor da função no ponto x à variável `fval`.

`x = fzero('funcao', x0, optimset('display', 'iter'))`: durante a determinação da solução, mostra cada resultado de saída no processo de iteração.

- Se a função puder ser escrita na forma de um polinômio, as soluções ou raízes podem ser encontradas mediante o comando `roots`, como explicado no Capítulo 8 (Seção 8.1.2).

- O comando `fzero` também pode ser utilizado na determinação do valor de x para um valor específico da função. Para tanto, basta transladar a função para cima ou para baixo, conforme o caso, e executar o comando `fzero`. Por exemplo, na função do Problema Exemplo 10-1, o primeiro valor de x onde a função vale 0.1 é determinado resolvendo-se a equação $x e^{-x} - 0.3 = 0$. Isso é mostrado a seguir:

```
>> x = fzero('x*exp(-x)-0.3',0.5)
x =
    0.4894
```

10.2 ENCONTRANDO O MÁXIMO OU O MÍNIMO DE UMA FUNÇÃO

O problema de determinar o máximo ou mínimo de uma função $y = f(x)$ é essencial em muitas aplicações. No cálculo tradicional, o valor de x é determinado a partir do zero (raiz) da derivada da função. Em seguida, o valor de y é calculado por substituição de x dentro da função original. No MATLAB, o valor de x onde uma função de uma única variável $f(x)$ tem um mínimo no intervalo $x_1 \leq x \leq x_2$ é determinado por meio do comando `fminbnd`:

`x = fminbnd('funcao', x1, x2)`
 Valor onde a função Função para determi- Intervalo de x .
 exibe um mínimo. nar o mínimo.

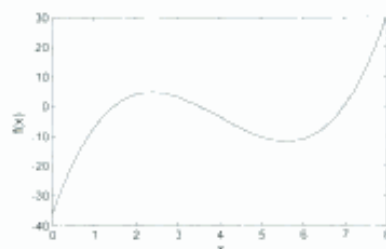
- A função pode ser digitada como uma *string*, como o nome de uma função personalizada ou como uma função *inline*, de maneira semelhante ao comando `fzero`. Veja a Seção 10.1 para maiores detalhes.
- O valor da função no mínimo pode ser passado para uma outra variável:

```
[x fval] = fminbnd('função', x1, x2)
```

Nesse caso, o valor da função no ponto x é passado para `fval`.

- Ao executar o comando `fminbnd`, ele tenta primeiramente encontrar um mínimo local (um ponto onde a inclinação da função seja nula) dentro do intervalo escolhido para x . Se não for encontrado um mínimo local, o comando retornará um dos pontos extremos (x_1 ou x_2), o que for menor.

Por exemplo, vamos considerar a função $f(x) = x^3 - 12x^2 + 40.25x - 36.5$, restrita ao intervalo $0 \leq x \leq 8$ (veja a figura abaixo).



Está claro que o mínimo local situa-se entre 5 e 6. Utilizando o comando `fminbnd` no intervalo mencionado é fácil encontrar a localização exata do mínimo e o valor da função nesse ponto:

```
>> [x fval] = fminbnd('x^3 - 12*x^2 + 40.25*x - 36.5',0,8)
x =
    5.6073
fval =
   -11.8043
```

O mínimo ocorre em $x = 5.6073$. O valor da função nesse ponto é -11.8043 .

Perceba que o comando `fminbnd` retorna o mínimo local e não o mínimo absoluto que, nesse caso, ocorre em $x = 0$. Mudando o intervalo para $0 \leq x \leq 8$, o mínimo local é excluído. Logo:

```
>> [x fval] = fminbnd('x^3 - 12*x^2 + 40.25*x - 36.5',0,8)
x =
    0
fval =
   -36.5000
```

Nesse intervalo, o mínimo absoluto ocorre em $x = 0$. O valor da função nesse ponto é -36.5 .

Nesse intervalo, a função não possui um mínimo local e `fminbnd` retorna o valor da extremidade $x = 0$, cujo valor correspondente da função é o menor.

- O comando `fminbnd` também pode ser utilizado para determinar o máximo de uma função. Basta multiplicar a função por -1 e determinar o mínimo. De fato, esse mínimo representa o máximo da função original. Por exemplo, o máximo da função $f(x) = x e^{-x} - 0.2$ (Problema-exemplo 10-1), no intervalo $0 \leq x \leq 8$, é determinado através do mínimo da função $f(x) = -x e^{-x} + 0.2$.

```
>> [x fval] = fminbnd('-x*exp(-x) + 0.2',0,8)
x =
    1.0000
fval =
   -0.1679
```

O máximo da função (nesse caso global) ocorre em $x = 1.0$. O valor da função nesse ponto é 0.1679 .

10.3 INTEGRAÇÃO NUMÉRICA

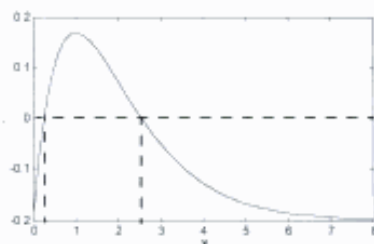
A integração é uma operação matemática indispensável nas ciências exatas e nas engenharias modernas. Calcular área e volume, velocidade a partir da aceleração, trabalho da força e do deslocamento são apenas alguns dos muitos exemplos onde as integrais são necessárias. Em geral, a integração de funções simples pode ser feita analiticamente, mas muitas funções relevantes são, com frequência, difíceis ou impossíveis de serem integradas analiticamente. Nos cursos formais de Cálculo, o integrando é a função a ser submetida à operação de integração. Em algumas aplicações das ciências e das engenharias, o

Problema-exemplo 10-1: Resolvendo uma equação não-linear

Determine a solução da equação $x e^{-x} = 0.2$.

Solução

Inicialmente, é necessário reescrever a equação na forma $f(x) = 0$. Daí, $f(x) = x e^{-x} - 0.2 = 0$. O gráfico da equação no intervalo $0 \leq x \leq 8$ é mostrado ao lado. Perceba que existem duas raízes para a equação: uma no intervalo $0 \leq x \leq 1$ e outra no intervalo $2 \leq x \leq 3$. O gráfico foi obtido pelo comando `fplot` na linha do *prompt* da janela Command Window:



```
>> fplot('x*exp(-x)-0.2',[0 8])
```

As raízes da função são encontradas usando o comando `fzero` duas vezes; uma tomando o valor de `x0` entre 0 e 1 (p. ex.: 0.7) e outra adotando o valor de `x0` entre 2 e 3 (p. ex.: 2.8):

```
>> x1 = fzero('x*exp(-x)-0.2',0.7)
```

```
x1 =
```

```
0.2592
```

A primeira raiz é 0.2592.

```
>> x1 = fzero('x*exp(-x)-0.2',2.8)
```

```
x1 =
```

```
2.5426
```

A segunda raiz é 2.5426.

Comentários adicionais:

- `fzero` determina os zeros de uma função somente onde a função cruza o eixo x . `fzero` não encontra um zero onde a função apenas toca, mas não cruza, o eixo x .
- Caso uma solução não possa ser determinada, NaN é atribuído a x .
- O comando `fzero` possui outras opções (veja o Help Window do comando). As duas opções mais importantes são:

`[x fval] = fzero('funcao', x0)`: atribui o valor da função no ponto x à variável `fval`.

`x = fzero('funcao', x0, optimset('display', 'iter'))`: durante a determinação da solução, mostra cada resultado de saída no processo de iteração.

- Se a função puder ser escrita na forma de um polinômio, as soluções ou raízes podem ser encontradas mediante o comando `roots`, como explicado no Capítulo 8 (Seção 8.1.2).

```
>> quad('x.*exp(-x.^0.8)+0.2',0,8)
ans =
    3.1604
```

No segundo modo, primeiramente é criada uma função personalizada (salva como Capítulo10_Exemplo2) que calcula a função a ser integrada:

```
function y = Capitulo10_Exemplo2 (x)
y = x.*exp(-x.^0.8) + 0.2;
```

Outra vez a função foi escrita utilizando-se operações escalares, de modo que o argumento x passa a ser um vetor. A integração é realizada na linha do *prompt* da janela Command Window, digitando-se o comando `quad` com o nome da função ocupando o campo 'função':

```
>> q = quad('Capitulo10_Exemplo2',0,8)
q =
    3.1604
```

O comando `trapz`:

O comando `trapz` é utilizado para integração de funções que consistem de um conjunto discreto de pontos. Esse comando faz uso do método trapezoidal para a integração numérica. A sintaxe do comando é:

$$q = \text{trapz}(x,y)$$

onde x e y são vetores cujos elementos são as coordenadas x e y dos pontos, respectivamente. Os dois vetores devem ter a mesma dimensão.

10.4 EQUAÇÕES DIFERENCIAIS ORDINÁRIAS (EDOs)*

Equações diferenciais desempenham um papel crucial nas ciências e engenharias, já que formam o alicerce de, virtualmente, todo fenômeno físico que aparece nessas áreas do conhecimento. Entretanto, um número limitado de EDOs pode ser resolvido por métodos analíticos. Conforme já mencionado, os métodos numéricos constituem uma solução alternativa para praticamente qualquer equação. Isso não seria diferente para as EDOs. Contudo, neste caso, obter a solução numérica pode não ser uma tarefa simples. Isso porque um método numérico capaz de resolver qualquer equação ainda não foi desenvolvido e, possivelmente, nunca o será. Existem muitos métodos que resolvem diferentes tipos de equações. O MATLAB possui um repertório enorme de ferramentas para tratar e resolver equações diferenciais. Para utilizar eficientemente o MATLAB para resolver EDOs, o usuário deve possuir conhecimentos sólidos sobre equações diferenciais e os vários métodos numéricos capazes de resolvê-las.

* N. de T. Nos comandos do MATLAB, a sigla é ODE (*ordinary differential equations*).

Esta seção descreve em detalhes como usar o MATLAB para resolver uma EDO de primeira ordem. Os métodos numéricos utilizados na solução de uma equação serão apenas mencionados e descritos em linhas gerais, mas não serão explicados de um ponto de vista matemático. Em suma, o objetivo desta seção é ensinar a resolver EDOs de primeira ordem simples e “não-problemáticas”. A solução para EDOs de primeira ordem é o ponto de partida para EDOs de ordens elevadas e sistemas de EDOs.

Uma Equação Diferencial Ordinária (EDO) é uma equação relacionando uma variável independente, uma variável dependente e derivadas da variável dependente. As EDOs são consideradas de primeira ordem quando:

$$\frac{dy}{dx} = f(x, y)$$

onde x é a variável independente e y é a variável dependente. Uma solução da EDO é uma função $y = f(x)$ que satisfaz a equação. Em geral, as soluções de uma EDO constituem uma família de soluções. Para distinguir as soluções são necessárias condições iniciais para o problema específico (denominado Problema de Valor Inicial – PVI). Para as EDOs de primeira ordem, a condição inicial é tão-simplesmente o valor da função y (variável dependente) em algum ponto x específico.

Etapas de resolução de uma EDO de primeira ordem:

Para o restante desta seção, a variável independente será t (tempo). Isso porque em muitas aplicações o tempo é a variável independente e, além disso, para ficar consistente com o menu **Help** do MATLAB.

1ª etapa: Escreva o problema na forma padrão.

Escreva a EDO na forma:

$$\frac{dy}{dt} = f(t, y) \quad \text{para } t_0 \leq t \leq t_f, \quad \text{com } y = y_0 \quad \text{em } t = t_0$$

Como mostrado acima, são necessários, além da equação propriamente dita, o valor de y para algum instante específico de tempo. A solução da equação é uma função para y , em termos de t , válida no intervalo de tempo entre t_0 e t_f . Um exemplo PVI de primeira ordem é:

$$\frac{dy}{dt} = \frac{t^3 - 2y}{t} \quad \text{para } 1 \leq t \leq 3, \quad \text{com } y = 4.2 \quad \text{em } t = 1$$

2ª etapa: Crie um arquivo para a função personalizada que resolve a EDO.

Crie uma função personalizada que calcule $\frac{dy}{dt}$, dados os valores de t e y . Por exemplo, para a EDO do exemplo anterior:

```
function dydt = ODEexpl(t,y)
dydt = (t^3-2*y)/t;
```



$$p = \frac{nRT}{V}$$

onde $R = 0.08206 \text{ L.atm/mol.K}$ é a constante universal dos gases.

A equação de van der Waals é um modelo para o comportamento de um gás real:

$$\left(P + \frac{n^2 a}{V^2} \right) (V - nb) = nRT$$

onde a e b são constantes específicas de cada gás.

Utilize a função `fzero` para determinar o volume de 2 mol de CO_2 , à temperatura de 50°C e pressão de 6 atm. Para o CO_2 , as constantes a e b são $3.59 \text{ L}^2.\text{atm/mol}^2$ e 0.0427 L/mol , respectivamente.

Solução

A solução do problema está escrita na rotina (salva como `Capitulo10_Exemplo3`) a seguir:

```
global P T n a b R
R=0.08206;
P=6; T=323.2; n=2; a=3.59; b=0.047;
Vest=n*R*T/P
V=fzero('Waals',Vest)
```

Estimando um valor para V .

Este programa calcula inicialmente o valor estimado do volume utilizando a equação do gás ideal. Então, esse valor entra na função `fzero` como estimativa inicial da solução da equação de van der Waals. A equação de van der Waals é escrita como uma função, denominada `Waals`:

```
function fofx=Waals(x)
global P T n a b R
fofx=(P+n^2*a/x^2)*(x-n*b)-n*R*T;
```

Para que as variáveis da rotina e da função fossem compatíveis entre si, foi necessário declará-las variáveis globais. Executando a rotina no *prompt* da janela Command Window, os valores real e estimado de V são mostrados:

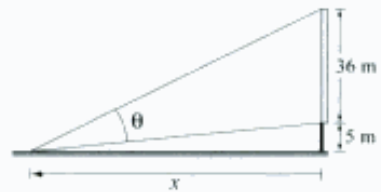
```
>> Capitulo10_Exemplo3
Vest =
8.8406
V =
8.6613
```

Valor estimado pela equação do gás ideal.

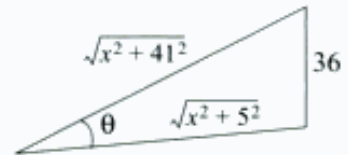
Valor encontrado pela equação de Van der Waals.

Problema-exemplo 10-4: Estabelecendo o ângulo máximo de observação

Um cinéfilo procura sentar-se a uma distância x da tela, de modo a maximizar o ângulo de visão θ . Determine a distância x para o ângulo θ seja máximo. Considere a figura esquemática da sala de projeção.

**Solução**

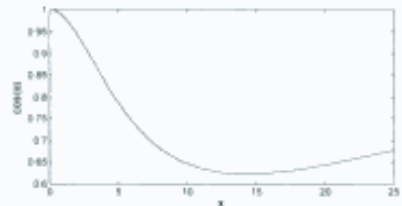
O problema foi resolvido escrevendo-se uma função para o ângulo θ em termos de x . Em seguida, foi determinado o valor de x para o qual o ângulo é máximo. No triângulo onde aparece θ , um dos lados é uma constante (altura da tela) e os outros dois podem ser escritos em função da distância x (observe a figura).



Uma maneira de escrever θ em termos de x é utilizar a lei dos co-senos:

$$\cos(\theta) = \frac{(x^2 + 5^2) + (x^2 + 41^2) - 36^2}{2\sqrt{x^2 + 5^2}\sqrt{x^2 + 41^2}}$$

O ângulo θ esperado deve estar entre 0 e $\pi/2$. Isso porque $\cos(0) = 1$ e o co-seno é uma função decrescente para incrementos em θ no intervalo supracitado. Assim, o ângulo de visão máximo corresponde ao menor $\cos(\theta)$. Foi plotado o gráfico de $\cos(\theta)$ em função de x , o que possibilita a localização do intervalo onde está situado o mínimo de $\cos(\theta)$.



A seguir, vemos o gráfico gerado pelos comandos acima:

```
>> fplot('((x^2 + 5^2)+(x^2 + 41^2) - 36^2)/(2*sqrt(x^2 + 5^2)*sqrt(x^2 + 41^2))',[0 25])
>> xlabel('x'); ylabel('cos(\theta)');
```

O gráfico de $\cos(\theta)$ em função de x mostra que a função tem um mínimo entre 10 e 20. O melhor valor para o mínimo é determinado através do comando `fminbnd`:

```
>> [x cosTeta] = fminbnd('((x^2 + 5^2)+(x^2 + 41^2) - 36^2)/(2*sqrt(x^2 + 5^2)*sqrt(x^2 + 41^2))',10,20)
```

```
x =
```

```
14.3178
```

```
cosTeta =
```

```
0.6225
```

```
>> Teta = cosTeta*180/pi
```

```
Teta =
```

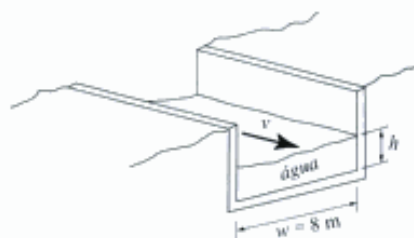
```
35.6674
```

O mínimo ocorre em $x = 14.3178\text{m}$.
Nesse ponto, $\cos(\theta) = 0.6225$.

Ângulo de observação máximo é 35.6674° .

Problema-exemplo 10-5: Vazão em um rio

Para estimar a quantidade de água que flui em um rio durante um ano é feita uma seção retangular no leito do rio (observe a figura). No início de cada mês (começando em 1º de janeiro) são medidas a velocidade, v , e o nível d'água, h . A medida do primeiro dia foi tomada como 1 e a última medida (1º de janeiro do ano seguinte) foi tomada como 366. Ao longo do ano, foram registradas as seguintes medidas:



Dia	1	32	60	91	121	152	182	213	244	274	305	335	366
$h(\text{m})$	2.0	2.1	2.3	2.4	3.0	2.9	2.7	2.6	2.5	2.3	2.2	2.1	2.0
$v(\text{m/s})$	2.0	2.2	2.5	2.7	5	4.7	4.1	3.8	3.7	2.8	2.5	2.3	2.0

Use os dados da tabela para determinar a vazão na seção retangular e, em seguida, integre-a para obter uma estimativa global da quantidade de água que flui no rio durante um ano.

Solução

A vazão Q (volume de água por segundo) para cada ponto é obtida multiplicando-se a velocidade da água pela área da seção retangular (altura $\times$ largura) do canal:

$$Q = vwh \quad (\text{m}^3/\text{s})$$

A quantidade total de água é estimada pela integral:

$$V = (60 \cdot 60 \cdot 24) \int_1^{366} Q dt$$

A vazão foi dada em metros cúbicos por segundo, o que significa que o tempo deve ser medido em segundos. Como os dados foram tabelados em termos do número de dias, a integral foi multiplicada pelo fator de conversão $(60 \cdot 60 \cdot 24)$ s/dia.

A rotina a seguir (salva como `Capitulo10_Exemplo5`) calcula Q e realiza a integração usando o comando `trapz`. O programa também gera um gráfico vazão *versus* tempo.

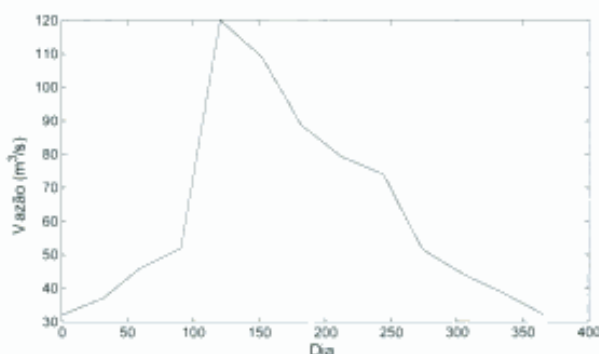
```
w = 8;
d = [1 32 60 91 121 152 182 213 244 274 305 335 366];
h = [2.0 2.1 2.3 2.4 3.0 2.9 2.7 2.6 2.5 2.3 2.2 2.1 2.0];
v = [2.0 2.2 2.5 2.7 5.0 4.7 4.1 3.8 3.7 2.8 2.5 2.3 2.0];
Q = v.*w.*h;
Vol = 60*60*24*trapz(d,Q);
```

```
fprintf('A quantidade estimada de agua que flui no rio durante um ano e %g m^3.',Vol)
plot(d,Q)
xlabel('Dia'); ylabel('Vazao (m^3/s)');
```

Executando a rotina na linha do *prompt* da janela Command Window é exibida uma estimativa da quantidade de água e o gráfico correspondente aos dados da tabela.

```
>> Capitulo10_Exemplo5
```

A quantidade estimada de agua que flui no rio durante um ano e: 2.03095e+009 m³.



Problema-exemplo 10-6: Teste de colisão de um carro contra um pára-choque hidráulico

Um pára-choque hidráulico foi colocado em um laboratório de modo a simular as condições de colisão de um carro em alta velocidade (figura ao lado).



O pára-choque foi projetado de maneira que a força de impacto que o carro recebe durante a colisão seja uma função da velocidade do carro (v) e do deslocamento (x) do eixo hidráulico do pára-choque, de acordo com a equação:

$$F = Kv^3(x+1)^3$$

onde $K = 30 \text{ s.kg/m}^5$ é uma constante.

Um carro com uma massa m de 1500kg e a 90km/h foi submetido ao teste. Determine e plote a velocidade do carro em função da posição para $0 \leq x \leq 3\text{m}$.

Solução

A desaceleração do carro contra o pára-choque é dada pela Segunda Lei de Newton:

$$ma = -Kv^3(x+1)^3$$

a qual pode ser resolvida para expressar a aceleração em função de v e x :

$$a = \frac{-Kv^3(x+1)^3}{m}$$

A derivada da velocidade v em relação a x é explicitada substituindo-se a aceleração na equação:

$$v dv = a dx$$

o que resulta:

$$\frac{dv}{dx} = \frac{-Kv^2(x+1)^3}{m}$$

A última equação é uma EDO de primeira ordem que deve ser resolvida no intervalo $0 \leq x \leq 3$ m e sujeita às condições: $v = 90$ km/h em $x = 0$.

Uma solução numérica da equação diferencial é mostrada a seguir:

```
global k m
k = 30; m = 1500; v0 = 90;
xspan = [0:0.2:3];
v0mps = v0*1000/3600;
[x v] = ode45('parachoque',xspan,v0mps)
plot(x,v)
xlabel('x(m)'); ylabel('velocidade (m/s)')
```

Vetor contendo o intervalo de validade da solução.

Conversão de km/h para m/s.

Resolvendo a EDO.

A função `parachoque` chamada na rotina acima tem o seguinte corpo:

```
function dvdx = parachute(x,v)
global k m
dvdx = -(k*v^2*(x+1)^3)/m;
```

Executando a rotina no *prompt* da janela Command Window, os vetores x e v são exibidos na tela (os dois vetores não aparecem lado a lado, como sugere a caixa abaixo. A rotina, do modo como está, lista um vetor abaixo do outro.)

```
>> Capitulo10_Exemplo6
x =
    0
 0.2000
 0.4000
 0.6000
 0.8000
v =
 25.0000
 22.0420
 18.4478
 14.7561
 11.4302
```

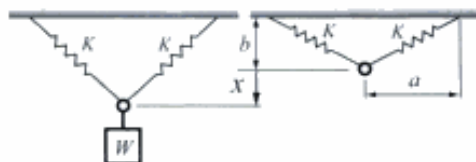
tendem-se e o anel é deslocado para baixo de uma distância x . O peso do objeto (W) pode ser expresso em termos da distância x por:

$$W \frac{2K}{L} (L - L_0)(b + x)$$

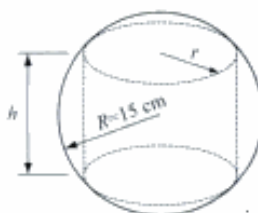
onde $L_0 = \sqrt{a^2 + b^2}$ e

$L = \sqrt{a^2 + (b + x)^2}$ são o comprimento inicial e a elongação da mola, respectivamente.

Dados $a = 0.15\text{m}$, $b = 0.05\text{m}$ e a constante elástica da mola $K = 2800\text{N/m}$, determine a distância x quando um objeto de 250N é pendurado no anel da balança.



6. Considere novamente o bloco do Problema 4. Determine o ângulo θ que minimiza a força para puxar o bloco. Qual é o módulo dessa força?
7. Um copo de papel cilíndrico foi projetado para receber 200ml . Determine o raio e a altura do copo de maneira que se utilize um mínimo de papel para confeccioná-lo.
8. Determine as dimensões (raio r e altura h) do cilindro de maior volume que pode ser inscrito em uma esfera de raio $R = 15\text{cm}$.
9. A lei de radiação de corpo negro de Planck dá a radiância espectral R em função do comprimento de onda λ e da temperatura T (em Kelvin):



$$R = \frac{2\pi c^2 h}{\lambda^5} \frac{1}{e^{(hc)/(\lambda kT)} - 1}$$

onde $c = 3.0 \times 10^8\text{m/s}$ é a velocidade da luz, $h = 6.63 \times 10^{-34}\text{J}\cdot\text{s}$ é a constante de Planck e $k = 1.38 \times 10^{-23}\text{J/K}$ é a constante de Boltzmann.

Plote R em função de λ , para $0.2 \times 10^{-6} \leq \lambda \leq 6.0 \times 10^{-6}\text{m}$ e à temperatura $T = 1500\text{K}$. Em seguida, determine o comprimento de onda que resulta na radiância espectral máxima.

10. Use o MATLAB para calcular a seguinte integral:

$$\int_0^5 \frac{1}{0.8x^2 + 0.5x + 2} dx$$

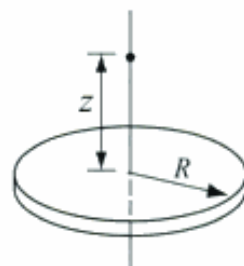
11. Use o MATLAB para calcular a seguinte integral:

$$\int_0^\pi \cos^2(0.5x) \sin^4(0.5x) dx$$

12. O módulo do campo elétrico E , devido à distribuição superficial de cargas no disco, em um ponto situado sobre o eixo z acima da superfície do disco, é dado por:

$$E = \frac{\sigma z}{4\epsilon_0} \int_0^R (z^2 + r^2)^{-3/2} (2r) dr$$

onde σ é a densidade superficial de cargas, $\epsilon_0 = 8.85 \times 10^{-12} \text{C}^2/\text{N}\cdot\text{m}^2$ é a permissividade do vácuo e R é o raio do disco. Determine o campo elétrico em um ponto localizado no eixo z , a 5cm do disco de $R = 6\text{cm}$ carregado com $\sigma = 300\mu\text{C}/\text{m}^2$.



13. A aceleração da gravidade g varia em função da altura y relativa ao nível do mar por:

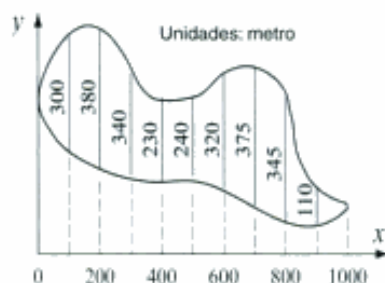
$$g = \frac{R^2}{(R + y)^2} g_0$$

onde $R = 6.371\text{km}$ é o raio da Terra e $g_0 = 9.81\text{m/s}^2$ é a aceleração da gravidade ao nível do mar. A variação na energia potencial gravitacional (ΔU) de um objeto subindo no campo gravitacional da Terra é dada por:

$$\Delta U = \int_0^h mg dy$$

Determine a variação na energia potencial gravitacional de um satélite posicionado 800km acima da superfície da Terra e cuja massa é 500kg.

14. A superfície de um lago é estimada medindo-se a largura do lago em intervalos de 100m. As medidas estão na figura ao lado. Use integração numérica para estimar a área da superfície do lago.



15. Resolva o PVI:

$$\frac{dy}{dx} = \frac{x^2 - 2x + 3}{y^2} \quad \text{para } 0.5 \leq x \leq 3 \quad \text{com } y(0.5) = 2$$

Esboce a solução.

16. Resolva o PVI:

$$\frac{dy}{dx} = 0.2xy + 0.5y^2 \quad \text{para } 0 \leq x \leq 4 \quad \text{com } y(0) = -0.5$$

Esboce a solução.

A equação diferencial que descreve o comportamento da tensão do capacitor no circuito é:

$$\frac{dv_c}{dt} + \frac{1}{RC} v_c = \frac{1}{RC} v_s$$

onde v_c é a tensão do capacitor. Inicialmente, $v_s = 0$ e, em $t = 0$, a tensão da fonte é repentinamente ligada. Determine a resposta do circuito nos três casos a seguir:

- a) $v_s = 12\text{V}$ para $t \geq 0$
- b) $v_s = 12\sin(2.60\pi t)\text{V}$ para $t \geq 0$
- c) $v_s = 12\text{V}$ para $0 \leq t \leq 0.01\text{s}$ e $v_s = 0$ para $t \geq 0.01\text{s}$ (pulso retangular)

Cada caso corresponde a uma equação diferencial diferente. A solução da EDO é a tensão do capacitor em função do tempo. Resolva cada situação para $0 \leq t \leq 0.02\text{s}$. Em seguida, esboce v_s e v_c em função do tempo (construa gráficos separados na mesma janela).

CAPÍTULO 11

Matemática Simbólica

Nos dez capítulos anteriores, todas as operações matemáticas realizadas no MATLAB foram numéricas. As operações eram realizadas escrevendo as expressões numéricas que continham as constantes e variáveis necessárias à solução do problema. Quando uma expressão numérica é executada no MATLAB, o resultado de saída também é numérico (seja um escalar ou vetorial). O número ou os números são escritos na notação de ponto fixo ou ponto flutuante. Por exemplo, digitando $1/4$, o resultado exato é 0.2500 , e, digitando $1/3$, o resultado aproximado é 0.3333 .

Muitas aplicações na matemática, ciências ou engenharias requerem operações simbólicas nas equações. Operações simbólicas são operações matemáticas com expressões contendo variáveis simbólicas, isto é, variáveis que não receberam nenhum valor numérico quando a operação de execução foi disparada no MATLAB. O resultado de tais operações também é uma expressão matemática em termos das variáveis simbólicas. Um exemplo simples é resolver uma equação algébrica a muitas variáveis para explicitar a dependência funcional de uma dessas variáveis. Se a , b e x são as variáveis simbólicas e a expressão algébrica é $ax - b = 0$, x pode ser resolvido em termos de a e b , o que resulta em $x = b/a$. Outros exemplos importantes de operações simbólicas são a integração e diferenciação analítica de expressões matemáticas. Por exemplo, a derivada com relação a t de $2t^3 + 5t - 8$ vale $6t^2 + 5$.

O MATLAB possui a capacidade de realizar muitos tipos de operações simbólicas. A parte numérica das operações simbólicas do MATLAB são realizadas exatamente e não utilizando aproximações, como vimos nos capítulos anteriores. Por exemplo, o resultado da adição de $\frac{x}{4}$ e $\frac{x}{3}$ é $\frac{7}{12}x$ e não $0.5833x$.

As operações simbólicas podem ser realizadas pelo MATLAB quando o Symbolic Math Toolbox está instalado. Essa *toolbox* é uma coleção de funções do MATLAB que é

utilizada para a execução de operações simbólicas. Os comandos e as funções para as operações simbólicas têm o mesmo estilo e sintaxe que aqueles encontrados nas operações numéricas. As operações simbólicas são executadas primeiramente pelo Maple®, *software* projetado especialmente para lidar com variáveis simbólicas. Ele está embutido no MATLAB e é ativado sempre que uma função matemática simbólica é executada no MATLAB. O Maple tem existência separada do MATLAB. Entretanto, possui uma estrutura e comandos completamente diferente do MATLAB. O Symbolic Math Toolbox está incluído na versão do MATLAB para estudantes. Na versão-padrão, ele deve ser adquirido separadamente. Para verificar se essa *toolbox* está instalado no seu computador, você pode digitar o comando `ver` na linha do *prompt* do Command Window. Em resposta, o MATLAB exibe informações sobre a versão que está sendo utilizada, assim como a lista de *toolboxes* instaladas.

O ponto de partida das operações simbólicas são os objetos simbólicos. Os objetos simbólicos são constituídos de variáveis e constantes que, quando utilizados em expressões matemáticas, ensinam o MATLAB a executar a expressão simbolicamente. Frequentemente, o usuário declara (cria) as(os) variáveis simbólicas (objetos) necessárias(os) e as(os) utiliza para criar expressões simbólicas que serão utilizadas subsequentemente nas operações simbólicas. Se necessário, as expressões simbólicas podem ser usadas nas operações numéricas.

A primeira seção deste capítulo descreve como definir objetos simbólicos e como usá-los para criar expressões simbólicas. A segunda seção ensina como modificar a forma de expressões existentes no MATLAB. Uma vez que uma expressão simbólica foi declarada, ela pode ser utilizada nas operações matemáticas. O MATLAB possui uma seleção enorme de funções para esse propósito. As próximas quatro seções (11.3–11.6) descrevem como usar o MATLAB para resolver equações algébricas, calcular derivadas e integrais, e resolver equações diferenciais. A Seção 11.7 trata do modo de exibição gráfica das expressões simbólicas. Como utilizar expressões simbólicas em cálculos numéricos é explicado na Seção 11.8.

11.1 OBJETOS SIMBÓLICOS E EXPRESSÕES SIMBÓLICAS

Um objeto simbólico pode ser uma variável não inicializada, uma constante (número) ou uma expressão constituída de variáveis simbólicas e números. Uma expressão simbólica nada mais é do que uma expressão matemática contendo um ou mais objetos simbólicos. Quando digitada, uma expressão simbólica lembra uma expressão numérica padrão. Contudo, como a expressão contém objetos simbólicos, é executada pelo MATLAB simbolicamente.

11.1.1 Criando objetos simbólicos

Como vimos, objetos simbólicos podem ser variáveis ou números. Podemos criá-los usando comandos `sym` e/ou `syms`. Um único objeto simbólico pode ser criado com o comando `sym`:

```
Nome_objeto = sym('string')
```

onde a *string* é o nome do objeto simbólico que se deseja criar. Uma *string* pode ser:

- Uma única letra ou uma combinação de letras sem espaços. Exemplos: 'a', 'x' ou 'yad'.
- Uma combinação de letras e números começando sempre com uma letra e não tendo espaços. Exemplos: 'xh12' ou 'r2d2'.
- Um número. Exemplos: '15' ou '4'.

Nos primeiros dois casos, o objeto simbólico é uma variável simbólica. Nesse caso é conveniente, mas não necessário, nomear o objeto com o mesmo nome da *string*. Por exemplo, *a*, *bb* e *x* podem ser definidos como variáveis simbólicas por:

```
>> a=sym('a')
a =
a
>> bb=sym('bb')
bb =
bb
>> x=sym('x');
>>
```

Cria um objeto simbólico a e o atribui à variável a.

Os objetos simbólicos são exibidos na tela sem indentação.

A variável simbólica x é criada e ocultada, pois um ponto-e-vírgula foi digitado no final do comando.

O nome dos objetos simbólicos podem diferir dos nomes das variáveis. Por exemplo:

```
>> g=sym('gama')
g =
gama
```

O objeto simbólico é gama e o nome da variável é g.

Conforme mencionado, objetos simbólicos também podem ser números. Os números não precisam ser digitados como *string*. Por exemplo, o comando `sym` é utilizado a seguir para criar objetos simbólicos dos números 5 e 7 e atribuí-los às variáveis *c* e *d*, respectivamente.

```
>> c=sym(5)
c =
5
>> d=sym(7)
d =
7
```

Cria um objeto simbólico do número 5 e o atribui à variável c.

Os objetos simbólicos não aparecem indentados.

Como vimos, quando um objeto simbólico é criado e um ponto-e-vírgula não é digitado no final do comando, o MATLAB exibe o nome da variável objeto e o objeto propriamente dito nas linhas seguintes. Os objetos simbólicos não aparecem indentados como acon-

tece para as variáveis numéricas. A diferença é ilustrada no exemplo abaixo, onde uma variável numérica é declarada.

```
>> e=13
e=
13
```

O número 13 é atribuído à variável numérica e.

A variável numérica é exibida com indentação.

Muitas variáveis simbólicas podem ser criadas com um único comando através do comando `syms`, cuja sintaxe é:

```
syms nome_variavel nome_variavel nome_variavel
```

O comando cria objetos simbólicos e atribui a eles o mesmo nome das variáveis. Por exemplo, as variáveis y , z e d podem todas ser criadas como variáveis simbólicas em um único comando:

```
>> >> syms y z d
>> y
y=
y
```

As variáveis criadas pelo comando `syms` não são exibidas automaticamente. Digitando-se o nome da variável, é exibida a variável numérica criada.

Quando o comando `syms` é executado, as variáveis que ele cria não são exibidas automaticamente, até mesmo se um ponto-e-vírgula não foi digitado no final do comando.

11.1.2 Criando expressões simbólicas

Expressões simbólicas são expressões matemáticas escritas em termos das variáveis simbólicas. Uma vez que as variáveis simbólicas foram criadas, elas podem ser utilizadas para criar expressões simbólicas. A expressão simbólica é um objeto simbólico, cuja exibição não aparece indentada. A forma-padrão para criar expressões simbólicas é:

```
Nome_expressao = Expressao matematica
```

Alguns exemplos:

```
>> syms a b c x y
>> f=a*x^2+b*x+c
f=
a*x^2+b*x+c
```

Declara a , b , c , x e y como variáveis simbólicas.

Cria a expressão simbólica $ax^2 + bx + c$ e o atribui a f .

A expressão simbólica não aparece indentada na tela.

Quando uma expressão simbólica, incluindo operações matemáticas executáveis (adição, subtração, multiplicação e divisão), é executada, o MATLAB realiza as operações da mesma forma que a expressão foi criada. Por exemplo:

```
>> g=2*a/3+4*a/7-6.5*x+x/3+4*5/3-1.5
```

$\frac{2a}{3} + \frac{4a}{7} - 6.5x + \frac{x}{3} + 4\frac{5}{3} - 1.5$ foi executada na linha do *prompt*.

```
g =
```

```
26/21*a-37/6*x+31/6
```

$\frac{26}{21}a - \frac{37}{6}x + \frac{31}{6}$ é exibida.

Observe que todos os cálculos são realizados exatamente, sem nenhuma aproximação numérica. No último exemplo, $\frac{2}{3}a$ e $\frac{4}{7}a$ foram somados pelo MATLAB e resultaram em $\frac{26}{21}a$, e $-6.5x + \frac{x}{3}$ foi somada e resultou em $\frac{37}{6}x$. As operações com os termos que contém somente números na expressão simbólica também foram somadas exatamente, sem aproximações. No mesmo exemplo, $4 \cdot \frac{5}{3} + 1.5$ foi substituído por $\frac{31}{6}$.

A diferença entre os resultados exatos e numéricos é demonstrada no próximo exemplo, onde as mesmas operações matemáticas são executadas, primeiro como variáveis simbólicas, depois como variáveis numéricas.

```
>> a=sym(3); b=sym(5);
```

Declara a e b simbolicamente como 3 e 5.

```
>> e=b/a+sqrt(2)
```

Cria uma expressão que utiliza a e b.

```
e =
```

O valor exato de e é exibido como um objeto simbólico.

```
5/3+2^(1/2)
```

```
>> c=3; d=5;
```

Declara c e d numericamente como 3 e 5.

```
>> f=d/c+sqrt(2)
```

Cria uma expressão que utiliza c e d.

```
f =
```

```
3.0809
```

Um valor aproximado para f é exibido como um número.

Uma expressão pode incluir tanto objetos simbólicos quanto variáveis numéricas. Entretanto, se uma expressão inclui um objeto simbólico, toda a expressão matemática será realizada exatamente. Por exemplo, se c for substituído por a na última expressão, o resultado será exato, como no primeiro exemplo.

```
>> g=d/a+sqrt(2)
g =
5/3+2^(1/2)
```

Fatos adicionais sobre expressões e objetos simbólicos:

- Expressões simbólicas podem incluir variáveis numéricas que foram obtidas da execução de expressões numéricas. Quando essas variáveis são inseridas nas expressões simbólicas, o valor exato delas é utilizado, mesmo se a variável for exibida antes com um valor aproximado. Por exemplo:

```
>> h=10/3
h =
3.3333
>> k=sym(5); m=sym(7);
>> p=k/m+h
p =
85/21
```

h é declarado como 10/3.

Um valor aproximado é exibido para h .

Declara $k = 5$ e $m = 7$ simbolicamente.

h , k e m são utilizados em uma expressão.

Um valor exato de h é utilizado na determinação de p .
Um valor exato de p é exibido.

- O comando `double(s)` pode ser utilizado para converter uma expressão simbólica S , escrita na forma exata, em expressão numérica (o nome *double* originou-se no fato de que esse comando sempre retorna um número com precisão em ponto flutuante duplo). Dois exemplos são mostrados para esclarecer a situação. No primeiro, p do último exemplo é convertido em variável numérica. No segundo, um objeto simbólico é criado e então convertido para a forma numérica.

```
>> pN=double(p)
pN =
4.0476
>> y=sym(10)*cos(5*pi/6)
y =
-5*3^(1/2)
>> yN=double(y)
yN =
-8.6603
```

p é convertido para a forma numérica (atribuída a pN).

Cria uma expressão simbólica y .

O valor exato de y é mostrado.

y é convertido para forma a numérica (atribuída a yN).

- Um objeto simbólico criado em termos das variáveis simbólicas também pode ser escrito diretamente como uma expressão simbólica, sem antes terem sido declaradas as variáveis simbólicas. Por exemplo, $ax^2 + bx + c$, criada anteriormente como um objeto simbólico, pode ser recriada pelo comando `sym` e atribuída à variável `f`.

```
>> f=sym('a*x^2+b*x+c')
f =
a*x^2+b*x+c
```

É importante salientar que, nesse caso, as variáveis a , b , c e x não têm existência isolada como objetos simbólicos independentes. Toda a expressão é tratada como objeto simbólico. Isso significa que é impossível realizar operações matemáticas simbólicas com os elementos individuais de um objeto. Por exemplo, não será possível diferenciar f em relação a x . Isso difere bastante do modo como a expressão quadrática foi criada no primeiro exemplo da Seção 11.1.2, onde as variáveis individuais tinham existência isolada, pois foram declaradas como objetos simbólicos, e foram utilizadas para compor a expressão quadrática.

- Expressões simbólicas declaradas podem ser utilizadas para criar novas expressões simbólicas. Para tanto, basta simplesmente usar o nome da expressão declarada na nova expressão. Por exemplo:

```
>> syms x y
>> SA=x+y, SB=x-y
SA =
x+y
SB =
x-y
>> F=SA^2/SB^3+x^2
F =
(x+y)^2/(x-y)^3+x^2
```

Declara as variáveis simbólicas x e y .

Cria duas expressões simbólicas SA e SB .

$$SA = x + y$$

$$SB = x - y$$

Cria uma expressão simbólica nova F baseada nas expressões de SA e SB .

$$F = (SA^2)/(SB^3) + x^2 = \frac{(x+y)^2}{(x-y)^3} + x^2$$

11.1.3 O comando `findsym` e a variável simbólica padrão (default)

O comando `findsym` é bastante útil para determinarmos quais variáveis simbólicas estão presentes em uma certa expressão simbólica. O formato do comando é:

`findsym(S)`

e

`findsym(S,n)`

O comando `findsym(S)` exibe em ordem alfabética os nomes de todas as variáveis simbólicas (separadas por vírgulas) que estão presentes na expressão S . O comando

`findsym(S, n)` exibe n variáveis simbólicas que estão presentes na expressão S na ordem-padrão. A ordem-padrão começa com x e é seguida das variáveis mais próximas de x , como y e z . Se houver duas letras igualmente próximas a x , prevalece a ordem alfabética, por exemplo, y antes de w e z antes de v . A variável simbólica padrão em uma expressão simbólica é a primeira variável na ordem-padrão. A variável simbólica padrão em uma expressão S pode ser identificada digitando-se `findsym(S, 1)`. Exemplos:

```
>> syms x h w y d t
```

Declara x, h, w, y, d e t como variáveis simbólicas.

```
>> S=h*x^2+d*y^2+t*w^2
```

Cria uma expressão simbólica para S .

```
S =
```

```
h*x^2+d*y^2+t*w^2
```

```
>> findsym(S)
```

Usa o comando `findsym(S)`.

```
ans =
```

As variáveis simbólicas são exibidas em ordem alfabéticas.

```
d, h, t, w, x, y
```

```
>> findsym(S,5)
```

Usa o comando `findsym(S, n)` com $n = 5$.

```
ans =
```

```
x,y,w,t,h
```

5 variáveis simbólicas são exibidas na ordem-padrão.

```
>> findsym(S,1)
```

Usa o comando `findsym(S, n)` com $n = 1$.

```
ans =
```

```
x
```

A variável simbólica padrão é exibida.

```
>>
```

11.2 MODIFICANDO A FORMA DE UMA EXPRESSÃO SIMBÓLICA

Expressões simbólicas são criadas essencialmente de duas formas: pelo usuário ou pelo MATLAB, como resultado de operações simbólicas. As expressões criadas pelo MATLAB podem não se apresentar na forma mais simples ou numa forma muito interessante ao usuário. A forma de uma expressão residente na sessão do MATLAB pode ser modificada agrupando-se os termos de mesma potência, expandindo produtos, evidenciando multiplicadores comuns, utilizando identidades matemáticas ou trigonométricas e muitas outras opções. As subseções a seguir ilustram vários comandos que podem ser utilizados para modificar a forma de uma expressão simbólica existente na sessão de memória do MATLAB.

11.2.1 Os comandos `collect`, `expand` e `factor`

Vejamos como utilizar os comandos `collect`, `expand` e `factor` para realizar operações de modificação em expressões existentes na sessão de trabalho do MATLAB.

O comando collect:

O comando `collect` desenvolve os termos em uma expressão que possuem variáveis com a mesma potência, reagrupando-os no final. Na expressão nova, os termos aparecem ordenados decrescentemente. O comando possui duas formas:

`collect(S)`
`collect(S,nome_variavel)`

onde S é a expressão de interesse. A forma `collect(S)` funciona melhor quando uma expressão possui uma única variável. Se uma expressão tiver mais de uma variável, primeiramente o MATLAB agrupa os termos de uma variável, em seguida de uma segunda variável e assim por diante. O usuário tem a liberdade de especificar a primeira variável usando a forma `collect(S,nome_variavel)`. Exemplos:

```
>> syms x y
>> S=(x^2+x-exp(x))*(x+3)
S =
(x^2+x-exp(x))*(x+3)
>> F=collect(S)
F =
x^3+4*x^2+(-exp(x)+3)*x-3*exp(x)
>> T=(2*x^2+y^2)*(x+y^2+3)
T =
(2*x^2+y^2)*(x+y^2+3)
>> G=collect(T)
G =
2*x^3+(2*y^2+6)*x^2+y^2*x+y^2*(y^2+3)
>> H=collect(T,y)
H =
y^4+(2*x^2+x+3)*y^2+2*x^2*(x+3)
```

Declara x e y como variáveis simbólicas.

Cria a expressão simbólica:
 $(x^2 + x - e^x)(x + 3)$.

Usa o comando `collect`.

O MATLAB retorna a expressão
 $x^3 + 4x^2 + (-e^x + 3)x - 3e^x$.

Cria a expressão simbólica T :
 $(2x^2 + 2y^2)(x + y^2 + 3)$.

Usa o comando `collect`.

O MATLAB retorna a expressão:
 $2x^3 + (2y^2 + 6)x^2 + xy^2 + y^2(y^2 + 3)$.

Usa o comando `collect`.

O MATLAB retorna a expressão:
 $y^4 + (2x^2 + x + 3)y^2 + 2x^2(x + 3)$.

Observe no exemplo acima que quando `collect(T)` é utilizado, a expressão reformatada é escrita em termos das potências reduzidas de x , mas quando o comando `collect(T,y)` é utilizado a expressão reformatada é escrita em termos das potências reduzidas de y .

O comando expand:

O comando `expand` expande expressões de duas maneiras diferentes. Ele desenvolve os produtos dos termos realizando a distributividade em relação à soma e/ou utiliza identidades trigonométricas, exponenciais, logarítmicas para expandir termos correspondentes incluídos no somatório. A forma do comando é:

`expand(S)`

• **O comando `simplify`:**

O comando `simplify` utiliza operações matemáticas (soma, multiplicação, regras de frações, potências, logaritmos, etc.), identidades funcionais e trigonométricas para gerar a forma mais simples de uma expressão. O formato do comando `simplify` é:

`simplify(s)`

S é o nome de uma expressão
já declarada

ou

S pode ser a expressão que
se deseja simplificar.

Dois exemplos:

```
>> syms x y
```

Declara x e y como variáveis simbólicas.

```
>> S=x*(x*(x-8)+10)-5
```

Cria a expressão simbólica:
 $x(x(x-8)+10)-5$ e a atribui à variável S.

```
S =
```

```
x*(x*(x-8)+10)-5
```

```
>> SA=simplify(S)
```

Usa o comando `simplify` para simplificar S.

```
SA =
```

O MATLAB simplifica a expressão para:
 $x^3-8x^2+10x-5$.

```
x^3-8*x^2+10*x-5
```

```
>> simplify((x+y)/(1/x+1/y))
```

Simplifica $(x+y) / \left(\frac{1}{x} + \frac{1}{y}\right)$.

```
ans =
```

```
x*y
```

O MATLAB simplifica a expressão para: xy.

```
>>
```

O comando `simple`:

O comando `simple` determina a forma da expressão com o menor número de caracteres. Em muitos casos, essa forma é também a mais simples. Quando o comando é executado, o MATLAB cria diversas formas da expressão aplicando os comandos `collect`, `expand`, `factor`, `simplify` e outras funções simplificadoras não mencionadas neste livro. Então, o MATLAB retorna a expressão com a forma mais simples. O comando `simple` possui três formas básicas:

`F = simple(S)`

A forma mais
simples de S é
atribuída a F.

`simple(S)`

Todas formas de simplificação
são tentadas. A resposta é atri-
buída a ans.

`[F how]=simple(S)`

A forma mais simples de S é
atribuída a F. O nome do mé-
todo de simplificação é atri-
buído a how.

A diferença entre as três formas do comando `simple` é a saída. O uso de duas das formas é mostrado a seguir:

```
>> syms x
>> S=(x^3-4*x^2+16*x)/(x^3+64)
S =
(x^3-4*x^2+16*x)/(x^3+64)
>> F=simple(S)
F =
x/(x+4)
>> [G how]=simple(S)
G =
x/(x+4)
how =
factor
```

Declara x como uma variável simbólica.

Cria a expressão simbólica: $\frac{x^3 - 4x^2 + 16x}{x^3 + 64}$ e a atribui a S .

Usa o comando `F=simple(S)` para simplificar S .

A forma mais simples de S , $x/(x+4)$ é atribuída a F .

Usa o comando `[G how]=simple(S)`.

A forma mais simples de S , $x/(x+4)$ é atribuída a G .

A palavra *factor* é atribuído a G . Isso significa que a forma mais simples foi obtida pelo método do comando `factor`.

O uso da forma `simple(S)` não foi demonstrado porque a saída geralmente é muito extensa para ser colocada em uma página de livro. O MATLAB exibe dez métodos de tentativa diferentes e atribui a forma mais simples à variável padrão `ans`. Recomenda-se ao leitor testar o comando e examinar os resultados de saída.

11.2.3 O comando `pretty`:

O comando `pretty` exibe uma expressão simbólica em um formato que lembra a maneira como as expressões matemáticas são escritas pelas pessoas. O comando possui a forma:

`pretty(S)`

Por exemplo:

```
>> syms a b c x
>> S=sqrt(a*x^2+b*x+c)
S =
(a*x^2+b*x+c)^(1/2)
>> pretty(S)
```

Declara a , b , c e x como variáveis simbólicas.

Cria a expressão simbólica: $\sqrt{ax^2 + bx + c}$ e a atribui a S .

O comando `pretty` mostra a expressão em um formato matemático.

$$\begin{array}{c} 2 \qquad 1/2 \\ (a x + b x + c) \end{array}$$

11.3 • RESOLVENDO EQUAÇÕES ALGÉBRICAS

Tanto uma equação algébrica a uma variável quanto um sistema de equações a muitas variáveis podem ser resolvidos através da função `solve`.

Resolvendo uma equação a uma variável:

Uma equação algébrica pode ser composta de uma ou mais variáveis simbólicas. Se a equação tiver apenas uma variável, a solução é numérica. Se a equação tiver mais de uma variável simbólica, uma solução pode relacionar qualquer variável com as demais. A solução de equações algébricas pode ser obtida usando o comando `solve` em uma das duas formas possíveis:

`h=solve(eq)` ou `h=solve(eq,var)`

- O argumento `eq` pode ser o nome de uma expressão simbólica declarada previamente ou uma expressão digitada literalmente. Quando a expressão já existir na sessão do MATLAB, basta colocar `S` no lugar de `eq`. Quando uma expressão não tiver o sinal de igualdade (`=`), o MATLAB resolve a equação $eq = 0$.
- Um equação da forma $f(x) = g(x)$ pode ser resolvida digitando-se a equação (incluindo o sinal `=`) como uma *string* no lugar de `eq`.
- Se a equação a ser resolvida tiver mais de uma variável simbólica, o comando `solve(eq)` resolve a equação para a variável simbólica padrão (Seção 11.1.3). Uma solução para qualquer variável pode ser obtida por meio do comando `solve(eq,var)` digitando-se o nome da variável de interesse no campo `var`.
- Se o usuário digitar: `solve(eq)`, como sempre a solução é atribuída à variável padrão `ans`.
- Se a equação tiver mais de uma solução, a saída `h` será um vetor coluna simbólico com uma solução para cada elemento. Os elementos do vetor são objetos simbólicos. Quando um arranjo de objetos simbólicos é exibido, cada linha é delimitada por colchetes (veja o próximo exemplo).

Os exemplos a seguir ilustram a forma de uso do comando `solve`.

```
>> syms a b x y z
```

Declara `a`, `b`, `x`, `y` e `z` como variáveis simbólicas.

```
>> h=solve(exp(2*z)-5)
```

Usa o comando `solve` para resolver: $e^{2z} - 5 = 0$.

```
h =
```

```
1/2*log(5)
```

```
>> S=x^2-x-6
```

Cria a expressão simbólica: $x^2 - x - 6$ e atribui a `S`.

```
S =
```

```
x^2-x-6
```

```
>> k=solve(S)
```

Usa o comando `solve` para resolver: $x^2 - x - 6 = 0$.

```
k =
```

```
[-2]
```

```
[ 3]
```

```
>> solve('cos(2*y)+3*sin(y)=2')
```

```
ans =
```

```
[ 1/2*pi]
```

```
[ 1/6*pi]
```

```
[ 5/6*pi]
```

```
>> T=a*x^2+5*b*x+20
```

```
T =
```

```
a*x^2+5*b*x+20
```

```
>> solve(T)
```

```
ans =
```

```
[ 1/2/a*(-5*b+(25*b^2-80*a)^(1/2))]
```

```
[ 1/2/a*(-5*b-(25*b^2-80*a)^(1/2))]
```

```
>> M=solve(T,a)
```

```
M =
```

```
-5*(b*x+4)/x^2
```

A equação possui duas raízes. Elas são atribuídas a *k*, o qual é vetor coluna com objetos simbólicos.

Usa o comando `solve` para resolver a equação: $\cos(2y) + 3 \sin(y) = 2$. A equação foi digitada como uma *string* no argumento do comando.

A solução é atribuída à variável *ans*.

Cria a expressão simbólica $ax^2 + 5bx + 20$ e a atribui à variável *T*.

Usa o comando `solve(S)` para resolver $T = 0$.

A equação $T = 0$ é resolvida para a variável *x*, que é a variável simbólica padrão.

Usa o comando `solve(eq, var)` para resolver $T = 0$.

A equação $T = 0$ é resolvida para a variável *a*.

- É possível também usar o comando `solve` para digitar a equação a ser resolvida como uma *string* sem ter que declarar as variáveis simbólicas antecipadamente. Contudo, se a solução contiver variáveis (quando a equação original tiver mais de uma variável simbólica), as variáveis não terão existência como objetos simbólicos independentes. Por exemplo:

```
>> ts= solve('4*t^2+h^2+20*t-5*g')
```

A expressão $4th^2 + 20t - 5g$ foi digitada no comando `solve`.

```
ts =
```

```
5/4*g/(h^2+5)
```

As variáveis *t*, *h* e *g* não foram criadas como variáveis simbólicas antes de a expressão ser digitada no comando `solve`.

O MATLAB resolve a equação $4th^2 + 20t - 5g$ para *t*.

A equação também pode ser resolvida para outra variável diferente de *t*. Por exemplo, uma solução para *g* é obtida por:

```
>> gs= solve('4*t^2+h^2+20*t-5*g','g')
```

```
gs =
```

```
4/5*t^2+4*t
```


Resolvendo um sistema de equações:

O comando `solve` também resolve um sistema de equações. Se o número de equações e variáveis for o mesmo, a solução será numérica. Se o número de variáveis for maior do que o número de equações, a solução será simbólica para as variáveis desejadas em função das demais. Um sistema de equações (dependendo do tipo de equações) pode admitir uma ou mais soluções. Se o sistema tiver apenas uma solução, cada uma das variáveis do sistema é resolvida e recebe um valor ou uma expressão numérica. Se o sistema tiver mais de uma solução, cada uma das variáveis pode assumir diversos valores.

O formato do comando `solve` para resolver um sistema de n equações é:

```
output = solve (eq1, eq2,..., eqn)
```

ou

```
output = solve (eq1, eq2,..., eqn,var1, var2,...,varn)
```

- Os argumentos `eq1, eq2,..., eqn` são as equações a serem resolvidas. Cada argumento pode receber o nome de uma expressão simbólica previamente declarada ou então receber uma expressão digitada como *string*. Quando uma expressão simbólica previamente declara é utilizada, a equação a ser resolvida é $S = 0$. Quando digitamos uma *string* que não contém o sinal ($=$), a equação a ser resolvida é *expressão* = 0. Uma equação que contiver o sinal de igualdade deverá ser digitada como uma *string*.
- No primeiro formato, se o número de equações n é igual ao número de variáveis nas equações, o MATLAB apresenta uma solução numérica para todas as variáveis. Se o número de variáveis supera o número de equações, o MATLAB apresenta uma solução para n variáveis em termos do restante das variáveis. As variáveis para os quais as soluções são obtidas são escolhidas pelo MATLAB de acordo com a ordem simbólica padrão (Seção 11.1.3).
- Quando número de variáveis é maior do que o número de equações, o usuário pode escolher para que variáveis o sistema é resolvido. Isso é feito usando-se o segundo formato do comando `solve` e digitando-se os nomes das variáveis nos argumentos `var1, var2,...,varn`.

A saída (`output`) do comando `solve`, que é a solução do sistema de equações, pode assumir duas formas diferentes. Uma é um arranjo célula e a outra é uma estrutura. Um arranjo célula é um arranjo no qual cada um dos elementos pode ser outro arranjo. Uma estrutura é um arranjo no qual os elementos (denominados campos) são endereçados pelos designadores de texto do campo. Os campos de uma estrutura podem ser arranjos de diferentes tamanhos e tipos. Arranjos células e estruturas não são apresentados em detalhes neste livro, mas uma breve explanação é feita a seguir, a fim de que o leitor seja capaz de usá-los com o comando `solve`.

Quando um arranjo célula é utilizado na saída do comando `solve`, o comando assume a seguinte forma (no caso de um sistema a três equações):

```
[varA, varB, varC] = solve (eq1, eq2, eq3)
```



```
>> syms x y R
```

As duas equações são digitadas como argumentos do comando `solve`.

```
>> [xc, yc]=solve('(x-2)^2+(y-4)^2=R^2','y=x/2+1')
```

A saída (*output*) é um arranjo célula.

```
xc =
```

```
[ 14/5+2/5*(-16+5*R^2)^(1/2)]
```

```
[ 14/5-2/5*(-16+5*R^2)^(1/2)]
```

```
yc =
```

```
[ 12/5+1/5*(-16+5*R^2)^(1/2)]
```

```
[ 12/5-1/5*(-16+5*R^2)^(1/2)]
```

A saída (*output*) está disposta em um arranjo célula com duas células, `xc` e `yc`. Cada célula contém um par de soluções em um vetor coluna simbólico.

A segunda solução utiliza uma estrutura como saída (*output*):

```
>> COORD = solve('(x-2)^2+(y-4)^2=R^2','y=x/2+1')
```

Saída (*output*) disposta em uma estrutura.

```
COORD =
```

```
  x: [2x1 sym]
```

```
  y: [2x1 sym]
```

A saída (*output*) da estrutura `COORD` possui dois campos, `x` e `y`. Cada campo é um vetor simbólico 2 por 1.

```
>> COORD.x
```

Digite o endereço do campo `x`.

```
ans =
```

```
[ 14/5+2/5*(-16+5*R^2)^(1/2)]
```

```
[ 14/5-2/5*(-16+5*R^2)^(1/2)]
```

O conteúdo do campo (a solução de `x`) é exibido.

```
>> COORD.y
```

Digite o endereço do campo `y`.

```
ans =
```

```
[ 12/5+1/5*(-16+5*R^2)^(1/2)]
```

```
[ 12/5-1/5*(-16+5*R^2)^(1/2)]
```

O conteúdo do campo (a solução de `y`) é exibido.

11.4 DIFERENCIAÇÃO

A diferenciação simbólica é realizada pelo comando `diff`. As formas desse comando são:

`diff(S)`

ou

`diff(S, var)`

- `S` pode ser uma expressão simbólica previamente declarada ou uma expressão digitada.
- No comando `diff(S)`, se a expressão possui apenas uma variável simbólica, a diferenciação é tomada em relação a essa variável. Se a expressão possui mais de

as variáveis simbólicas da expressão diferenciada não têm existência isolada, independente da expressão.

11.5 INTEGRAÇÃO

A integração simbólica é realizada através do comando `int`. O comando pode ser utilizado para determinar integrais indefinidas (antiderivadas) e integrais definidas. Para a integração indefinida, as formas dos comandos são:

`int(S)` ou `int(S, var)`

- S pode ser uma expressão simbólica previamente declarada ou uma expressão digitada.
- No comando `int(S)`, se a expressão possui apenas uma variável simbólica, a integração é tomada em relação a essa variável. Se a expressão possui mais de uma variável, a integração é realizada em relação à variável simbólica padrão (Seção 11.1.3).
- No comando `diff(S, var)`, o qual é utilizado para realizar a integração de expressões simbólicas a muitas variáveis, a integração é calculada em relação à variável `var`.

Alguns exemplos*:

```
>> syms x y t
>> S=2*cos(x)-6*x;
>> int(S)
ans =
2*sin(x)-3*x^2
>> int(x*sin(x))
ans =
sin(x)-x*cos(x)
>> R=5*y^2*cos(4*t);
>> int(R)
ans =
5/3*y^3*cos(4*t)
>> int(R,t)
ans =
5/4*y^2*sin(4*t)
```

Declara as variáveis x , y e t como variáveis simbólicas.

Atribui a S a expressão: $2 \cos(x) - 6x$.

Usa o comando `int(S)` para integrar S .

A resposta exibida é: $2 \sin(x) - 3x^2$.

Usa o comando `int(S)` para determinar a integral de $x \sin(x)$.

A resposta exibida é: $\sin(x) - x \cos(x)$.

Atribui a R a expressão: $5y^2 \cos(4t)$.

Usa o comando `int(R)` para integrar R .

O MATLAB integra R em relação a y (variável simbólica padrão).

Usa o comando `int(R, t)` para integrar R em relação a t .

A resposta exibida é: $5y^2 \sin(4t)$.

* N. de T.: Perceba, nos exemplos sobre integração indefinida, que os resultados do MATLAB não levam a constante de integração (c) em consideração. De fato, $c = 0$ no MATLAB.

poder exibir uma dependência explícita na variável independente. A forma geral de uma EDO de segunda ordem é:

$$\frac{d^2y}{dt^2} = f\left(t, y, \frac{dy}{dt}\right)$$

A solução de uma EDO é uma função $y = f(t)$ que satisfaz a equação. A solução de uma EDO pode ser geral ou particular. A solução geral possui constantes que produzem uma família de soluções para a mesma EDO. Em uma solução particular, as constantes são determinadas com base no problema de valor inicial (PVI) ou nas condições de contorno da equação.

O comando `dsolve` pode ser utilizado para obter-se uma solução geral ou, quando as condições iniciais ou de contorno forem especificadas, também obter-se uma solução particular.

Solução geral:

Para obtermos a solução geral de uma EDO, o comando `dsolve` deve ser utilizado em uma das formas:

`dsolve ('eq')`

ou

`dsolve ('eq', 'var')`

- O argumento `eq` é a equação a ser resolvida. Ele deve ser digitado como uma *string* (até mesmo se as variáveis forem objetos simbólicos).
- As variáveis da equação não precisam ser declaradas como objetos simbólicos (Assim, na solução da EDO as variáveis não serão objetos simbólicos).
- Qualquer letra (maiúscula ou minúscula), exceto D, pode ser utilizada para representar a variável dependente.
- No comando `dsolve ('eq')`, a variável independente é tomada por *default* (padrão) com sendo a variável `t`.
- No comando `dsolve ('eq', 'var')`, o usuário define a variável independente digitando-a no argumento `var` como uma *string*.
- Quando a equação estiver sendo digitada, a letra D denota diferenciação, lembre disso. Se y é a variável dependente e t é a variável independente, Dy representa $\frac{dy}{dt}$.

Por exemplo: a equação diferencial $\frac{dy}{dt} + 3y = 100$

digitada como *string* seria: `'Dy + 3*y = 100'`.

- A segunda derivada deve ser digitada como D2, a terceira como D3 e assim por diante. Por exemplo, a equação: $\frac{d^2y}{dt^2} + 3\frac{dy}{dt} + 5y = \sin(t)$ digitada como *string* seria: `'D2y+3*Dy + 5*y = sin(t)'`.
- As variáveis da equação EDO digitada no argumento do comando `dsolve` não precisam ser declaradas com antecedência.
- Na solução, o MATLAB usa C1, C2, C3, e assim por diante, para referenciar as constantes de integração.

Por exemplo, a solução geral da EDO de primeira ordem:

$$\frac{dy}{dt} = 4t + 2y$$

pode ser obtida por:

```
>> dsolve('Dy=4*t+2*y')
ans =
-2*t-1+exp(2*t)*C1
```

A resposta exibida é $y = -2t - 1 + C_1 e^{2t}$

A solução geral da EDO de segunda ordem:

$$\frac{d^2x}{dt^2} + 2 \frac{dx}{dt} + x = 0$$

é obtida por:

```
>> dsolve('D2x+2*Dx+x=0')
ans =
C1*exp(-t)+C2*exp(-t)*t
```

A resposta exibida é $x = C_1 e^{-t} + C_2 t e^{-t}$

Os exemplos a seguir ilustram a solução de EDOs contendo variáveis simbólicas em adição às variáveis independentes e dependentes.

```
>> dsolve('Ds=a*x^2')
ans =
a*x^2*t+C1
```

A variável independente é t (default). O MATLAB resolve a equação: $\frac{ds}{dt} = ax^2$.

A solução exibida é: $s = ax^2 t + C_1$.

```
>> dsolve('Ds=a*x^2','x')
ans =
1/3*a*x^3+C1
```

A variável definida independente é x . O MATLAB resolve a equação: $\frac{ds}{dx} = ax^2$.

A solução exibida é: $s = \frac{1}{3} ax^3 + C_1$.

```
>> dsolve('Ds=a*x^2','a')
ans =
1/2*a^2*x^2+C1
>>
```

A variável definida independente é a . O MATLAB resolve a equação: $\frac{ds}{da} = ax^2$.

A solução exibida é: $s = \frac{1}{2} a^2 x^2 + C_1$.

```
>> dsolve('D2y-2*Dy+2*y=0','y(0)=1','Dy(0)=0')
```

```
ans =
```

```
-exp(t)*sin(t)+exp(t)*cos(t)
```

```
>> factor(ans)
```

```
ans =
```

```
exp(t)*(-sin(t)+cos(t))
```

A solução exibida é: $y = -e^t \sin(t) + e^t \cos(t)$.

A resposta pode ser simplificada por meio do comando `factor`.

A solução simplificada é: $y = -e^t (\sin(t) - \cos(t))$.

Outros exemplos de solução de EDOs são mostrados no Problema-exemplo 11-5.

Se o MATLAB não puder determinar uma solução, ele retornará um objeto simbólico inválido e uma mensagem: `Warning: explicit solution could not be found`.

11.7 PLOTANDO EXPRESSÕES SIMBÓLICAS

Muitas vezes desejamos esboçar uma expressão simbólica. No MATLAB, isso pode ser feito muito facilmente por meio do comando `ezplot`. Para uma expressão simbólica S que utiliza uma certa variável var , o MATLAB trata o problema como uma função $S(var)$ e o comando `ezplot` cria um gráfico de S em função de var . Para uma expressão simbólica portando duas variáveis simbólicas, $var1$ e $var2$, o MATLAB considera uma função a duas variáveis na forma $S(var1, var2) = 0$ e o comando `ezplot` gera um gráfico de uma variável *versus* a outra.

Para esboçar uma expressão S que utiliza uma ou duas variáveis, a forma do comando `ezplot` é:



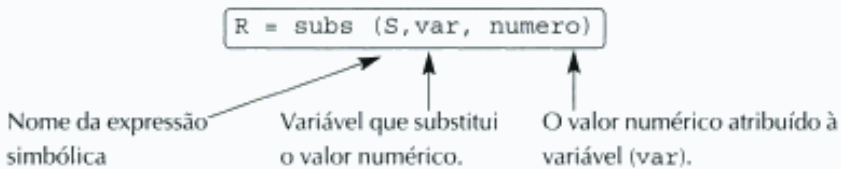
- S pode ser uma expressão simbólica previamente declarada ou uma expressão digitada no argumento.
- É possível digitar diretamente a expressão a ser plotada como uma *string* no argumento do comando, sem mesmo ter que declarar as variáveis simbólicas como objetos simbólicos.
- Se a expressão S possuir, entre as variáveis dela, uma variável simbólica, é criado um gráfico $S(var)$ *versus* (var) , sendo que os valores de var (variável independente) são dispostos na abscissa (eixo horizontal), enquanto que os valores de $S(var)$ são colocados na ordenadas (eixo vertical).
- Se a expressão simbólica S utilizar duas variáveis simbólicas, $var1$ e $var2$, a expressão é tratada como uma função na forma $S(var1, var2) = 0$. Nesse caso, o MATLAB cria um gráfico de uma variável *versus* a outra. A menor variável na ordem

tas formas fáceis de serem utilizadas na maioria das aplicações. Em uma forma, a variável (ou variáveis) é inicializada com um valor numérico em um comando separado e, então, essa variável é substituída na expressão. Em outra forma, a variável para a qual um valor numérico é substituído e os valores numéricos são digitados dentro do comando `subs`.

O comando `subs` será apresentado em dois casos: no primeiro, faremos a substituição de um valor numérico por uma variável simbólica, e no outro substituiremos valores numéricos por duas ou mais variáveis simbólicas.

Substituindo um valor numérico por uma variável simbólica:

Um valor numérico pode ser substituído por uma variável simbólica quando uma expressão simbólica possui uma ou mais variáveis simbólicas. Nesse caso, o comando `subs` possui a forma:



- O número pode ser um escalar ou um arranjo com muitos elementos (vetor ou matriz).
- O valor de S é calculado para cada valor do número e o resultado é passado à variável R , a qual terá a mesma dimensão do número (escalar, vetor ou matriz).
- Se S tiver apenas uma variável simbólica, a saída R será um número. Se S tiver muitas variáveis simbólicas e o valor numérico for substituído em apenas uma delas, a saída R ainda será uma expressão simbólica.

Um exemplo de expressão que inclui uma variável simbólica é:

```
>> syms x
```

Declara x como uma variável simbólica.

```
>> S=0.8*x^3+4*exp(0.5*x)
```

Atribui a S a expressão: $0.8x^3 + 4e^{0.5x}$

```
S =
```

```
4/5*x^3+4*exp(1/2*x)
```

```
>> SD=diff(S)
```

Usa o comando `diff(S)` para diferenciar S .

```
SD =
```

SD recebe a resposta: $12x^2/5 + 2e^{0.5x}$

```
12/5*x^2+2*exp(1/2*x)
```

```
>> subs(SD,x,2)
```

Usa o comando `subs` para substituir $x = 2$ em SD .

```
ans =
```

O valor de SD é exibido.

```
15.0366
```

```
>> SDU=subs(SD,x,[2:0.5:4])
```

Usa o comando `subs` para substituir o vetor $x = [2, 2.5, 3, 3.5, 4]$ em SD .

```
SDU =
```

```
15.0366 21.9807 30.5634 40.9092 53.1781
```

Os valores de SD (passados a SDU) para cada valor de x são mostrados no vetor.

- Se os números substituídos estão organizados em um arranjo, as operações matemáticas realizadas são executadas elemento por elemento e o resultado é um arranjo de números ou expressões. Deve ser enfatizado que os cálculos são realizados elemento por elemento até mesmo se a expressão de S não for digitada na notação elemento por elemento. Isso significa também que todos os arranjos substituídos por variáveis diferentes devem ser do mesmo tamanho.
- É possível substituir arranjos (do mesmo tamanho) por algumas variáveis e escalares por outras variáveis. Nesse caso, de modo a realizar as operações elemento por elemento, o MATLAB utiliza os escalares para produzir um arranjo resultante.

A substituição de valores numéricos para duas ou mais variáveis é demonstrada nos próximos exemplos.

```
>> syms a b c e x
```

Declara a, b, c, e, x como variáveis simbólicas.

```
>> S=a*x^e+b*x+c
```

Cria a expressão simbólica: $ax^e + bx + c$ e a atribui a S .

```
S =
```

```
a*x^e+b*x+c
```

Substitui variáveis simbólicas em S por escalares.

```
>> subs(S,{a,b,c,e,x},{5,4,-20,2,3})
```

Arranjo célula

Arranjo célula

O valor de S é mostrado.

```
ans =
```

```
37
```

Substitui variáveis simbólicas em S por escalares.

```
>> T=subs(S,{a,b,c},{6,5,7})
```

```
T =
```

```
6*x^e+5*x+7
```

O resultado é uma expressão com as variáveis x e e .

```
>> R=subs(S,{b,c,e},{[2 4 6], 9, [1 3 5]})
```

Substitui variáveis simbólicas em S por um escalar para c , e vetores para b e e .

```
R =
```

```
[ a*x+2*x+9, a*x^3+4*x+9, a*x^5+6*x+9]
```

O resultado é um vetor de expressões simbólicas.

```
>> W=subs(S,{a,b,c,e,x},{[4 2 0], [2 4 6], [2 2 2], [1 3 5], [3 2 1]})
```

Substitui S vetores em todas as variáveis simbólicas.

```
W =
```

```
20 26 8
```

O resultado é um vetor de valores numéricos.

Um segundo método para substituir valores numéricos por variáveis simbólicas em uma expressão é primeiramente atribuir valores às variáveis e então usar o comando `subs`. Nesse método, após a declaração da expressão (nesse ponto, as variáveis na expressão são simbólicas) as variáveis são passadas aos valores numéricos. Assim, o comando `subs` adquire a forma:

Solução

- a) O movimento do projétil pode ser analisado considerando-se as componentes horizontal e vertical do projétil. A velocidade inicial pode ser decomposta nas componentes vertical e horizontal:

$$v_{0x} = v_0 \cos(\theta) \quad \text{e} \quad v_{0y} = v_0 \sin(\theta)$$

Na direção horizontal, a velocidade é constante e a posição do projétil em função do tempo é dada por:

$$x = v_{0x} t$$

Substituindo-se $x = 2600\text{m}$ pela distância horizontal que o projétil viaja até atingir o alvo e v_{0x} por $210 \cos(\theta)$, obtemos a equação para t :

$$t = \frac{2600}{210 \cos(\theta)}$$

Na direção vertical, a posição do projétil é dada por:

$$y = v_{0y} t - \frac{1}{2} g t^2$$

Substituindo-se $y = 350\text{m}$ para coordenada vertical do alvo, $210 \sin(\theta)$ por v_{0y} , $g = 9.81$ e t resulta:

$$350 = 210 \sin(\theta) \frac{2600}{210 \cos(\theta)} - \frac{1}{2} 9.81 \left(\frac{2600}{210 \cos(\theta)} \right)^2$$

$$350 = 260 \tan(\theta) - \frac{1}{2} 9.81 \left(\frac{2600}{210 \cos(\theta)} \right)^2$$

A solução dessa equação é o valor do ângulo de elevação de onde o projétil deve ser lançado.

- b) A solução da equação do item a é obtida usando o comando `solve` (na janela Command Window) é:

```
>> syms teta
>> Angulo=solve('2600*tan(teta)-0.5*9.81*(2600/(210*cos(teta)))^2=350')
Angulo =
[-2.6823398465577220256847788629067]
[-1.8962381563523770701488298026235]
[1.2453544972374161683138135806560]
[.45925280703207121277786452037279]
>> Angulo1=Angulo(3)*180/pi
Angulo1 =
224.16380950273491029648644451808/pi
```

O MATLAB exibe quatro soluções para a equação. As duas soluções positivas são relevantes, pois correspondem à situação física apresentada.

Convertendo a solução do terceiro valor de ângulo de radianos para graus.

O MATLAB exibe a resposta simbolicamente em função de π .

```
>> Angulo1=double(Angulo1)
```

```
Angulo1 =
```

```
71.3536
```

```
>> Angulo2=Angulo(4)*180/pi
```

```
Angulo2 =
```

```
82.665505265772818300015613667102/pi
```

```
>> Angulo2=double(Angulo2)
```

```
Angulo2 =
```

```
26.3132
```

```
>>
```

Usa o comando `double` para obter o valor numérico do `Angulo1`.

Convertendo a solução do quarto valor de `Angulo` de radianos para graus.

O MATLAB exibe a resposta simbolicamente em função de π .

Usa o comando `double` para obter o valor numérico do `Angulo2`.

- c) solução da parte *b* mostra que existem dois possíveis ângulos de lançamento e, portanto, de trajetórias. De modo a esboçar as trajetórias do projétil, as coordenadas x e y do projétil são reescritas em termos de t (forma paramétrica):

$$x = v_0 \cos(\theta)t \quad \text{e} \quad y = v_0 \sin(\theta)t - \frac{1}{2}gt^2$$

O domínio para t estende-se de $t = 0$ até $t = \frac{2600}{210 \cos(\theta)}$.

Essas equações podem ser utilizadas no comando `ezplot` para escrevermos uma rotina que trace as trajetórias solicitadas.

```
xmax=2600; v0=210; g=9.81;
```

```
teta1=1.24535; teta2=0.45925;
```

```
t1=xmax/(v0*cos(teta1));
```

```
t2=xmax/(v0*cos(teta2));
```

```
syms t
```

```
X1=v0*cos(teta1)*t;
```

```
X2=v0*cos(teta2)*t;
```

```
Y1=v0*sin(teta1)*t-0.5*g*t^2;
```

```
Y2=v0*sin(teta2)*t-0.5*g*t^2;
```

```
ezplot(X1,Y1,[0,t1])
```

```
hold on
```

```
ezplot(X2,Y2,[0,t2])
```

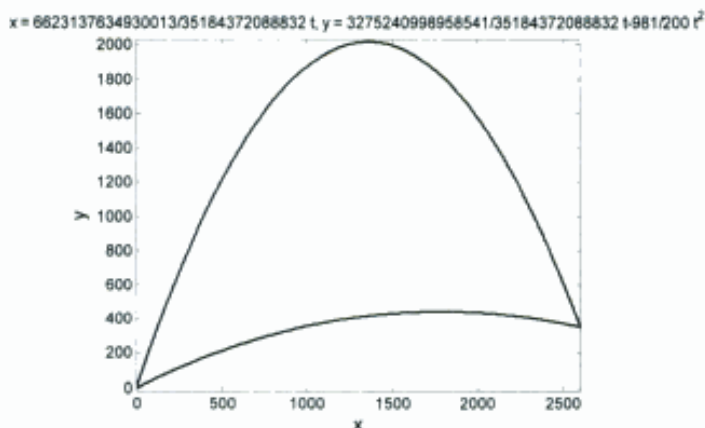
```
hold off
```

Atribui as duas soluções da parte *b* a `teta1` e `teta2`.

Esboça a primeira trajetória.

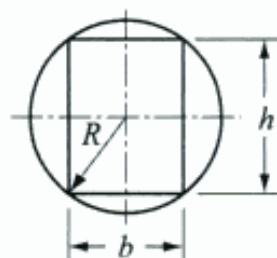
Esboça a segunda trajetória.

Quando a rotina é executada na linha do *prompt* da janela Command Window, visualizamos as duas trajetórias possíveis.



Problema-exemplo 11-3: Resistência de uma viga

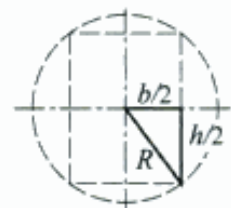
A resistência de flexão de uma viga retangular de largura b e altura h é proporcional ao momento de inércia I definido por $I = \frac{1}{12}bh^3$. Uma viga retangular foi retirada de uma peça cilíndrica original de raio R . Determine b e h (em função de R), de modo que a viga tenha o máximo valor I .



Solução

O problema será resolvido nas seguintes etapas:

- Escreva uma equação relacionando R , h e b .
- Determine uma expressão para I em termos de h .
- Calcule a derivada de I com relação a h .
- Imponha $dI/dh = 0$ e resolva para h .
- Determine o valor correspondente de b .



O primeiro passo é resolvido imediatamente a partir do triângulo destacado na figura. A

relação entre R , h e b é dada pelo Teorema de Pitágoras: $\left(\frac{b}{2}\right)^2 + \left(\frac{h}{2}\right)^2 = R^2$

Resolvendo essa equação para b , resulta: $b = \sqrt{4R^2 - h^2}$.

As demais etapas serão realizadas no MATLAB:

```
>> syms b h R
>> b=sqrt(4*R^2-h^2);
>> I=b*h^3/12
I =
1/12*(4*R^2-h^2)^(1/2)*h^3
>> ID=diff(I,h)
```

Cria uma expressão simbólica para b .

Passo b: Cria uma expressão simbólica para I .

O MATLAB substitui automaticamente b em I .

Passo c: usa o comando `diff(R)` para derivar I com relação a h .

```
ID =
-1/12/(4*R^2-h^2)^(1/2)*h^4+1/4*(4*R^2-h^2)^(1/2)*h^2
```

A derivada de I é exibida.

```
>> hs=solve(ID,h)
```

Passo d: usa o comando `solve` para resolver a equação $ID = 0$ para h . O vetor hs recebe a resposta.

```
hs =
```

```
[0]
```

```
[0]
```

```
[ 3^(1/2)*R]
```

```
[-3^(1/2)*R]
```

O MATLAB exibe quatro soluções. A solução positiva $3^{1/2}R$ é relevante ao problema.

```
>> bs=subs(b,hs(3))
```

Passo e: usa o comando `subs` para determinar b por substituição da solução h na expressão de b .

```
bs =
```

```
(R^2)^(1/2)
```

A resposta de b é exibida (A resposta vale R , mas o MATLAB insiste em exibir $(R^2)^{1/2}$).

```
>> bss=simple(bs)
```

Usa o comando `simple` para simplificar bs .

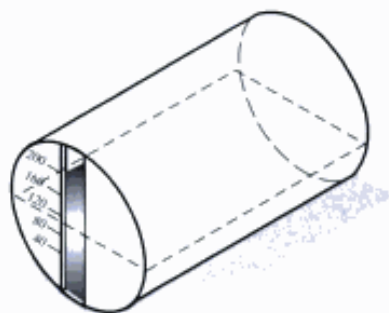
```
bss =
```

```
R
```

A resposta simplificada para b é exibida.

Problema-exemplo 11-4: Nível de combustível em um tanque

O tanque cilíndrico horizontal mostrado ao lado é utilizado para armazenar combustível em postos de gasolina. O tanque possui um diâmetro de 6m e mede 8m de comprimento. A quantidade de combustível no tanque pode ser estimada olhando-se um pequeno medidor de vidro posicionado na vertical e graduado com uma escala. A graduação da escala corresponde a 40, 60, 80, 120 e 160 mil litros de combustível. Determine a posição vertical (medida a partir do chão) onde devem estar localizadas as linhas da escala.



Solução

A relação entre o nível de combustível e o volume do tanque pode ser escrita na forma de uma integral definida. Uma vez realizada a integração, uma equação é obtida para o volume em termos da altura dentro do tanque. A altura correspondente a um volume específico pode ser determinada resolvendo-se a equação para a altura.

O volume de combustível V pode ser determinado multiplicando-se a área da seção transversal do combustível no tanque A (área sombreada) pelo comprimento do tanque L . A área da seção cruzada pode ser calculada por integração:

$$V = AL = L \int_0^h w dy$$

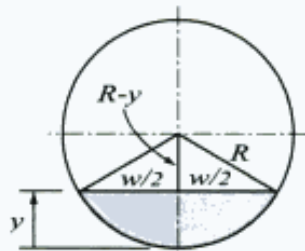
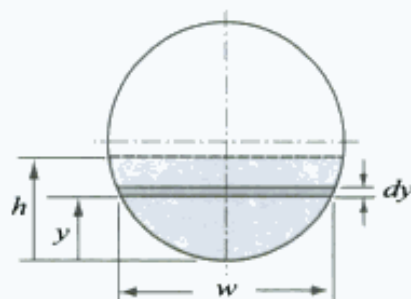
A largura w a partir do topo da superfície do combustível pode ser escrita em função de y . Do triângulo retângulo na figura à direita, as variáveis y , w e R são relacionadas por:

$$\left(\frac{w}{2}\right)^2 + (R - y)^2 = R^2$$

Resolvendo essa equação para w , resulta:

$$w = 2\sqrt{R^2 - (R - y)^2}$$

O volume de combustível no ponto de altura h pode ser determinado substituindo-se w na integral (equação para o volume) e realizando a integração. O resultado será uma equação que relaciona o volume V com a altura h . O valor de h para um dado V é obtido resolvendo-se a equação para h . Nesse problema, os valores de h devem ser determinados para os volumes 40, 60, 80, 120 e 160 mil litros. A solução desse problema é mostrada na rotina a seguir.



```
R=3; L=8;
```

```
syms w y h
```

```
w=2*sqrt(R^2-(R-y)^2)
```

```
S=L*w;
```

```
V=int(S,y,0,h)
```

```
Vescala=[40:40:200]
```

```
for i=1:5
```

```
    Veq=V-Vescala(i);
```

```
    hresp=solve(Veq);
```

```
end
```

```
hescala=double(hresp)
```

Cria uma expressão simbólica para w .

Cria a expressão que será integrada.

Usa o comando `int` para integrar S de 0 até h . O resultado é V em função de h .

Cria um vetor com os valores de V na escala.

Cada passo do `loop` resolve h para um valor de V .

Cria a equação para h que deve ser resolvida.

Usa o comando `solve` para resolver para h .

`hresp` é um vetor (simbólico com números) cujos valores de h correspondem aos valores V no vetor `Vescala`.

Usa o comando `double` para obter os valores numéricos dos elementos do vetor `hresp`.

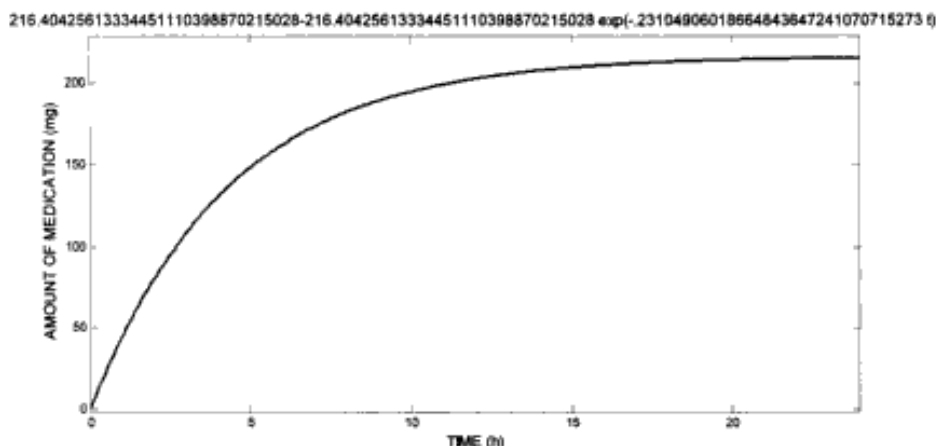
Quando a rotina (salva como `tanque`) é executada, obtemos as respostas esperadas.

```
>> tanque
```

```
w =
```

```
2*(6*y-y^2)^(1/2)
```

A expressão simbólica para w é exibida.



11.10 PROBLEMAS

1. Defina x como uma variável simbólica e crie as duas expressões simbólicas abaixo:

$$S_1 = x^3 - 9x^2 + 27x - 27 \quad \text{e} \quad S_2 = (x+3)^3 - x^3 - 5x - 12$$

Use operações simbólicas para determinar a forma mais simples das seguintes expressões:

- a) $S_1 \cdot S_2$
- b) $\frac{S_1}{S_2}$
- c) $S_1 + S_2$
- d) Use o comando `subs` para avaliar o valor numérico do resultado da parte c para $x = 10$.

2. Defina y como uma variável simbólica e crie as duas expressões simbólicas:

$$S_1 = (\sqrt{y} + 2)^2 - 2(2\sqrt{y} + 1) \quad \text{e} \quad S_2 = y_2 - 2y + 4$$

Use operações simbólicas para determinar a forma mais simples das seguintes expressões:

- a) $S_1 \cdot S_2$
- b) $\frac{S_1}{S_2}$
- c) $S_1 + S_2$
- d) Use o comando `subs` para avaliar o valor numérico do resultado da parte c para $x = 5$.

3. Defina w com uma variável simbólica e crie as seguintes expressões simbólicas:

$$S_1 = 2w^2 + 6w + 9 \quad \text{e} \quad S_2 = 2w^2 - 6w + 9$$

Use operações simbólicas para determinar a forma mais simples do produto $S_1 \cdot S_2$.

4. Defina x como uma variável simbólica.

- a) Mostre, usando o comando `factor`, que as raízes do polinômio:

$$f(x) = x^5 + 6x^4 - 6x^3 - 64x^2 - 27x + 90$$

são 1, -2, 3, -3 e -5.

- b) Determine a equação de polinômio que tem as raízes: $x = 6$, $x = -4$, $x = -1$ e $x = 2$.

5. Use os comandos da Seção 11.2 para mostrar que:

a) $\sin(3x) = 3\sin x - 4\sin^3 x$

b) $\sin x \sin y = \frac{1}{2} [\cos(x - y) - \cos(x + y)]$

c) $\cos(x + y + z) = \cos x \cos y \cos z - \sin x \sin y \cos z - \sin x \cos y \sin z - \cos x \sin y \sin z$

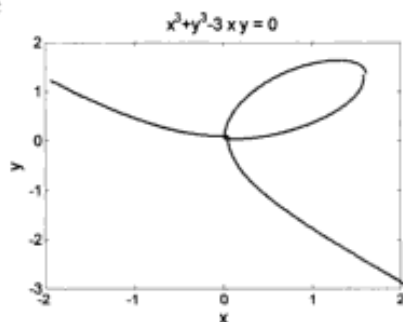
6. O *folium* de Descartes é o gráfico mostrado na figura do exercício. Na forma paramétrica, a equação do *folium* é:

$$x = \frac{3t}{1+t^3} \quad \text{e} \quad y = \frac{3t^2}{1+t^3} \quad \text{para } t \neq -1$$

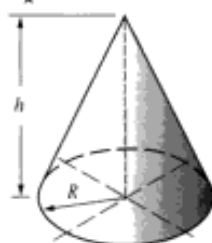
- a) Use o MATLAB para mostrar que a equação do *folium* de Descartes também pode ser escrita como:

$$x^3 + y^3 = 3xy$$

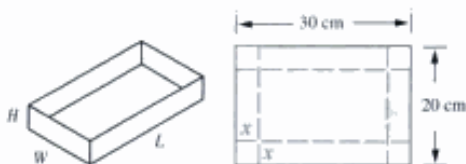
- b) Construa um gráfico do *folium* para o domínio mostrado na figura. Use o comando `ezplot`.



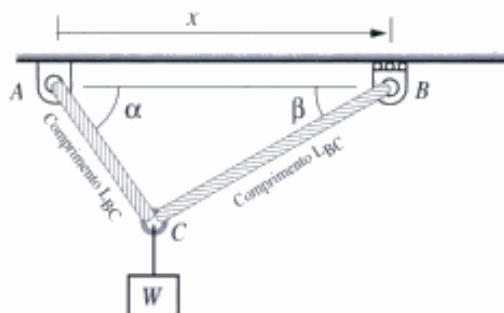
7. Um silo em formato de um cone possui a área da superfície igual a 280m^2 e altura h igual a 15m. Determine o raio R da base do cone. (Escreva uma equação para a área da superfície em termos do raio e da altura. Resolva a equação para o raio e use o comando `double` para obter um valor numérico).



8. Uma caixa é fabricada tomando-se um pedaço de papelão retangular de 20cm por 30cm, cortando-se os cantos (de acordo com a figura) e dobrando-se os lados que restaram. Determine x tal que o volume da caixa seja 1.000cm^3 (há duas possibilidades).

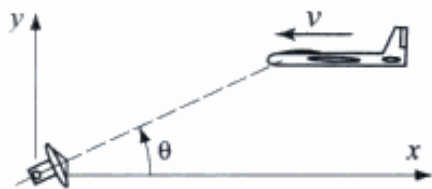


9. Um peso W é colocado em um anel montado em dois cabos, conforme mostra a figura ao lado. O suporte do cabo no ponto A é fixo e no ponto B pode ser transladada (sem atrito) na direção horizontal. As forças nos cabos F_{AC} e F_{BC} dependem da posição do suporte B (distante x de A) e podem ser calculadas por:

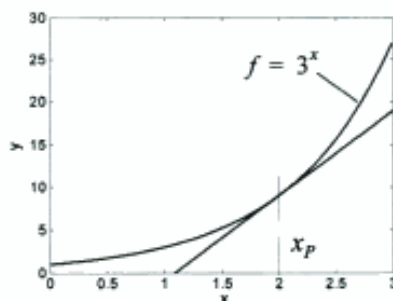


$$F_{AC} = \cos \alpha = F_{BC} \cos \beta \quad \text{e} \quad F_{AC} \sin \alpha + F_{BC} \sin \beta = W$$

- Use o MATLAB para determinar expressões para as forças F_{AC} e F_{BC} em termos de x , W e do comprimento dos cabos L_{AC} e L_{BC} .
 - Use o comando `subs` para substituir $W = 2000\text{N}$, $L_{AC} = 0.3\text{m}$ e $L_{BC} = 0.5\text{m}$ nas expressões encontradas na parte a. Isso dará as forças nos cabos em função da distância x .
 - Use o comando `ezplot` para esboçar as forças F_{AC} e F_{BC} em função de x (ambas no mesmo gráfico), começando em $x = 0.4\text{m}$ e terminando em $x = 0.8\text{m}$. O que acontece quando x aproxima-se de 0.8m ?
10. Uma equação de reta no plano x - y é dada por $y = 3x - 2$ e a equação de uma elipse é dada por $16x^2 + 32x + 4y^2 - 24y = 52$.
- Use o comando `ezplot` para esboçar a reta e a elipse no mesmo gráfico.
 - Determine as coordenadas dos pontos onde a reta intercepta a elipse.
11. Um radar está rastreando um avião voando a uma altitude constante de 5km e a uma velocidade constante de 540km/h. O avião viaja ao longo de um percurso que passa exatamente sobre o radar. O rastreamento começa quando o avião está a 100km de distância do posto onde se encontra instalado o radar.



- a) Determine uma expressão para o ângulo θ , em função do tempo, que a antena do radar faz com o solo.
- b) Determine uma expressão para a velocidade angular da antena, $\frac{d\theta}{dt}$, em função do tempo.
- c) Construa dois gráficos na mesma janela de saída. Um para θ versus t e outro para $\frac{d\theta}{dt}$ versus t , onde o ângulo está em graus e o tempo em minutos, para o intervalo $0 \leq t \leq 20$ min.
12. A função $f = 3^x$ possui uma tangente no ponto x_p . Determine a equação da reta tangente em termo de x_p . Isso significa que a equação da reta tangente terá a forma $y = mx + b$, onde m e b são funções de x_p . Considere o caso onde $x_p = 2$. Determine o ponto onde a tangente intercepta o eixo x e construa um gráfico de f e da reta tangente na mesma figura. Tome o domínio do gráfico entre $0 \leq x \leq 3$. A reta tangente deve iniciar na interseção do ponto com o eixo x e terminar em $x = 3$.



13. Calcule a integral indefinida: $I = \int e^{2x} \sqrt{2 - e^{2x}} dx$

14. A equação de uma elipse é:

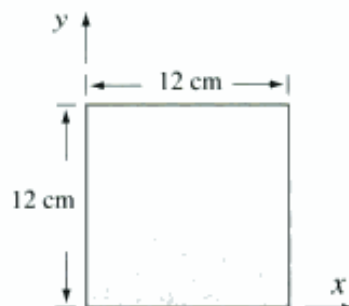
$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

Mostre que a área A encerrada pela elipse é dada por $A = \pi ab$.

15. Um ladrilho de cerâmica têm a forma mostrada na figura ao lado. A área sombreada foi pintada de vermelho e o restante do ladrilho é branco. A linha de borda entre as áreas em vermelho e branco respeitam a equação:

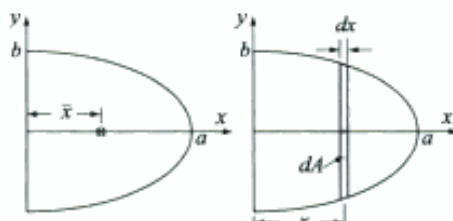
$$y = -kx^2 + 12kx$$

Determine k de modo que a área das partes em branco e vermelho sejam iguais.



16. Mostre que a localização do centróide de uma área semi-elíptica é dada por:

$$\bar{x} = \frac{4a}{3\pi}$$



A coordenada $\bar{x}$ pode ser calculada a partir de:

$$\bar{x} = \frac{\int x dA}{\int dA}$$

17. O valor *rms* de uma tensão CA é definido por:

$$v_{rms} = \sqrt{\frac{1}{T} \int_0^T v^2(t') dt'}$$

onde T é o período da forma de onda.

- a) Um tensão é dada por $v(t) = V \cos(\omega t)$. Mostre que o valor dessa forma de onda é:

$$v_{rms} = \frac{V}{\sqrt{2}}$$

e é independente de ω . (A relação entre o período T e a frequência angular ω é $T = \frac{2\pi}{\omega}$).

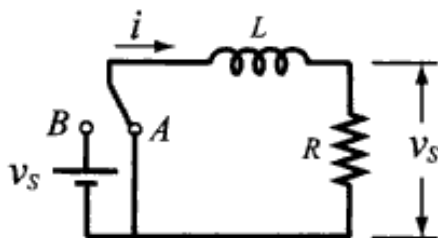
- b) Um tensão é dada por $v(t) = 2.5 \cos(350t) + 3V$. Determine o valor v_{rms} .

18. Determine a solução da equação diferencial:

$$\frac{dy}{dx} = 3y - 1.5yx$$

sujeita à condição inicial: $y(0) = 4$. Esboce a solução para $0 \leq x \leq 6$.

19. Um resistor R ($R = 0.4\Omega$) e um indutor L ($L = 0.08H$) foram conectados de acordo com a figura do problema. Inicialmente, a chave está conectada na posição A e não há corrente no circuito. Em $t = 0$, a chave é deslocada para a posição B , de forma que o resistor e o indutor sejam conectados à fonte v_s ($v_s = 6V$), e uma corrente começa a circular pelo circuito. A chave permanece conectada ao pólo B até que a tensão do resistor atinja 5V. Nesse instante (t_{RA}), a chave é deslocada novamente para a posição A . A corrente i no circuito pode ser calculada resolvendo as equações diferenciais:

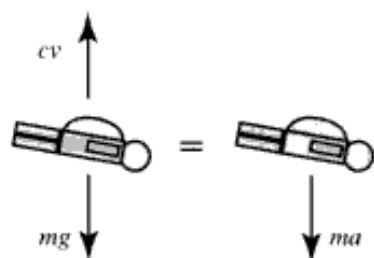


$$iR + L \frac{di}{dt} = v_s \quad \text{Durante o intervalo de tempo entre } t = 0 \text{ e } t_{RA}$$

$iR + L \frac{di}{dt} = 0$ Do instante quando a chave retorna à posição e lá permanece.

A tensão através do resistor v_R é dada, em qualquer instante de tempo, por $v_R = iR$.

- Determine uma expressão para a corrente i em termos de R , L , v_i e t para $0 \leq t \leq t_{BA}$ resolvendo a primeira equação diferencial.
 - Substitua os valores de R , L e v_i na solução para i e determine o tempo t_{BA} considerando que a tensão através dos terminais do resistor é de 5V.
 - Determine uma expressão para a corrente i em termos de R , L e t , para $t_{BA} \leq t$ resolvendo a segunda equação diferencial.
 - Construa dois gráficos (na mesma página). Uma para v_R versus t , no intervalo $0 \leq t \leq t_{BA}$, e outro para v_R versus t para $t_{BA} \leq t \leq 2t_{BA}$.
- 20.** A velocidade de um pára-quedista quando o pára-quedas ainda está fechado pode ser modelada levando-se em consideração que a resistência do ar é proporcional à velocidade. Pela segunda lei de movimento de Newton, a relação entre a massa m do pára-quedista e a velocidade v é dada por (direção para baixo é positiva):



$$mg - cv = m \frac{dv}{dt}$$

onde c é uma constante de atrito e g é a aceleração da gravidade ($g = 9.81 \text{ m/s}^2$).

- Resolva a equação para v em termos de m , g , c e t , considerando que a velocidade inicial do pára-quedista é zero.
 - Foi observado que, 4s após o salto de um pára-quedista de 90kg do avião, sua velocidade era de 28m/s. Determine c .
 - Construa um gráfico da velocidade em função do tempo para o pára-quedista no intervalo de tempo $0 \leq t \leq 30\text{s}$.
- 21.** Determine a solução geral da equação diferencial:

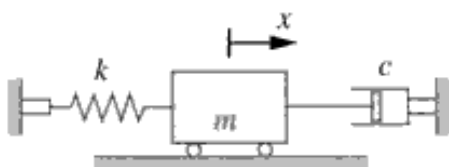
$$\frac{d^2 y}{dx^2} + 3 \frac{dy}{dx} - 2y = 0$$

Mostre que a solução está correta (obtenha a primeira e a segunda derivadas da solução e então faça a substituição de volta na equação).

22. Determine a solução da equação diferencial que satisfaz às seguintes condições iniciais:

$$\frac{d^2 y}{dx^2} - 4y = 5, \quad y(0) = 0, \quad \left. \frac{dy}{dx} \right|_{x=0} = 1$$

23. Oscilações amortecidas podem ser modeladas através do oscilador massa-mola amortecido. Da segunda lei de Newton, o deslocamento x de uma massa (m) em função do tempo pode ser determinado resolvendo-se a equação diferencial:



$$m \frac{d^2 x}{dt^2} + c \frac{dx}{dt} + kx = 0$$

onde k é a constante da mola e c é o coeficiente de atrito ou de amortização. Se a massa é deslocada da posição de equilíbrio, e então liberada de volta, inicia-se uma oscilação da mesma. A natureza das oscilações depende da massa e dos valores de c e k .

Para o sistema mostrado na figura, $m = 10\text{kg}$ e $k = 28\text{N/m}$. No tempo $t = 0$, considere que a massa está posicionada em $x = 0.18\text{m}$ e então é liberada do repouso. Determine uma expressão para o deslocamento x e outra para a velocidade v da massa em função do tempo. Considere os seguintes casos:

- a) $c = 3\text{N.s/m}$
 b) $c = 50\text{N.s/m}$

Para cada caso, esboce a posição x e a velocidade v em função do tempo (ambos os gráficos na mesma página). Para o caso (a), tome $0 \leq t \leq 20\text{s}$ e, para o caso (b), $0 \leq t \leq 10\text{s}$.

Apêndice:

Sumário de Caracteres, Comandos e Funções

As tabelas a seguir listam os caracteres, comandos e funções do MATLAB que aparecem neste livro. Os itens são agrupados por assunto.

Caracteres e operadores aritméticos

Caractere	Descrição	Página
+	Adição	24 , 66-67
-	Subtração	24 , 66-67
*	Multiplicação de escalares e arranjos	24 , 67-68
.*	Multiplicação de arranjos elemento a elemento	74-75
/	Divisão à direita	24 , 73-74
\	Divisão à esquerda	24 , 72-73
./	Divisão à direita elemento a elemento	74-75
.\	Divisão à esquerda elemento a elemento.	74-75
^	Exponenciação	24
.^	Exponenciação elemento a elemento	74-75
:	Dois pontos; cria vetores com elementos igualmente separados; representa a faixa de elementos no arranjo	42-43 , 49-50
=	Operador de atribuição	29
()	Parênteses; determina precedência em operações; especifica argumentos de entrada em funções	24 , 47-49 159-160
[]	Colchetes; forma arranjos. Especifica os argumentos de saída em funções	41-45 , 159-160
,	Vírgula; separa arranjos e argumentos de funções; separa comandos na mesma linha	22-23 , 31 , 47-51 , 159-160

Caracteres e operadores aritméticos

Caractere	Descrição	Página
;	Ponto-e-vírgula; suprime a exibição de algum comando; termina linhas de arranjos	23, 44-45
'	Aspa simples; transposta de uma matriz; inicia e termina a declaração de <i>strings</i>	47, 59-61
...	Reticências; continuação da linha	23
%	Porcentagem; denota uma linha de comentário; especifica algum formato de saída	23

Operadores lógicos e relacionais

Caractere	Descrição	Página
<	Menor que	181
>	Maior que	181
<=	Menor que ou igual a	181
>=	Maior que ou igual a	181
==	Igual a	181
~=	Diferente de	181
&	AND lógico	184-185
	OR lógico	184-185
~	NOT lógico	184-185

Comandos de controle

Comando	Descrição	Página
cd	Muda o diretório (ou pasta) corrente.	93-94
clc	Limpa a janela Command Window.	24
clear	Limpa todas as variáveis da memória.	33
clear x y z	Limpa as variáveis x, y e z da memória.	33
fclose	Fecha um arquivo.	106-107
fopen	Abre um arquivo.	106
global	Declara variáveis globais.	162
help	Exibe o Help de qualquer funcionalidade específica do MATLAB.	161-162
lookfor	Procura por uma palavra especificada em todas as entradas do Help.	160-161
who	Exibe as variáveis residentes na memória.	33, 56
whos	Exibe informações detalhadas sobre variáveis na memória.	33, 56

Variáveis predefinidas

Variável	Descrição	Página
<code>ans</code>	Valor da última expressão, operação, etc.	32
<code>eps</code>	A menor diferença entre dois números.	32
<code>i</code>	$\sqrt{-1}$	32
<code>inf</code>	Infinito.	32
<code>j</code>	Idêntico ao <code>i</code> .	32
<code>NaN</code>	Não é número.	32
<code>pi</code>	O número π .	32

Formatação dos resultados da janela Command Window

Comando	Descrição	Página
<code>format bank</code>	Dois algarismos decimais.	26-27
<code>format compact</code>	Elimina linhas em branco nos resultados.	26-27
<code>format long</code>	Formato de ponto fixo com 14 algarismos decimais.	26-27
<code>format long e</code>	Notação científica com 15 algarismos decimais.	26-27
<code>format long g</code>	Melhor em 15 algarismos entre a notação de ponto fixo ou flutuante.	26-27
<code>format loose</code>	Adiciona linhas em branco ao resultado.	26-27
<code>format short</code>	Formato de ponto fixo com 4 algarismos decimais.	26-27
<code>format short e</code>	Notação científica com 4 algarismos decimais.	26-27
<code>format short g</code>	Melhor em 5 algarismos entre a notação de ponto fixo ou flutuante.	26-27

Funções matemáticas elementares

Função	Descrição	Página
<code>abs</code>	Valor absoluto	28
<code>exp</code>	Exponencial	28
<code>factorial</code>	Função fatorial	28
<code>log</code>	Logaritmo neperiano	28
<code>log10</code>	Logaritmo base 10	28
<code>sqrt</code>	Raiz quadrada	28

Funções trigonométricas

Função	Descrição	Página	Função	Descrição	Página
<code>acos</code>	Inversa do co-seno	28	<code>cos</code>	Co-seno	28
<code>acot</code>	Inversa da cotangente	28	<code>cot</code>	Cotangente	28
<code>asin</code>	Inversa do seno	28	<code>sin</code>	Seno	28
<code>atan</code>	Inversa da tangente	28	<code>tan</code>	Tangente	28

Funções hiperbólicas

Função	Descrição	Página	Função	Descrição	Página
cosh	Co-seno hiperbólico	28	sinh	Seno hiperbólico	28
coth	Cotangente hiperbólica	28	tanh	Tangente hiperbólica	28

Funções de arredondamento

Função	Descrição	Página
ceil	Arredonda para cima (inteiro mais próximo).	29
fix	Arredonda para baixo (inteiro mais próximo).	29
floor	Arredonda para o menor inteiro negativo.	29
rem	Retorna o resto da divisão de x por y.	29
round	Arredonda para o inteiro mais próximo.	29
sign	Função sinal.	29

Criação de arranjos

Função	Descrição	Página
diag	Cria uma matriz diagonal a partir de um vetor. Cria um vetor da diagonal de uma matriz.	55
eye	Cria uma matriz identidade.	45-46, 71
linspace	Cria um vetor com elementos igualmente espaçados.	43
ones	Cria uma matriz de um's.	45-46
rand	Cria um arranjo com números aleatórios.	79-81
randn	Cria um arranjo com números normalmente distribuídos.	81
randperm	Cria vetor a partir de permutação de inteiros.	80-81
zeros	Cria um arranjo de zeros.	45-46

Manipulação de arranjos

Função	Descrição	Página
length	Retorna o número de elementos no vetor.	55
reshape	Rearranja uma matriz.	55
size	Retorna a dimensão de um arranjo.	55

Funções de arranjo

Função	Descrição	Página
cross	Calcula o produto vetorial de dois vetores.	79
det	Calcula o determinante de uma matriz.	72, 79
dot	Calcula o produto escalar de dois vetores.	68, 79

Funções de arranjo (continuação)

Função	Descrição	Página
inv	Calcula a inversa de uma matriz.	71-72, 79
max	Retorna o maior valor de uma matriz.	78-79
mean	Calcula o valor médio.	78-79
median	Semelhante a mean.	78-79
min	Calcula o menor valor de uma matriz.	78-79
sort	Arranja os elementos de uma matriz em ordem crescente.	78-79
std	Calcula o desvio-padrão.	78-79
sum	Calcula a soma dos elementos.	78-79

Entrada e saída

Comando	Descrição	Página
disp	Mostra o resultado de saída.	98-101
fprintf	Mostra ou salva o resultado de saída.	101-108
input	Fornece o controle do <i>prompt</i> para o usuário entrar com dados.	97-98
uiimport	Inicializa o assistente de importação.	110-111
xlsread	Importa dados do Excel.	109-110
xlswrite	Exporta dados para o Excel.	110

Gráficos bidimensionais

Comando	Descrição	Página
bar	Cria um gráfico em barras verticais.	140
barh	Cria um gráfico em barras horizontais.	140
fplot	Plota uma função.	129-130
hist	Cria um histograma.	141-144
hold off	Desliga o hold on.	132
hold on	Mantém o gráfico atual aberto.	132
line	Adiciona curvas a um gráfico existente.	132-133
loglog	Cria um gráfico com escalas bilog.	138-139
pie	Cria um gráfico de pizza.	141
plot	Plota um gráfico.	123-129
polar	Cria um gráfico polar.	143-144
semilogx	Cria um gráfico com escala logarítmica no eixo x.	138-139
semilogy	Cria um gráfico com escala logarítmica no eixo y.	138-139
stairs	Cria um gráfico em degraus.	140
stem	Cria um gráfico de hastes.	140

Gráficos tridimensionais

Comando	Descrição	Página
bar3	Cria um gráfico de barras verticais 3-D.	262
contour	Cria as curvas de nível de uma superfície.	261
contour3	Cria as órbitas de uma superfície.	261
cylinder	Plota um cilindro.	262
mesh	Cria um gráfico em malhas.	258-259
meshc	Cria um gráfico em malhas e as curvas de nível correspondentes.	260
meshgrid	Cria um grid para um gráfico 3-D.	256-257
meshz	Cria um gráfico em malhas e desenha cortinas laterais.	260
pie3	Cria um gráfico em pizza 3-D.	263
plot3	Cria um gráfico 3-D.	254-255
scatter3	Cria um gráfico em dispersão.	263
sphere	Plota uma esfera.	262
stem3	Cria um gráfico de hastes 3-D.	262
surf	Cria uma superfície.	258-260
surfc	Cria uma superfície e desenha as curva de nível.	260
surfl	Cria uma superfície em degradê.	260
waterfall	Cria uma superfície semelhante a uma queda d'água.	261

Formatação de gráficos

Comando	Descrição	Página
axis	Estabelece os limites dos eixos.	136-137
colormap	Configura cores.	258-259
grid	Adiciona grid ao gráfico.	137, 259
gtext	Adiciona texto ao gráfico.	134
legend	Adiciona legenda ao gráfico.	134
subplot	Cria múltiplos gráficos numa única saída.	144-145
text	Adiciona texto ao gráfico.	134
title	Adiciona título ao gráfico.	134
view	Controla o ângulo de visão de um gráfico 3-D.	263-264
xlabel	Adiciona rótulo (título) ao eixo x.	133-134
ylabel	Adiciona rótulo (título) ao eixo y.	133-134

Funções matemáticas (cria, avalia e resolve)

Comando	Descrição	Página
feval	Determina o valor de uma função matemática.	169-170
fminbnd	Determina o mínimo de uma função.	277-278
fzero	Resolve uma equação a uma variável.	275-276
inline	Cria uma função inline.	166-168

Integração numérica

Função	Descrição	Página
quad	Integra uma função.	280
quadl	Integra uma função.	280
trapz	Integra uma função.	281

Equações diferenciais ordinárias

Comando	Descrição	Página
ode113	Resolve uma EDO de primeira ordem.	283
ode15s	Resolve uma EDO de primeira ordem.	283
ode23	Resolve uma EDO de primeira ordem.	283
ode23s	Resolve uma EDO de primeira ordem.	283
ode23t	Resolve uma EDO de primeira ordem.	283
ode23tb	Resolve uma EDO de primeira ordem.	283
ode45	Resolve uma EDO de primeira ordem.	283

Funções lógicas

Função	Descrição	Página
all	Determina se todos os elementos de um arranjo são diferentes de zero.	187-188
and	Lógica AND.	186-187
any	Determina se algum elemento de um arranjo é diferente de zero.	187-188
find	Determina índices de certos elementos de um vetor.	187-188
not	Lógica NOT.	187
or	Lógica OR.	186-187
xor	Lógica XOR.	187-188

Comandos de controle de fluxo

Comando	Descrição	Página
break	Termina a execução de um laço (<i>loop</i>).	207
continue	Termina um passo de um laço.	207
else	Executa comandos condicionalmente.	192
elseif	Executa comandos condicionalmente.	193-194
end	Termina sentenças condicionais e laços.	190-191, 194-196, 198-199, 202
for	Repete a execução de um grupo de comandos.	198-199
if	Executa comandos condicionalmente.	189-190
switch	Seleciona entre vários casos, baseado numa expressão.	194-196
while	Repete a execução de um grupo de comandos.	202

Funções polinomiais

Função	Descrição	Página
conv	Multiplica polinômios.	226-227
deconv	Divide polinômios.	226-227
poly	Determina os coeficientes de um polinômio.	225-226
polyder	Determina a derivada de um polinômio.	227-228
polyval	Calcula o valor de um polinômio.	223-224
roots	Determina as raízes de um polinômio.	224-225

Ajuste de curvas e interpolação

Função	Descrição	Página
interp1	Interpolação 1-D.	236-237
polyfit	Ajusta polinomialmente um conjunto de pontos.	229-231

Matemática Simbólica

Função	Descrição	Página
collect	Agrupa termos de uma expressão.	303-304
diff	Diferencia uma equação.	313-314
double	Converte números da forma simbólica para a forma numérica.	301-302
dsolve	Resolve uma equação diferencial ordinária.	317-318
expand	Expande uma expressão.	304-305
ezplot	Plota uma expressão simbólica.	320-321
factor	Fatora em produto de polinômios.	305
findsym	Exibe as variáveis simbólicas em uma expressão.	302-303
int	Integra uma expressão.	315-316
pretty	Exibe a expressão no formato matemático.	307
simple	Determina uma forma de uma expressão simplificada.	305-307
simplify	Simplifica uma expressão.	305-306
solve	Resolve uma equação ou um sistema de equações.	307-308
subs	Substitui números em uma expressão.	
sym	Cria um objeto simbólico.	297-299
syms	Cria um objeto simbólico.	299-300

Respostas de Problemas Seleccionados

Capítulo 1

2. a) $1.7584\text{e}+003$
b) $1.0174\text{e}+007$
4. a) 1.4395
b) 0.2325
6. a) 629.1479
b) 2.0279
8. $r = 11,1\text{cm}$, $A = 0,155\text{m}^2$
12. a) $b = 17.8885\text{ cm}$
b) $\alpha = 31.5881^\circ$
14. $d = 2.4010$
16. a) \$1233.82
b) \$1301.68
c) \$1302.00
18. 707.95

Capítulo 3

2. $y = 1.0\text{e}+004 *$
-0.5954 -0.1000 -0.0116 -0.0008
-0.0000 0 0.0000 0.0008 0.0116
0.1000 0.5954 2.7000
4. 6.7415 5.4965 5.2579
5.2421 5.2415
6. $y = 2.0000$ 2.5937 2.7048
2.7156 2.7169 2.7176 2.7179
2.7181
8. a) $\text{Sn} = 0.6883$
b) $\text{Sn} = 0.6926$
c) $\text{Sn} = 0.6931$
12. 5 9957
10 19611
15 28670
20 36857
25 43925
30 49657

35	53881
40	56468
45	57339
50	56468
55	53881
60	49657
65	43925
70	36857
75	28670
80	19611
85	9957

3.0000	6.9750	4.6500
4.0000	12.4000	6.2000
5.0000	19.3750	7.7500
6.0000	27.9000	9.3000
7.0000	37.9750	10.8500
8.0000	49.6000	12.4000
9.0000	62.7750	13.9500
10.0000	77.5000	15.5000

8. $a = 2$ $b = 3.5$
 $c = 4.2$ $b = 7$

14. Resistor Corrente (A)

R_1	0.6750
R_2	1.1036
R_3	0.8011
R_4	0.3025
R_5	0.9324
R_6	1.2349
R_7	1.9553
R_8	1.3610
R_9	3.3163
R_{10}	2.8877

Capítulo 4

2. teta = 37.2750 28.1630
 22.0930 18.0097 15.1373

4.

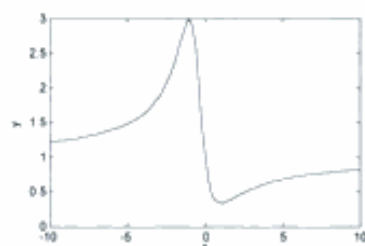
teta	tempo (h)
0	4.0857
10.0000	3.9469
20.0000	3.8407
30.0000	3.7650
40.0000	3.7232
50.0000	3.7276
60.0000	3.8126

6.

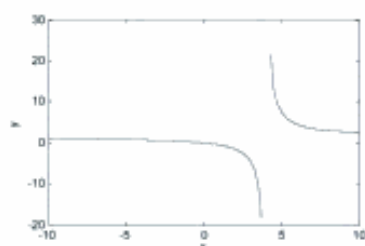
Tempo (s)	Distância (m)	Velocidade (m/s)
0	0	0
1.0000	0.7750	1.5500
2.0000	3.1000	3.1000

Capítulo 5

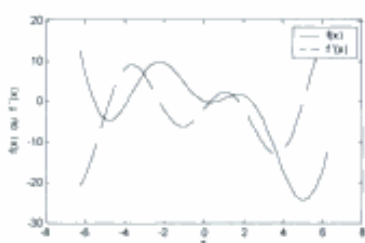
2.



4.

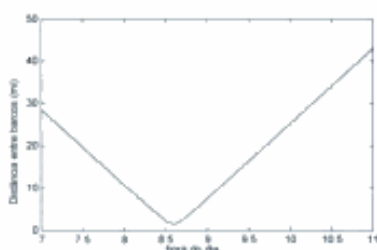


6.



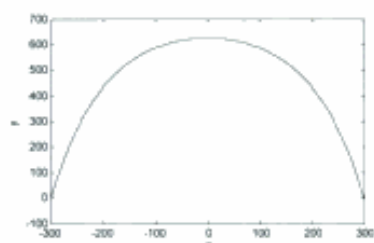
e

8.

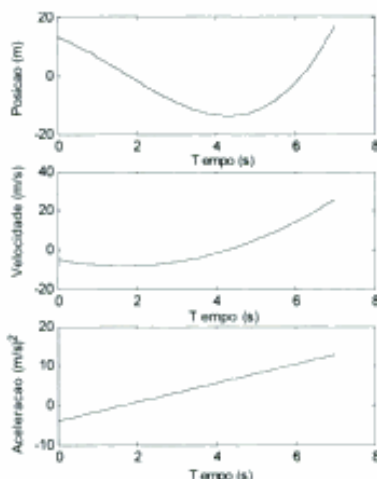


Visibilidade menor que 8 milhas ocorre entre 8,1 e 9 horas da manhã.

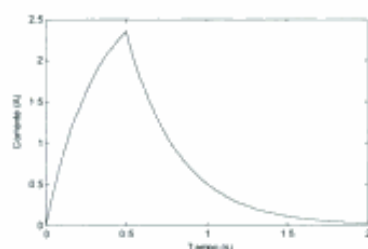
10.



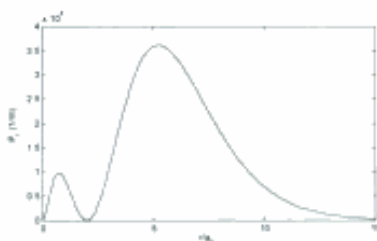
12.



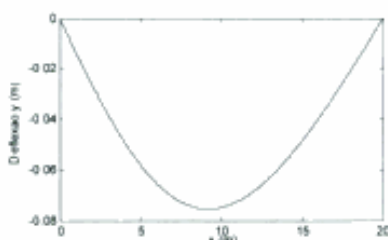
14.



16.



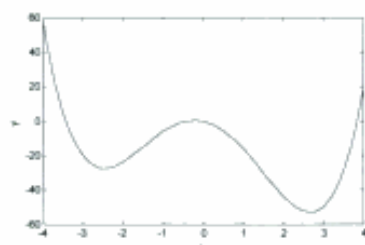
18.



Capítulo 6

2. a) $y(-3) = -20.1$, $y(5) = 237.5$

b)



4. a) (3, 21)

b) (1, 2)

6. 56.5 N·m

8. a) -0.7730 0.5797 0.2577

b) 0.7730 -0.5797 -0.2577

c) 0.7071 0 -0.7071

10. a) 75.6000

b) 81.6000

55.8000

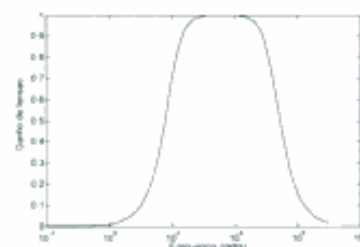
68.8000

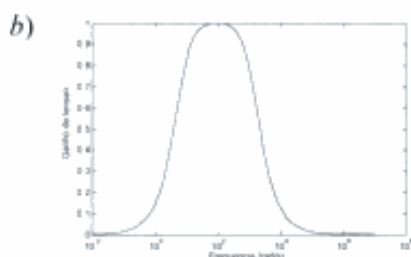
83.4000

14. a) 179.1974 MPa -69.1974 MPa

b) -6.7199 psi -21.2801 psi

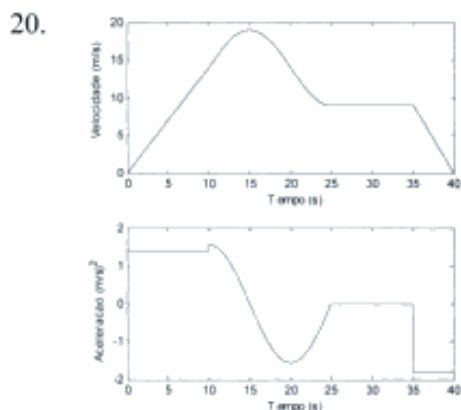
16. a)





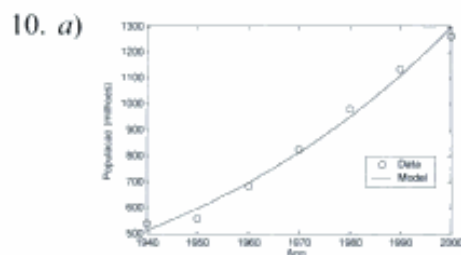
Capítulo 7

2. a) $y = 1$
b) $y = 0$
c) $y = 10$
4. $y = [2 \ 1 \ 4]$
6. a) New York 37.6774 °F
Anchorage 33.1290 °F
b) New York 17
Anchorage 13
c) 11 dias, nos dias: 1 7 9
14 15 18 19 21 22
25 26
d) 1 dia, no 23° dia.
e) 16 dias, nos dias: 7 8 9
13 14 15 16 17 18
19 20 23 24 25 26
27
8. a) A equação possui duas raízes
 $x_1 = 0.345208$,
 $x_2 = -4.345208$
b) A equação não possui raízes reais.
c) A equação possui uma raiz real:
 $x = -0.333333$
18. (11.3099, 15.2971)
(120.2564, 13.8924)
(207.8973, 19.2354)
(-33.0239, 11.9269)

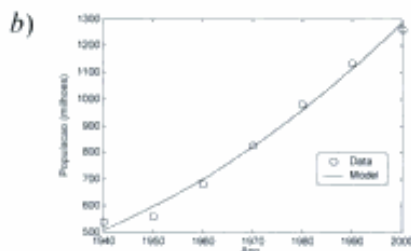


Capítulo 8

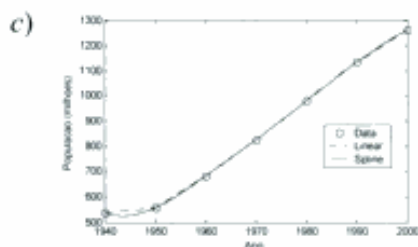
2. $3x^2 + 7x - 5$
4. $x = 0.0022785 \text{ m}$
8. $a = 7.5 \text{ m}$, $b = 5 \text{ m}$



Pop. 1955 = 644.7597 milhões

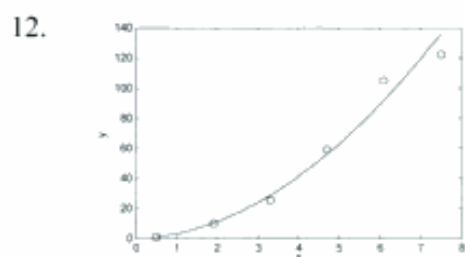


Pop. 1955 = 645.1964 milhões



Pop. 1955L = 619.5 milhões

Pop. 1955S = 613.058 milhões

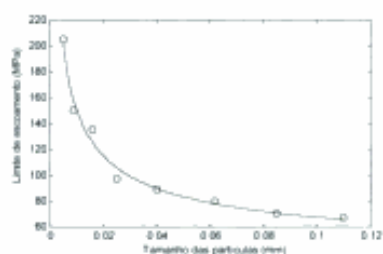


14. a) $k = 12.2603 \text{ MPa} \cdot \sqrt{\text{mm}}$

$\sigma_0 = 28.2938 \text{ MPa}$

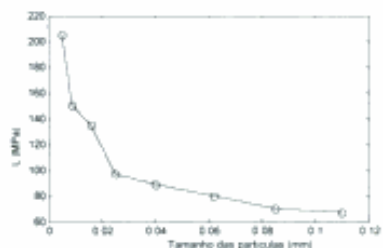
$\sigma_y = 83.1237 \text{ MPa}$ para $d = 0.05$

mm

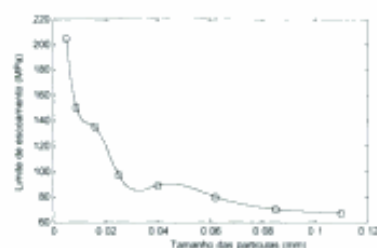


b) $\sigma_y = 84.9091 \text{ MPa}$ para $d = 0.05$

mm

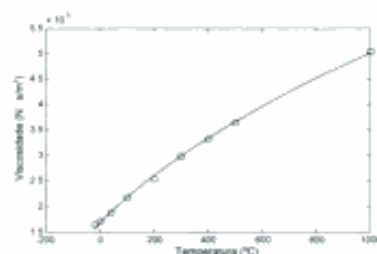


c) $\sigma_y = 88.5457 \text{ MPa}$ para $d = 0.05$
mm



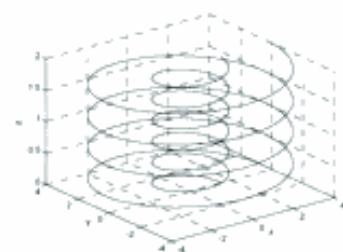
16. $C = 1.5682 \times 10^{-6} \frac{\text{Kg}}{\text{m} \cdot \text{s} \cdot \text{K}^{1/2}}$

$S = 148.1622 \text{ K}$

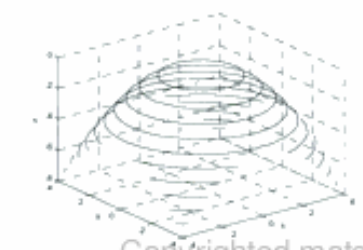
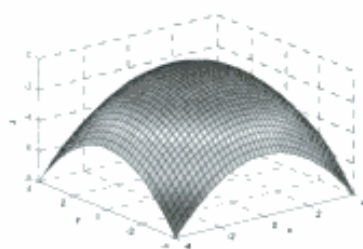


Capítulo 9

2.



4.



Índice

A

abs, 28, 342-343
acos, 28, 343-345
acot, 28, 343-345
ajuste de curvas, 222-223, 228-229
 função exponencial, 232-233
 função hiperbólica, 232-233
 função logarítmica, 232-233
 potência, 232-233
all, 185-186, 346-347
and, 184-185, 346-347
ans, 32-33, 342-343
any, 185-186, 346-347
arquivo função
 argumentos de entrada e saída, 159-160
 declarando, 157-158
 estrutura, 158-159
 linha de definição da função, 158-161
 linha H1, 160-161
 linhas de texto help, 160-161
 salvando, 158-159
 usando, 163-164
arranjo lógico, 181-182
arranjos
 adição, subtração, 66-67
 bidimensionais (matrizes), 43-44
 declarando, 40-41
 divisão, 70-71
 divisão à esquerda, 72-73
 endereçamento, matriz, 48-49
 endereçamento, vetor, 47-48
 multiplicação, 67-68

 operações elemento a elemento, 74-75
 unidimensionais (vetor), 40-41
asin, 28, 343-345
assistente de importação, 110-111
atan, 28, 343-345
axis, 134-135, 346-347

B

BackgroundColor, 136-137
bar, 137-138, 345-346
bar3, 260, 345-346
barh, 137-138, 345-346
break, 205-206, 347-348

C

caminho de procura, 95-96
caractere de escape, 101-102
caracteres gregos, 135-136
case, 193-195
cd, 93-94, 342-343
ceil, 29, 343-345
clc, 24, 342-343
clear, 33, 342-343
color, 125-126, 136-137
colormap, 258-259, 346-347
comandos de saída, 98-99
comentários, 22-23
Command Window, 19-20, 22-23
continue, 207, 347-348
contour, 260-261, 345-346
contour3, 260-261, 345-346
conv, 226-227, 347-348

H

help, 160-161,
 Help Window, 21-22
 hist, 141-144, 345-346
 histogramas, 140-144
 hold off, 131-132, 345-346
 hold on, 131-132, 345-346

I

i, 32-33, 342-343
 if, 189-190, 347-348
 importando dados, 108-109
 inf, 32-33, 342-343
 inicializando, *string*, 98-99
 inline, 164-165, 346-347
 inline, funções, 166-167
 input, 97-98, 345-346
 int, 314-315, 347-348
 integração numérica, 278-279
 integração simbólica, 314-315
 interface de ajuste de curvas, 239
 interp1, 237, 347-348
 interpolação, 235-236
 linear, 237
 nearest, 237
 spline cúbica, 237
 inv, 71-72, 79-80, 343-345

J

j, 32-33, 342-343

L

laço (*loop*)
 alinhados, 205-206
 for-end, 197-198
 while, 202
 laço infinito, 202-203
 legend, 133-134, 346-347
 length, 54-55, 343-345
 line, 131-132, 345-346
 linestyle, 125-126
 linewidth, 125-126
 LineWidth, 136-137
 linspace, 43-44, 343-345
 log, 28, 342-343
 log24, 28, 342-343
 loglog, 137-139, 345-346
 lookfor, 160-161, 342-343

M

marker, 126-127
 markeredgecolor, 125-126
 Markerfacecolor, 125-126
 Markersize, 125-126

matemática simbólica

 cálculos numéricos com, 322-323
 diferenciação, 313-314
 expressão, 299-300
 integração, 314-315
 objeto, 297-298
 plotando expressão, 319-320
 resolvendo uma equação, 307-308
 solução da EDO, 317-318
 variável, 298-299, 347-348
 variável padrão, 302-303

matriz

 adicionando elementos, 52-53
 apagando elementos, 53-54
 determinante, 72-73
 dimensão, 44-45
 identidade, 70-71
 inversa, 71-72

max, 78-79, 343-345

mean, 78-79, 343-345

median, 78-79, 343-345

mesh, 258-259, 259, 345-346

meshc, 260, 345-346

meshgrid, 256-257, 345-346

meshz, 260, 298-299

M-file, 92-93, 166

min, 78-79, 343-345

mínimos quadrados, 229-230

modificadores, texto, 134-135

N

NaN, 32-33, 342-343

nome da propriedade, 125-126, 135-136

not, 186-187, 346-347

números aleatórios, 79-80

O

ode113, 283-284, 346-347

ode15s, 283-284, 346-347

ode23, 283-284, 346-347

ode23s, 283-284, 346-347

ode23t, 283-284, 346-347

ode23tb, 283-284, 346-347

ode45, 283-284, 346-347

ones, 45-46, 343-345

operações aritméticas com escalares, 24

operações elemento por elemento, 74-75

operador de atribuição, 29

operador lógico, 184-185

operador transposta, 46-47

operadores relacionais, 181-182

or, 186-187, 347-348

ordem de precedência, 24, 183-186

otherwise, 194-196

P

pi, 32-33, 342-343
 pie, 137-139, 345-346
 pie3, 260-261, 345-346
 plot, 123, 345-346
 plot3, 254-255, 345-346
 plotando uma função, 127-130
 polar, 143-144, 345-346
 polinômio
 adição, 225-226
 derivada, 227-228
 divisão, 226-227
 representação no MATLAB, 222-223
 multiplicação, 226-227
 raízes, 224-225
 valor do, 223-224
 poly, 225-226, 347-348
 polyder, 227-228, 347-348
 polyfit, 230-231, 299-300
 polyval, 223-224, 347-348
 ponto-e-vírgula, 22-23, 29-31
 pretty, 217-218, 347-348

Q

quad, 279, 346-347
 quadl, 280-281, 346-347

R

rand, 79-81, 343-345
 randn, 81-82, 343-345
 randperm, 80-81, 343-345
 rem, 29, 343-345
 reshape, 55-56, 343-345
 reticências, 22-23
 roots, 224-225, 347-348
 Rotation, 135-136
 rotinas, 91-92
 declarando, 92-93
 executando, 93-94
 inicializando, 95-99
 saída, 98-109
 salvando, 92-93
 round, 29, 343-345

S

salvar num arquivo, 105-106
 scatter, 262-263, 345-346
 scatter3, 262-263
 semilogx, 137-139, 345-346
 semilogy, 137-139, 345-346
 sentença condicional
 if-else-end, 191-192
 if-elseif-else-end, 193-194
 if-end, 189-190

sentença switch-case, 192-193
 Set Path Window, 95-96
 sign, 29, 347-348
 símbolo de porcentagem, 22-23
 sin, 28, 347-348
 sinh, 28, 347-348
 size, 55-56, 347-348
 sobrescrito, 134-135
 sort, 78-79, 345-346
 sphere, 261-262, 345-346
 sqrt, 28, 342-343
 stairs, 140-141, 345-346
 std, 79-80, 345-346
 stem, 140-141, 345-346
 stem3, 262-263, 345-346
 string, entrada, 98-99
 strings, 59-62
 subplot, 144-145, 346-347
 subs, 323-324, 347-348
 subscrito, 134-135
 sum, 78-79, 345-346
 surf, 258-259, 260, 345-346
 surfc, 260, 345-346
 surf1, 260-261, 345-346
 switch, 193-194, 299-300
 sym, 297-298, 347-348
 Syms, 299-300, 347-348

T

tabela, exibição, 86-87, 100-101
 tabela verdade, 188-189
 tan, 28, 343-345
 tanh, 28, 343-345
 tecla de navegação, 22-23
 text, 133-134, 346-347
 texto, modificadores, 134-135
 title, 133-134, 346-347
 trapz, 281-282, 346-347

U

uiimport, 110-111, 345-346

V

valor da propriedade, 125-126, 135-136
 variável
 declarando, escalar, 29
 declarando, matriz, 43-47
 declarando, vetor, 41-44
 global, 95-96, 162-163
 local, 161-162
 nome, 32-33
 predefinida, 32-33
 vetor
 adicionando elementos, 51-52
 apagando elementos, 53-54

declarando, [41-42](#)
 espaçamento constante, [42-44](#)
vetores lógicos, [183-184](#)
vetorização, [77-78](#)
view, [263-264](#), [346-347](#)

W

waterfall, [260-261](#), [346-347](#)
while, [202](#), [347-348](#)
who, [33](#), [56-57](#), [342-343](#)
whos, [33](#), [56-57](#), [342-343](#)

X

xlabel, [133-134](#), [346-347](#)
xlsread, [109-110](#), [345-346](#)
xlswrite, [109-110](#), [345-346](#)
xor, 187-188, 347-348

Y

ylabel, [133-134](#), [346-347](#)

Z

zeros, [45-46](#), [343-345](#)

SEGUNDA EDIÇÃO

MATLAB®

COM APLICAÇÕES EM ENGENHARIA

UM MODO EFICIENTE DE APRENDER A UTILIZAR O MATLAB

Esta obra é um guia pelo universo desse poderoso software. Atualizado para incorporar a versão 7.0 do MATLAB, trata de todos os aspectos necessários à utilização efetiva e adequada do programa. Escrito de forma didática, inicia pelos tópicos mais básicos, passando por operações aritméticas com escalares e arranjos (vetores e matrizes) e finalizando com gráficos tridimensionais e equações diferenciais ordinárias. Telas detalhadas do programa, tutoriais, exemplos trabalhados e problemas propostos na matemática, ciências e engenharias proporcionam o domínio completo do programa.

DESTAQUES:

- ✓ Atualizado para incorporar a versão 7.0 do MATLAB
- ✓ Introdução à matemática simbólica, um recurso importante do MATLAB
- ✓ Seção sobre importação e exportação de dados
- ✓ Exemplos de programas bem elaborados e completos são uma valiosa fonte de experiência para os iniciantes
- ✓ Exemplos comentados auxiliam os usuários a aprender o programa e os seus comandos no formato faça você mesmo
- ✓ Programas/exemplos, aplicações e problemas propostos permitem aos professores mostrar o poder do MATLAB na resolução de problemas na ciência e engenharia

MATLAB, com Aplicações em Engenharia, 2.ed., é um recurso valioso para professores, estudantes e profissionais das diversas engenharias e demais áreas das ciências exatas. É um livro único por conter aplicações em uma grande quantidade de áreas científicas e tecnológicas.

ISBN 978-85-363-0692-6



bookman®
EMPRESA DO GRUPO ARTMED

www.bookman.com.br

Copyrighted material