

cordigostorly
L2602466

ALGORITMOS NUMÉRICOS

2.^a edição

Frederico Ferreira Campos, filho

Professor do Departamento de Ciência da Computação da
Universidade Federal de Minas Gerais



LTC
EDITORIA

No Capítulo 4 são descritas as técnicas de regressão linear simples e múltipla, por quadrados mínimos, para ajuste de curvas. Também é mostrada uma forma alternativa de estimar os parâmetros de regressão via decomposição em valores singulares, bem como a diferença entre regressão e interpolação.

O Capítulo 5 apresenta os métodos de integração numérica de Newton-Cotes (trapézio, 1/3 e 3/8 de Simpson) e quadratura de Gauss-Legendre, sendo proposta uma técnica iterativa para integração. É feita a análise de erro para cada uma dessas regras de integração. São também demonstradas as fórmulas de Newton-Cotes e Gauss-Legendre para integração dupla.

O Capítulo 6 mostra, inicialmente, como isolar raízes de equações algébricas e transcendentais. Em seguida, são descritos métodos para fazer o refinamento de uma raiz: bisseção, métodos baseados em interpolação linear (secante, *regula falsi* e pégaso), métodos baseados em interpolação quadrática (Muller e van Wijngaarden-Dekker-Brent) e métodos baseados em tangentes (Newton e Schröder). Para cada método é feita uma análise da sua ordem de convergência.

O Capítulo 7 é dedicado às equações diferenciais ordinárias (EDO). São apresentados o problema de valor inicial e os métodos de Runge-Kutta e de Adams para resolvê-lo. Também se mostra como resolver sistemas de EDO, o que possibilita a solução de equações diferenciais ordinárias de ordem superior.

Após a descrição dos métodos são mostrados dois exemplos de aplicação para a solução de problemas reais. Ao final de cada capítulo são propostos exercícios de fixação. Em todos os capítulos são apresentados algoritmos dos métodos, descritos na notação algorítmica proposta, tornando fácil a sua implementação em qualquer linguagem de programação.

Nos apêndices é mostrado como implementar os algoritmos em diversas linguagens de programação: FORTRAN (Apêndice A), Pascal (Apêndice B) e MATLAB (Apêndice C). O Apêndice D apresenta as respostas de alguns exercícios propostos.

O autor gostaria de agradecer a todas as pessoas que tornaram possível a elaboração de Algoritmos Numéricos, em especial a Líliam César de Castro Medeiros, Silvana Bocanegra, Jones Oliveira de Albuquerque, Daniela Dias Rodrigues, Helton Fábio de Matos, Marcos Augusto dos Santos e Letícia Pereira Pinto.

As sugestões para aprimorar o presente texto, bem como para efetuar correções, serão bem-vindas pelo e-mail: ffcampos@dcc.ufmg.br. O site de Algoritmos Numéricos é www.dcc.ufmg.br/algoritmosnumericos. Nele os professores poderão encontrar um vasto material de suporte às aulas.

Belo Horizonte, janeiro de 2007

Frederico Ferreira Campos, filho

Sumário

| | |
|--|----|
| 1 Computação numérica | 1 |
| 1.1 Etapas na solução de um problema | 2 |
| 1.1.1 Definição do problema | 2 |
| 1.1.2 Modelagem matemática | 2 |
| 1.1.3 Solução numérica | 2 |
| 1.1.4 Análise dos resultados | 4 |
| 1.2 Notação algorítmica | 4 |
| 1.2.1 Estrutura do algoritmo | 4 |
| 1.2.2 Variáveis e comentários | 5 |
| 1.2.3 Expressões e comando de atribuição | 6 |
| 1.2.4 Comandos de entrada e saída | 8 |
| 1.2.5 Estruturas condicionais | 8 |
| 1.2.6 Estruturas de repetição | 10 |
| 1.2.7 Falha no algoritmo | 11 |
| 1.2.8 Exemplos de algoritmos | 12 |
| 1.3 Notação matemática | 14 |
| 1.4 Complexidade computacional | 17 |
| 1.5 Implementação de algoritmos | 19 |
| 1.6 Tipos de erros | 21 |
| 1.7 Aritmética de ponto flutuante | 23 |
| 1.8 Exercícios | 27 |
| 2 Sistemas lineares | 29 |
| 2.1 Conceitos fundamentais | 29 |
| 2.1.1 Alguns tipos de matrizes | 30 |
| 2.1.2 Operações matriciais | 31 |
| 2.1.3 Noções sobre autovalores e autovetores | 35 |
| 2.1.4 Normas | 39 |
| 2.1.5 Sistemas de equações lineares | 42 |
| 2.1.6 Classificação de sistemas | 42 |
| 2.2 Sistemas triangulares | 45 |
| 2.2.1 Sistema triangular inferior | 45 |
| 2.2.2 Sistema triangular superior | 47 |
| 2.2.3 Complexidade computacional | 49 |
| 2.3 Eliminação de Gauss | 50 |

| | | | |
|---|-----|--|-----|
| 2.3.1 Sistemas equivalentes | 50 | 3.1.1 Interpolação linear | 126 |
| 2.3.2 Operações l-elementares | 51 | 3.1.2 Interpolação quadrática | 127 |
| 2.3.3 Sistema triangular equivalente | 51 | 3.2 Polinômios de Lagrange | 128 |
| 2.3.4 Cálculo do determinante | 54 | 3.2.1 Fórmula de Lagrange | 128 |
| 2.3.5 Pivotação parcial | 56 | 3.2.2 Dispositivo prático | 130 |
| 2.4 Decomposição LU | 57 | 3.2.3 Algoritmo e complexidade computacional | 131 |
| 2.4.1 Cálculo dos fatores | 58 | 3.3 Polinômios de Newton | 133 |
| 2.4.2 Pivotação parcial | 59 | 3.3.1 Operador de diferença dividida | 133 |
| 2.4.3 Cálculo do determinante | 61 | 3.3.2 Fórmula de Newton | 134 |
| 2.4.4 Sistema com matriz singular | 64 | 3.3.3 Algoritmo e complexidade computacional | 136 |
| 2.4.5 Algoritmos e complexidade | 66 | 3.4 Polinômios de Gregory-Newton | 139 |
| 2.4.6 Sistemas lineares complexos | 69 | 3.4.1 Operador de diferença finita ascendente | 139 |
| 2.5 Decomposição de Cholesky e LDL^T | 71 | 3.4.2 Fórmula de Gregory-Newton | 140 |
| 2.5.1 Cálculo do fator | 72 | 3.4.3 Algoritmo e complexidade computacional | 142 |
| 2.5.2 Cálculo do determinante | 73 | 3.5 Escolha dos pontos para interpolação | 144 |
| 2.5.3 Algoritmo e complexidade | 77 | 3.6 Erro de truncamento da interpolação polinomial | 145 |
| 2.5.4 Fatoração LDL^T | 79 | 3.7 Comparação das complexidades | 148 |
| 2.6 Decomposição espectral | 82 | 3.8 <i>Splines</i> cúbicos | 149 |
| 2.6.1 Cálculo dos autovetores | 83 | 3.8.1 Cálculo dos coeficientes | 149 |
| 2.6.2 Solução de sistema linear | 84 | 3.8.2 Sistema linear subdeterminado | 150 |
| 2.7 Uso da decomposição | 85 | 3.9 <i>Splines</i> cúbicos naturais | 151 |
| 2.7.1 Refinamento da solução | 85 | 3.9.1 Cálculo das derivadas | 151 |
| 2.7.2 Cálculo da matriz inversa | 87 | 3.9.2 Algoritmo e complexidade | 155 |
| 2.8 Métodos iterativos estacionários | 88 | 3.10 <i>Splines</i> cúbicos extrapolados | 157 |
| 2.8.1 Condição de convergência | 89 | 3.10.1 Cálculo das derivadas | 157 |
| 2.8.2 Critério de parada | 89 | 3.10.2 Algoritmo e complexidade | 161 |
| 2.8.3 Método de Jacobi | 90 | 3.11 Avaliação dos <i>splines</i> cúbicos | 163 |
| 2.8.4 Método de Gauss-Seidel | 96 | 3.11.1 Algoritmo e complexidade | 164 |
| 2.8.5 Método da sobre-relaxação sucessiva | 102 | 3.11.2 Avaliação | 164 |
| 2.8.6 Análise de convergência | 105 | 3.12 Comparação dos <i>splines</i> cúbicos | 166 |
| 2.8.7 Comparação dos métodos iterativos estacionários | 107 | 3.13 Exemplos de aplicação | 168 |
| 2.8.8 Refinamento como método estacionário | 109 | 3.13.1 Curva de titulação | 168 |
| 2.9 Análise de erro na solução de sistemas | 110 | 3.13.2 Interpolação inversa | 171 |
| 2.9.1 Malcondicionamento | 111 | 3.14 Exercícios | 173 |
| 2.9.2 Número de condição | 112 | | |
| 2.9.3 Sensibilidade da solução | 115 | | |
| 2.10 Exemplos de aplicação | 117 | 4 Ajuste de curvas | 177 |
| 2.10.1 Tensões em circuito elétrico | 117 | 4.1 Regressão linear simples | 178 |
| 2.10.2 Estequiometria de reação química | 118 | 4.1.1 Diagrama de dispersão | 178 |
| 2.11 Exercícios | 121 | 4.1.2 Retas de regressão | 178 |
| 3 Interpolação polinomial | 125 | 4.1.3 Método dos quadrados mínimos | 181 |
| 3.1 Polinômios interpoladores | 125 | 4.2 Qualidade do ajuste | 183 |
| | | 4.2.1 Coeficiente de determinação | 183 |
| | | 4.2.2 Variância residual | 185 |

| | | |
|-------|---|------------|
| 4.3 | Regressão linear múltipla | 187 |
| 4.3.1 | Equações normais | 187 |
| 4.3.2 | Regressão polinomial | 189 |
| 4.3.3 | Algoritmo e complexidade | 189 |
| 4.3.4 | Transformações não lineares | 195 |
| 4.3.5 | Malcondicionamento | 196 |
| 4.4 | Ajuste via decomposição em valores singulares | 197 |
| 4.4.1 | Cálculo dos parâmetros | 197 |
| 4.4.2 | Decomposição em valores singulares | 198 |
| 4.4.3 | Algoritmo e complexidade | 200 |
| 4.4.4 | Comparação dos métodos computacionais para RLM | 203 |
| 4.5 | Diferença entre regressão e interpolação | 204 |
| 4.6 | Exemplos de aplicação | 205 |
| 4.6.1 | Tensão-deformação de aço | 205 |
| 4.6.2 | Produto iônico da água | 206 |
| 4.7 | Exercícios | 209 |
| 5 | Integração numérica | 211 |
| 5.1 | Fórmulas de Newton-Cotes | 211 |
| 5.1.1 | Regra do trapézio | 216 |
| 5.1.2 | Regra do 1/3 de Simpson | 218 |
| 5.1.3 | Regra dos 3/8 de Simpson | 220 |
| 5.1.4 | Erro de integração dos métodos de Newton-Cotes | 222 |
| 5.1.5 | Algoritmo e complexidade | 226 |
| 5.2 | Quadratura de Gauss-Legendre | 229 |
| 5.2.1 | Fórmula para dois pontos | 229 |
| 5.2.2 | Fórmula geral | 234 |
| 5.2.3 | Erro de integração da fórmula de Gauss-Legendre | 239 |
| 5.2.4 | Algoritmos e complexidade | 240 |
| 5.3 | Comparação dos métodos de integração simples | 244 |
| 5.4 | Integração numérica iterativa | 247 |
| 5.5 | Integração dupla pelas fórmulas de Newton-Cotes | 248 |
| 5.5.1 | Fórmulas simples | 249 |
| 5.5.2 | Fórmulas compostas | 252 |
| 5.5.3 | Algoritmo | 253 |
| 5.6 | Integração dupla via fórmulas de Gauss-Legendre | 256 |
| 5.6.1 | Fórmula para dois pontos | 256 |
| 5.6.2 | Fórmula geral | 258 |
| 5.6.3 | Algoritmo | 260 |
| 5.7 | Comparação dos métodos para integração dupla | 262 |
| 5.8 | Exemplos de aplicação | 263 |
| 5.8.1 | Distribuição de probabilidade | 264 |
| 5.8.2 | Integral imprópria | 265 |

| | | |
|-------|---|------------|
| 5.9 | Exercícios | 268 |
| 6 | Raízes de equações | 271 |
| 6.1 | Isolamento de raízes | 272 |
| 6.1.1 | Equações algébricas | 272 |
| 6.1.2 | Equações transcendentas | 284 |
| 6.1.3 | Convergência da raiz | 287 |
| 6.2 | Método da bisseção | 288 |
| 6.3 | Métodos baseados em aproximação linear | 292 |
| 6.3.1 | Método da secante | 293 |
| 6.3.2 | Método da <i>regula falsi</i> | 295 |
| 6.3.3 | Método pégaso | 297 |
| 6.3.4 | Ordem de convergência | 300 |
| 6.4 | Métodos baseados em aproximação quadrática | 301 |
| 6.4.1 | Método de Muller | 301 |
| 6.4.2 | Método de van Wijngaarden-Dekker-Brent | 305 |
| 6.5 | Métodos baseados em tangente | 308 |
| 6.5.1 | Método de Newton | 308 |
| 6.5.2 | Método de Schröder | 313 |
| 6.6 | Comparação dos métodos para cálculo de raízes | 314 |
| 6.7 | Exemplos de aplicação | 316 |
| 6.7.1 | Juros de financiamento | 316 |
| 6.7.2 | Cabo suspenso | 319 |
| 6.8 | Exercícios | 321 |
| 7 | Equações diferenciais ordinárias | 323 |
| 7.1 | Solução numérica de EDO | 324 |
| 7.1.1 | Problema de valor inicial | 324 |
| 7.1.2 | Método de Euler | 325 |
| 7.1.3 | Definições | 326 |
| 7.2 | Métodos de Runge-Kutta | 328 |
| 7.2.1 | Métodos de segunda ordem | 328 |
| 7.2.2 | Métodos de quarta ordem | 331 |
| 7.2.3 | Método de Dormand-Prince | 333 |
| 7.3 | Métodos de Adams | 335 |
| 7.3.1 | Métodos de passo dois | 336 |
| 7.3.2 | Métodos de passo três | 338 |
| 7.3.3 | Método de Adams-Bashforth-Moulton de quarta ordem | 339 |
| 7.4 | Comparação de métodos para EDO | 341 |
| 7.4.1 | Métodos de Runge-Kutta | 341 |
| 7.4.2 | Métodos de Adams | 342 |
| 7.4.3 | Comparação de métodos para EDO | 342 |
| 7.5 | Sistemas de equações diferenciais ordinárias | 343 |

| | |
|---|------------|
| 7.5.1 Algoritmo de Runge-Kutta para sistema de ordem dois | 344 |
| 7.5.2 Equações diferenciais de segunda ordem | 345 |
| 7.6 Exemplos de aplicação | 347 |
| 7.6.1 Controle de poluição | 347 |
| 7.6.2 Deflexão de viga | 349 |
| 7.7 Exercícios | 351 |
| A Linguagem FORTRAN | 353 |
| A.1 Estrutura de um programa FORTRAN | 353 |
| A.2 Variáveis e comentários | 354 |
| A.3 Expressões e comando de atribuição | 355 |
| A.4 Comandos de entrada e saída | 357 |
| A.5 Estruturas condicionais | 360 |
| A.6 Estruturas de repetição | 362 |
| A.7 Falha no programa | 364 |
| A.8 Subprogramas | 364 |
| A.9 Exemplos de programas | 367 |
| B Linguagem Pascal | 371 |
| B.1 Estrutura de um programa Pascal | 371 |
| B.2 Variáveis e comentários | 372 |
| B.3 Expressões e comando de atribuição | 373 |
| B.4 Comandos de entrada e saída | 375 |
| B.5 Estruturas condicionais | 379 |
| B.6 Estruturas de repetição | 381 |
| B.7 Falha no programa | 384 |
| B.8 Subprogramas | 384 |
| B.9 Exemplos de programas | 386 |
| C Linguagem MATLAB | 391 |
| C.1 Estrutura de um programa MATLAB | 391 |
| C.2 Variáveis e comentários | 391 |
| C.3 Expressões e comando de atribuição | 392 |
| C.4 Comandos de entrada e saída | 394 |
| C.5 Estruturas condicionais | 400 |
| C.6 Estruturas de repetição | 401 |
| C.7 Falha no algoritmo | 403 |
| C.8 Subprogramas | 403 |
| C.9 Exemplos de programas | 405 |
| D Respostas dos exercícios | 409 |
| Referências Bibliográficas | 419 |
| Índice | 423 |

ALGORITMOS NUMÉRICOS

Capítulo 1

Computação numérica

Uma etapa intermediária importante durante a solução de um problema envolve a elaboração de um algoritmo, o qual deverá ser posteriormente implementado em uma linguagem de programação para a obtenção dos resultados numéricos em um computador.

O Cálculo Numérico é uma metodologia para resolver problemas matemáticos por intermédio de um computador, sendo amplamente utilizado por engenheiros e cientistas. Uma solução via Cálculo Numérico é sempre numérica, enquanto os métodos analíticos usualmente fornecem um resultado em termos de funções matemáticas. Muito embora uma solução numérica seja uma aproximação do resultado exato, ela pode ser obtida em grau crescente de exatidão. Uma solução numérica é calculada mesmo quando o problema não tem solução analítica, fato comum nas equações diferenciais. A integral indefinida

$$\int e^{-x^2} dx,$$

de grande utilidade em Estatística, possui primitiva que não pode ser representada, explicitamente, por funções elementares. A área sob a curva descrita por e^{-x^2} de a até b pode ser determinada por meio de algoritmos numéricos que são aplicáveis a qualquer outro integrando, não sendo, portanto, necessário fazer substituições especiais ou mesmo a integração por partes a fim de obter o resultado.

Para computar um resultado numérico, são necessárias as operações aritméticas (adição, subtração, multiplicação e divisão) e lógicas (comparação, conjunção, disjunção e negação). Considerando que estas são as únicas operações matemáticas que os computadores são capazes de realizar, então os computadores e o Cálculo Numérico formam uma combinação perfeita. Cumpre relembrar que o primeiro computador de grande porte totalmente eletrônico, o ENIAC (Electronic Numerical Integrator And Calculator), foi projetado para fazer cálculos balísticos e, atualmente, os maiores supercomputadores no mundo inteiro estão dedicados a realizar cálculos numéricos.

1.1 Etapas na solução de um problema

Dado um problema qualquer, como resolvê-lo no computador utilizando as técnicas do Cálculo Numérico? Será mostrado, a partir de um exemplo simples, que a solução de um problema pode ser obtida em quatro etapas: definição do problema, modelagem matemática, solução numérica e análise dos resultados.

1.1.1 Definição do problema

Nesta etapa, define-se qual é o *problema real* a ser resolvido. Seja, por exemplo, calcular \sqrt{a} , $a > 0$ usando apenas as quatro operações aritméticas.

1.1.2 Modelagem matemática

O *problema real* é transformado no *problema original* por meio de uma formulação matemática. No exemplo,

$$x = \sqrt{a} \longrightarrow x^2 = a \longrightarrow f(x) = x^2 - a = 0.$$

O *problema real*, calcular \sqrt{a} , $a > 0$, foi transformado no *problema original*, que é determinar a raiz de uma equação algébrica de grau 2.

Geralmente, o *problema original* possui mais soluções que o *problema real*. No exemplo, $+\sqrt{a}$ e $-\sqrt{a}$ são as duas raízes da equação algébrica.

1.1.3 Solução numérica

Nesta etapa, é feita a escolha do método numérico mais apropriado para resolver o *problema original* oriundo da modelagem matemática. Feita a escolha do método, este é descrito por intermédio de um algoritmo, o qual é posteriormente implementado em um computador para obtenção dos resultados numéricos.

Esta etapa pode ser subdividida em três fases: elaboração do algoritmo, codificação do programa e processamento do programa.

Elaboração do algoritmo

Um algoritmo é a descrição de um conjunto de comandos que, quando ativados, resultam em uma sucessão finita de acontecimentos. Em vez de implementar um método diretamente em uma linguagem de programação, é preferível descrevê-lo por meio de uma notação algorítmica. Com isso, é possível abstrair dos detalhes da linguagem de programação do computador e concentrar apenas nos aspectos matemáticos do método.

Além do mais, a descrição do método em uma notação algorítmica facilita a sua implementação em qualquer linguagem de programação. Na Seção 1.2, é apresentada a notação algorítmica adotada para descrever os métodos numéricos incluídos neste texto.

Codificação do programa

Nesta fase, o algoritmo é implementado na linguagem de programação escolhida. Como os aspectos matemáticos do método já foram pensados na fase de elaboração do algoritmo, a questão agora é só se preocupar com os detalhes de implementação da linguagem adotada. Os apêndices mostram como passar da notação algorítmica descrita na Seção 1.2 para as linguagens de programação FORTRAN (Apêndice A), Pascal (Apêndice B) e MATLAB (Apêndice C).

Processamento do programa

Finalmente, o código do programa obtido da implementação do algoritmo em uma linguagem de programação deve ser editado em um arquivo para que possa ser executado pelo computador. Se for detectado algum erro de sintaxe oriundo da fase de codificação, ele tem que ser corrigido para que o programa possa ser executado. Se na fase de processamento ocorrer algum erro de lógica, ou seja, a execução do programa produz resultados inesperados, então deve-se retornar à fase de elaboração para corrigir o algoritmo.

Exemplo 1.1 Para exemplificar a etapa de solução numérica no exemplo de cálculo de \sqrt{a} , será utilizado o método de Newton, a ser descrito na Seção 6.5.1, para calcular uma raiz de $f(x) = x^2 - a = 0$

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}.$$

Substituindo $f(x)$ e $f'(x)$ na expressão acima, tem-se que

$$x_{k+1} = x_k - \frac{x_k^2 - a}{2x_k} = x_k - \frac{x_k}{2} + \frac{a}{2x_k},$$

ou seja,

$$x_{k+1} = \left(x_k + \frac{a}{x_k} \right) \times 0,5.$$

Este é um processo iterativo para calcular \sqrt{a} , a partir de um valor inicial x_0 , usando apenas as operações aritméticas. Ele foi proposto pelos matemáticos babilônicos, mas, às vezes, também é atribuído a Heron de Alexandria (≈ 100 d. C.) ou ao grego Arquitas (428-365 a. C.). Para o cálculo de $\sqrt{9}$, usando $x_0 = 1$, o processo babilônico produz os resultados

| i | x_i | x_i-3 |
|---|--------|--------|
| 0 | 1.0000 | |
| 1 | 5.0000 | 2.0000 |
| 2 | 3.4000 | 0.4000 |
| 3 | 3.0235 | 0.0235 |
| 4 | 3.0001 | 0.0001 |
| 5 | 3.0000 | 0.0000 |

A coluna x_i mostra as sucessivas aproximações de $\sqrt{9}$ a cada iteração i e a coluna x_i-3 apresenta a diferença entre o valor aproximado x_i e o valor exato 3.

1.1.4 Análise dos resultados

A adequação da solução numérica ao *problema real* é verificada nesta última etapa. Se a solução não se mostrar satisfatória, deve-se obter um novo *problema original* por intermédio de uma nova formulação matemática e determinar uma nova solução numérica.

Exemplo 1.2 Para o exemplo de cálculo de raiz quadrada, se for atribuído o valor inicial $x_0 = -1$ (ou qualquer $x_0 < 0$), então o processo convergirá para -3 , que, embora seja uma raiz de $f(x) = x^2 - 9 = 0$, não é $\sqrt{9}$.

| i | x_i | x_i-3 |
|---|---------|---------|
| 0 | -1.0000 | |
| 1 | -5.0000 | -8.0000 |
| 2 | -3.4000 | -6.4000 |
| 3 | -3.0235 | -6.0235 |
| 4 | -3.0001 | -6.0001 |
| 5 | -3.0000 | -6.0000 |

Alguns modelos matemáticos podem produzir soluções que não têm sentido físico ou químico, como, por exemplo, tempo negativo, concentração complexa etc. O objetivo da análise dos resultados é justamente discernir qual a solução válida dentre as várias fornecidas pelo modelo matemático.

1.2 Notação algorítmica

A descrição de um algoritmo¹, por intermédio de uma notação algorítmica, melhora o seu entendimento, pois apenas os aspectos do raciocínio lógico são enfatizados, sem ser necessário levar em consideração os detalhes de implementação de uma linguagem de programação. Os algoritmos deste texto são descritos em uma notação baseada naquela proposta por Farrer e outros [18]. Apesar da descrição nos apêndices de como implementar os algoritmos deste texto em algumas linguagens, a literatura deve ser consultada para obter mais material de referência das linguagens de programação FORTRAN [16], Pascal [17], MATLAB [32] ou mesmo outra que se deseja utilizar.

1.2.1 Estrutura do algoritmo

Um algoritmo deve iniciar com

Algoritmo <nome-do-algoritmo>

e terminar com

finalgoritmo

¹Esta palavra deriva do nome do matemático árabe Mohammed ibu-Musa al-Khowarizmi (≈ 800 d.C.).

Também,

{ Objetivos <objetivo-do-algoritmo> }

deve ser utilizado para descrever a finalidade do algoritmo. Os dados necessários para a execução de um algoritmo são requisitados por meio do comando

parâmetros de entrada <lista-de-variáveis>

onde *<lista-de-variáveis>* são os nomes das variáveis, separadas por vírgulas, contendo os valores fornecidos. Não se faz necessário descrever exatamente como os valores dessas variáveis serão fornecidos ao algoritmo. Compete ao programador decidir durante a codificação do programa se os dados serão fornecidos interativamente pelo teclado, lidos de um arquivo, passados como argumentos de um subprograma ou até mesmo definidos como constantes dentro do próprio programa.

Por outro lado, os valores de interesse calculados pelo algoritmo são disponibilizados pelo comando

parâmetros de saída <lista-de-variáveis>

podendo a *<lista-de-variáveis>* ser ampliada ou reduzida pelo programador.

Exemplo 1.3 Este exemplo ilustra a estrutura básica de um algoritmo que deve ser implementado como um subprograma. Ele recebe os parâmetros de entrada necessários à sua execução e retorna os parâmetros de saída calculados.

Algoritmo Exemplo
{ Objetivo: Mostrar a estrutura de um algoritmo }
parâmetros de entrada a, b, c
parâmetros de saída x, y

⋮
finalgoritmo

1.2.2 Variáveis e comentários

Uma variável corresponde a uma posição de memória do computador onde está armazenado um determinado valor. As variáveis são representadas por identificadores que são cadeias de caracteres alfanuméricos, podendo os elementos de vetores e matrizes ser referenciados por subscritos ou índices. Por exemplo, v_i ou $v(i)$ e m_{ij} ou $m(i,j)$.

Um comentário é um texto inserido em qualquer parte do algoritmo para aumentar a sua clareza. Esse texto deve ser delimitado por chaves { <texto> }, como, por exemplo, { cálculo da raiz }.

1.2.3 Expressões e comando de atribuição

Existem três tipos de expressões: aritméticas, lógicas e literais, dependendo dos tipos dos operadores e das variáveis envolvidas.

Expressões aritméticas

Expressão aritmética é aquela cujos operadores são aritméticos e cujos operandos são constantes e/ou variáveis aritméticas. A notação é semelhante àquela utilizada para representar uma fórmula como $\sqrt{b^2 - 4 * a * c}$, $\cos(2 + x)$ ou *massa * velocidade*.

O símbolo \leftarrow é usado para atribuir o resultado de uma expressão a uma variável, ou seja,

<variável> \leftarrow <expressão>

Por exemplo, *velocidade* \leftarrow *deslocamento/tempo*. A Tabela 1.1 apresenta algumas funções matemáticas que aparecem nos algoritmos descritos neste texto.

Tabela 1.1 Funções matemáticas.

| Função | Descrição | | Função | Descrição | |
|------------------|--|--|-------------------|---|--|
| Trigonométricas | | | | | |
| sen | seno | | cos | co-seno | |
| tan | tangente | | sec | secante | |
| Exponenciais | | | | | |
| exp | exponencial | | log ₁₀ | logaritmo decimal | |
| log _e | logaritmo natural | | raiz ₂ | raiz quadrada | |
| Numéricas | | | | | |
| abs | valor absoluto | | quociente | divisão inteira | |
| arredonda | arredonda em direção ao inteiro mais próximo | | sinal | sinal(<i>x</i>) = 1 se <i>x</i> > 0, = 0 se <i>x</i> = 0 e = -1 se <i>x</i> < 0 | |
| max | maior valor | | resto | resto de divisão | |
| min | menor valor | | trunca | arredonda em direção a 0 | |

Exemplo 1.4 A Tabela 1.2 mostra exemplos de uso das funções matemáticas numéricas. ■

Tabela 1.2 Resultados de funções matemáticas numéricas.

| Função | <i>x</i> e <i>y</i> | Valor | <i>x</i> e <i>y</i> | Valor |
|----------------------------------|---------------------|-------|---------------------|-------|
| abs(<i>x</i>) | 5 | 5 | -3 | 3 |
| arredonda(<i>x</i>) | 0,4 | 0 | 0,5 | 1 |
| quociente(<i>x</i> , <i>y</i>) | 5 e 3 | 1 | 3 e 5 | 0 |
| resto(<i>x</i> , <i>y</i>) | 5 e 3 | 2 | 3 e 5 | 3 |
| sinal(<i>x</i>) | -2 | -1 | 7 | 1 |
| trunca(<i>x</i>) | 1,1 | 1 | 1,9 | 1 |

Expressões lógicas

Expressão lógica é aquela cujos operadores são lógicos e cujos operandos são relações e/ou variáveis do tipo lógico. Uma relação é uma comparação realizada entre valores do mesmo tipo. A natureza da comparação é indicada por um operador relacional definido conforme a Tabela 1.3. O resultado de uma relação ou de uma expressão lógica é verdadeiro ou falso.

Tabela 1.3 Operadores relacionais.

| Operador relacional | Descrição |
|---------------------|------------------|
| > | maior que |
| \geq | maior ou igual a |
| < | menor que |
| \leq | menor ou igual a |
| = | igual a |
| \neq | diferente de |

Exemplo 1.5 Sendo *c* = 1 e *d* = 3, então *c* \leq *d* é verdadeiro, enquanto se *x* = 2, *y* = 3 e *z* = 10, então *x* + *y* = *z* é falso. ■

Os operadores lógicos mostrados na Tabela 1.4 permitem a combinação ou negação das relações lógicas.

Tabela 1.4 Operadores lógicos.

| Operador lógico | Uso |
|-----------------|-----------|
| e | conjunção |
| ou | disjunção |
| não | negação |

A Tabela 1.5 mostra os resultados obtidos com os operadores lógicos, sendo o significado de V verdadeiro e o de F falso.

Tabela 1.5 Resultados com operadores lógicos.

| <i>a</i> e <i>b</i> | | <i>a</i> ou <i>b</i> | | não <i>a</i> | |
|---------------------|---|----------------------|---|--------------|---|
| <i>a</i> \ <i>b</i> | V | F | V | F | V |
| V | V | F | V | V | V |
| F | F | F | F | V | F |

Exemplo 1.6 Para os valores definidos no Exemplo 1.5, o resultado de (*d* > *c* e *x* + *y* = *z*) é V e V, que implica verdadeiro. Por outro lado, (*d* = *c* ou *x* + *y* = *z*) é F ou F, implicando falso. ■

Expressões literais

Uma expressão literal é formada por operadores literais e operandos, os quais são constantes e/ou variáveis do tipo literal.

O caso mais simples de uma expressão literal é uma constante literal, a qual é constituída por uma cadeia de caracteres delimitada por aspas, por exemplo, *mensagem* ← “matriz singular”.

1.2.4 Comandos de entrada e saída

O comando

leia <lista-de-variáveis>

é usado para indicar que a *<lista-de-variáveis>* está disponível para leitura em algum dispositivo externo. Por sua vez, o comando

escreva <lista-de-variáveis>

deve ser utilizado para indicar onde certos valores de interesse estão disponíveis no programa e podem ser escritos em algum dispositivo externo. Compete ao programador decidir pela ampliação da *<lista-de-variáveis>* ou mesmo a omissão do comando escreva.

Exemplo 1.7 Elaborar um algoritmo para ler uma temperatura em grau Fahrenheit e converter para grau Celsius.

Algoritmo Converte_grau
 { Objetivo: Converter grau Fahrenheit para Celsius }
 leia *Fahrenheit*
 $Celsius \leftarrow (Fahrenheit - 32) * 5/9$
 escreva *Fahrenheit, Celsius*
 finalgoritmo

1.2.5 Estruturas condicionais

O uso de uma estrutura condicional torna possível a escolha dos comandos a serem executados quando certa condição for ou não satisfeita, possibilitando, desta maneira, alterar o fluxo natural de comandos. Esta condição é representada por uma expressão lógica. As estruturas condicionais podem ser simples ou compostas.

Estrutura condicional simples

Esta estrutura apresenta a forma

**se <condição> então
 <comandos>
 fimse**

Neste caso, a lista de *<comandos>* será executada se, e somente se, a expressão lógica *<condição>* tiver como resultado o valor verdadeiro.

Exemplo 1.8 Fazer um algoritmo para calcular o logaritmo decimal de um número positivo.

Algoritmo Logaritmo_decimal
 { Objetivo: Calcular logaritmo decimal }
 leia *x*
 se *x* > 0 então
 $LogDec \leftarrow \log_{10}(x)$
 escreva *x, LogDec*
 fimse
 finalgoritmo

Os comandos $LogDec \leftarrow \log_{10}(x)$ e escreva *x, LogDec* só serão executados se a variável *x* contiver um valor maior que zero.

Estrutura condicional composta

Quando houver duas alternativas possíveis, deve ser usada uma estrutura da forma

**se <condição> então
 <comandos_1>
 senão
 <comandos_2>
 fimse**

Se a expressão lógica *<condição>* tiver como resultado o valor verdadeiro, então a seqüência *<comandos_1>* será executada e a seqüência *<comandos_2>* não será executada.

Por outro lado, se o resultado de *<condição>* for falso, então será a lista *<comandos_2>* a única a ser executada.

Exemplo 1.9 Elaborar um algoritmo para avaliar a função modular $f(x) = |2x|$.

Algoritmo Função_modular
 { Objetivo: Avaliar uma função modular }
 leia *x*
 se *x* ≥ 0 então
 $f_x \leftarrow 2 * x$
 senão
 $f_x \leftarrow -2 * x$
 fimse
 escreva *x, fx*
 finalgoritmo

Se a variável x contiver um valor positivo ou nulo, então o comando $fx \leftarrow 2 * x$ será executado, seguindo-se o comando escreva x , fx . No entanto, se x contiver um valor negativo, os comandos $fx \leftarrow -2 * x$ e escreva x , fx serão os únicos a serem executados. ■

1.2.6 Estruturas de repetição

Uma estrutura de repetição faz com que uma seqüência de comandos seja executada repetidamente até que uma dada condição de interrupção seja satisfeita.

Existem, basicamente, dois tipos dessas estruturas, dependendo de ser o número de repetições indefinido ou definido.

Número indefinido de repetições

Este tipo de estrutura de repetição apresenta a forma

```

repita
  <comandos_1>
  se <condição> então
    interrompa
  fimse
  <comandos_2>
fimrepita
<comandos_3>
  
```

O comando **interrompa** faz com que o fluxo de execução seja transferido para o comando imediatamente a seguir do **fim repita**. Assim, as listas **<comandos_1>** e **<comandos_2>** serão repetidas até que a expressão lógica **<condição>** resulte no valor verdadeiro. Quando isso ocorrer, a repetição será interrompida (**<comandos_2>** não será executada) e a lista **<comandos_3>**, após ao **fim repita**, será executada.

Exemplo 1.10 Escrever um algoritmo para determinar o maior número de ponto flutuante (ver Seção 1.7) que, somado a 1, seja igual a 1.

```

Algoritmo Epsilon
{ Objetivo: Determinar a precisão da máquina }
  Epsilon ← 1
  repita
    Epsilon ← Epsilon/2
    se Epsilon + 1 = 1 então
      interrompa
    fimse
  fimrepita
  escreva Epsilon
  finalgoritmo
  
```

Esta seqüência faz com que seja calculada a chamada precisão da máquina ε . Quando a variável **Epsilon** assumir um valor que, adicionado a 1, seja igual a 1, então a estrutura **repita-fim repita** é abandonada e o comando **escreva Epsilon** será executado. ■

A forma **repita-fim repita** é o caso geral de uma estrutura de repetição. Se a lista **<comandos_1>** não existir, ter-se-á uma estrutura de repetição com interrupção no início (estrutura **enquanto**). Similarmente, se não houver a lista **<comandos_2>**, então será uma estrutura com interrupção no final (estrutura **repita-até**).

Número definido de repetições

Quando se souber com antecedência quantas vezes a estrutura deve ser repetida, pode ser usado um comando de forma mais simples

```

para <controle> ← <valor-inicial> até <valor-final> passo <delta> faça
  <comandos>
fimpara
  
```

Nesta estrutura, inicialmente, é atribuído à variável **<controle>** o valor de **<valor-inicial>** e verificado se ele é maior do que o **<valor-final>**. Se for maior, a estrutura **para-faça** não será executada. Se for menor ou igual, então os **<comandos>** serão executados e a variável **<controle>** será incrementada com o valor de **<delta>**. Novamente, é verificado se a variável **<controle>** é maior do que o **<valor-final>**; se não for maior, então os **<comandos>** serão executados e assim sucessivamente.

As repetições se processam até que a variável **<controle>** seja maior do que o **<valor-final>**. Quando o incremento **<delta>** tiver o valor 1, então o passo **<delta>** pode ser omitido da estrutura **para-faça**.

Exemplo 1.11 Escrever um algoritmo para mostrar que a soma dos n primeiros números ímpares é igual ao quadrado de n .

```

Algoritmo Primeiros_imparés
{ Objetivo: Verificar propriedade dos números ímpares }
  leia n
  Soma ← 0
  para i ← 1 até 2 * n - 1 passo 2 faça
    Soma ← Soma + i
  fimpara
  escreva Soma, n2
  finalgoritmo
  
```

1.2.7 Falha no algoritmo

O comando

abandone

é usado para indicar que haverá uma falha evidente na execução do algoritmo, por exemplo, uma divisão por zero, uma singularidade da matriz ou mesmo o uso inapropriado de parâmetros. Neste caso, a execução será cancelada. O programador deve implementar o **abandone** na linguagem de programação a ser usada, caso o comando não esteja disponível.

1.2.8 Exemplos de algoritmos

Exemplo 1.12 Dado um vetor x com n componentes, a Figura 1.1 mostra um algoritmo para calcular a média aritmética \bar{x} e o desvio padrão s^2 de seus elementos, sabendo que

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \text{ e } s^2 = \frac{1}{n-1} \left(\sum_{i=1}^n x_i^2 - \frac{1}{n} \left(\sum_{i=1}^n x_i \right)^2 \right).$$

Algoritmo Média_desvio
 { Objetivo: Calcular média aritmética e desvio padrão }
 parâmetros de entrada n, x
 { tamanho e elementos do vetor }
 parâmetros de saída $Média, DesvioPadrão$
 Soma $\leftarrow 0$
 Soma2 $\leftarrow 0$
 para $i \leftarrow 1$ até n faça
 Soma $\leftarrow Soma + x(i)$
 Soma2 $\leftarrow Soma2 + x(i)^2$
 fimpara
 Média $\leftarrow Soma/n$
 DesvioPadrão $\leftarrow \text{raiz}_2((Soma2 - Soma^2/n)/(n-1))$
 escreva Média, DesvioPadrão
 finalgoritmo

Figura 1.1 Algoritmo para cálculo da média aritmética e desvio padrão.

(Ver significado da função raiz_2 na Tabela 1.1, na página 6.)

Exemplo 1.13 A Figura 1.2 apresenta um algoritmo para determinar o maior elemento em cada linha de uma matriz A de dimensão $m \times n$.

Exemplo 1.14 Um algoritmo para calcular o valor de π , com precisão dada, utilizando a série

$$\pi = 4 \left(1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \dots \right)$$

é exibido na Figura 1.3. Cumpre ressaltar que esta não é a maneira mais eficiente de calcular π , mas serve para ilustrar o uso da estrutura de repetição **repita-fim repita**.

Algoritmo Matriz_maior

{ Objetivo: Determinar maior elemento em cada linha da matriz }
 parâmetros de entrada m, n, A
 { número de linhas, número de colunas e elementos da matriz }
 parâmetros de saída $Maior$
 { vetor contendo o maior elemento de cada linha }
 para $i \leftarrow 1$ até m faça
 Maior(i) $\leftarrow A(i, 1)$
 para $j \leftarrow 2$ até n faça
 se $A(i, j) > Maior(i)$ então
 Maior(i) $\leftarrow A(i, j)$
 fimse
 fimpara
 escreva $i, Maior(i)$
 fimpara
 finalgoritmo

Figura 1.2 Algoritmo para determinar o maior elemento da linha de uma matriz.

Algoritmo Calcular_pi

{ Objetivo: Calcular o valor de π }
 parâmetros de entrada Precisão
 { precisão no cálculo de π }
 parâmetros de saída pi
 Soma $\leftarrow 1$
 Sinal $\leftarrow -1$
 Denominador $\leftarrow 3$
 repita
 Soma $\leftarrow Soma + Sinal/Denominador$
 se $1/Denominador < \text{Precisão}$ então
 interrompa
 fimse
 Sinal $\leftarrow -Sinal$
 Denominador $\leftarrow Denominador + 2$
 fimrepita
 pi $\leftarrow 4 * Soma$
 finalgoritmo

Figura 1.3 Algoritmo para calcular o valor de π .

Exemplo 1.15 Conforme mostrado no Exemplo 4.5, o polinômio de quadrados mínimos que aproxima \sqrt{x} para $0,01 \leq x \leq 1$ é $P(x) = 1,01865x^3 - 2,17822x^2 + 2,06854x + 0,10113$.

Pelo processo de Horner (ver Seção 6.1.1), o polinômio pode ser escrito na forma $P(x) = ((1,01865x - 2,17822)x + 2,06854)x + 0,10113$, de forma a economizar operações aritméticas. Na Figura 1.4, é apresentado um algoritmo para calcular uma aproximação de \sqrt{x} usando o polinômio acima.

```
Algoritmo Aproximar_raiz
{ Objetivo: Calcular valor aproximado da raiz quadrada }
parâmetros de entrada x
{ valor que se deseja uma aproximação da raiz quadrada }
parâmetros de saída Aprox
{ aproximação de quadrados mínimos da raiz quadrada }
se x < 0,01 ou x > 1 então
    escreva "argumento fora dos limites"
    abandone
fimse
c(1) ← 1,01865; c(2) ← -2,17822; c(3) ← 2,06854; c(4) ← 0,10113
Aprox ← c(1)
para i ← 2 até 4 faça
    Aprox ← Aprox * x + c(i)
fimpara
finalgoritmo
```

Figura 1.4 Algoritmo para avaliar uma aproximação de quadrados mínimos de \sqrt{x} .

Exemplo 1.16 Um algoritmo para calcular \sqrt{a} , $a > 0$, baseado no processo babilônico, descrito na Seção 1.1.3, é mostrado na Figura 1.5.

1.3 Notação matemática

Definida a modelagem matemática por meio de expressões aritméticas e lógicas, o passo seguinte é passar dessa notação matemática para a notação algorítmica proposta. Esta passagem será ilustrada por meio de alguns exemplos.

Exemplo 1.17 A Figura 1.6 mostra um algoritmo para calcular a norma-2 ou norma Euclidiana de um vetor x de tamanho n , definida por

$$\|x\|_2 = \sqrt{\sum_{i=1}^n |x_i|^2}.$$

Exemplo 1.18 O algoritmo da Figura 1.7 determina a norma- ∞ ou norma de máxima magnitude de um vetor x de tamanho n

$$\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|.$$

```
Algoritmo Raiz2
{ Objetivo: Calcular raiz quadrada pelo processo babilônico }
parâmetros de entrada a, Toler
{ valor para calcular a raiz e tolerância }
parâmetros de saída Raiz { raiz quadrada de a }
{ teste se a é não positivo }
se a ≤ 0 então escreva "argumento inválido", abandone, fimse
{ cálculo do valor inicial z0 = z }
c(1) ← 1,01865; c(2) ← -2,17822; c(3) ← 2,06854; c(4) ← 0,10113
p ← 1; b ← a
se a > 1 então
    repita
        b ← b * 0,01; p ← p * 10
        se b ≤ 1 então interrompa, fimse
    fimrepita
fimse
se a < 0,01 então
    repita
        b ← b * 100; p ← p * 0,1
        se b ≥ 0,01 então interrompa, fimse
    fimrepita
fimse
z ← c(1)
para i ← 2 até 4 faça
    z ← z * b + c(i)
fimpara
z ← z * p; i ← 0; escreva i, z
{ cálculo da raiz }
repita
    x ← (z + a/z) * 0,5; Delta ← abs(x - z); i ← i + 1
    escreva i, x, Delta
    se Delta ≤ Toler ou i = 50 então interrompa, fimse; z ← x
fimrepita
{ teste de convergência }
se Delta ≤ Toler então
    Raiz ← x
senão
    escreva "processo não convergiu com 50 iterações"
fimse
finalgoritmo
```

Figura 1.5 Cálculo de raiz quadrada pelo processo babilônico.

(Ver significado da função abs na Tabela 1.1, na página 6.)

```

Algoritmo Norma2
{ Objetivo: Calcular a norma-2 de um vetor }
parâmetros de entrada  $n, x$ 
    { tamanho do vetor e o vetor }
parâmetros de saída  $N2$ 
    { norma-2 do vetor }
Soma  $\leftarrow 0$ 
para  $i \leftarrow 1$  até  $n$  faça
    Soma  $\leftarrow$  Soma + ( $\text{abs}(x(i))$ ) $^2$ 
fimpara
 $N2 \leftarrow \text{raiz}_2(\text{Soma})$ 
fimalgoritmo

```

Figura 1.6 Norma-2 de um vetor x de tamanho n .(Ver significado das funções abs e raiz₂ na Tabela 1.1, na página 6.)

```

Algoritmo NormalInf
{ Objetivo: Calcular a norma- $\infty$  de um vetor }
parâmetros de entrada  $n, x$ 
    { tamanho do vetor e o vetor }
parâmetros de saída  $Ninf$ 
    { norma- $\infty$  do vetor }
 $Ninf \leftarrow \text{abs}(x(1))$ 
para  $i \leftarrow 2$  até  $n$  faça
    se  $\text{abs}(x(i)) > Ninf$  então
         $Ninf \leftarrow \text{abs}(x(i))$ 
    fimse
fimpara
fimalgoritmo

```

Figura 1.7 Norma- ∞ de um vetor x de tamanho n .

(Ver significado da função abs na Tabela 1.1, na página 6.)

Exemplo 1.19 A Figura 1.8 apresenta um algoritmo para calcular o vetor x ($n \times 1$) resultante do produto de uma matriz A ($n \times m$) por um vetor v ($m \times 1$)

$$x_i = \sum_{j=1}^m a_{ij}v_j, \quad i = 1, 2, \dots, n.$$

```

Algoritmo Matvet
{ Objetivo: Calcular o produto de uma matriz por um vetor }
parâmetros de entrada  $n, m, A, v$ 
    { número de linhas, número de colunas, }
    { elementos da matriz e elementos do vetor }
parâmetros de saída  $x$ 
    { vetor resultante do produto matriz-vetor }
Soma  $\leftarrow 0$ 
para  $i \leftarrow 1$  até  $n$  faça
    Soma  $\leftarrow 0$ 
    para  $j \leftarrow 1$  até  $m$  faça
        Soma  $\leftarrow$  Soma +  $A(i,j) \cdot v(j)$ 
    fimpara
     $x(i) \leftarrow \text{Soma}$ 
fimpara
fimalgoritmo

```

Figura 1.8 Produto matriz-vetor.

1.4 Complexidade computacional

É usual definir uma função de complexidade para medir o custo de execução de um programa. Esta função pode ser tanto uma medida do tempo para executar o algoritmo que resolve um problema de tamanho n quanto o espaço de memória requerido para esta execução.

A complexidade computacional de um algoritmo se refere à estimativa do esforço computacional despendido para resolver o problema e é medido pelo número necessário de operações aritméticas e lógicas como, por exemplo, o número de adições e multiplicações efetuadas para resolver um sistema linear de ordem n .

Os problemas possuem complexidade de tempo que pode ser enquadrada em dois grupos [40]. O primeiro é composto pelos algoritmos polinomiais, sendo a função de complexidade da forma $O(c_p n^p + c_{p-1} n^{p-1} + \dots + c_1 n + c_0)$. O outro grupo é formado pelos algoritmos exponenciais, onde a função de complexidade tem a forma $O(c^n)$, $c > 1$.

Os algoritmos estudados neste texto são polinomiais. Como as operações aritméticas demandam diferentes tempos para serem executadas pelo computador, a função de complexidade será definida, separadamente, para adição, multiplicação e divisão, sendo uma subtração contada como uma adição.

Por exemplo, o número de adições necessárias para fazer a decomposição LU de uma matriz de ordem n é $O(\frac{1}{3}n^3 - \frac{1}{2}n^2 + \frac{1}{6}n)$, ou, simplesmente, $O(n^3)$. O número de multiplicações utilizadas para resolver um sistema triangular inferior de ordem n usando as substituições sucessivas é $O(\frac{1}{2}n^2 - \frac{1}{2}n)$ ou $O(n^2)$.

Exemplo 1.20 Seja o polinômio interpolador de Lagrange de grau n , definido por (3.5), na página 129

$$L_n(x) = \sum_{i=0}^n y_i \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j}.$$

Expandindo, resulta a Expressão 1, cujo algoritmo é mostrado na Figura 1.9

$$\begin{aligned} L_n(x) &= y_0 \times \frac{x - x_1}{x_0 - x_1} \times \frac{x - x_2}{x_0 - x_2} \times \dots \times \frac{x - x_n}{x_0 - x_n} \\ &+ y_1 \times \frac{x - x_0}{x_1 - x_0} \times \frac{x - x_2}{x_1 - x_2} \times \dots \times \frac{x - x_n}{x_1 - x_n} \\ &\dots + y_n \times \frac{x - x_0}{x_n - x_0} \times \frac{x - x_1}{x_n - x_1} \times \dots \times \frac{x - x_{n-1}}{x_n - x_{n-1}}. \end{aligned}$$

```
Algoritmo Lagrange Expressão 1
{ Objetivo: Interpolar usando polinômio de Lagrange }
parâmetros de entrada m, x, y, Z
{ número de pontos, abscissas }
{ ordenadas e valor a interpolar }
parâmetros de saída r { valor interpolado }
r ← 0
para i ← 1 até m faça
    p ← y(i)
    para j ← 1 até m faça
        se i ≠ j então
            p ← p * ((Z - x(j)) / (x(i) - x(j)))
        fimse
    fimpara
    r ← r + p
fimpara
finalgoritmo
```

Figura 1.9 Exemplo de algoritmo do polinômio de Lagrange.

Considerando que o número de pontos m usados na interpolação é igual a $n + 1$, onde n é o grau do polinômio, então a complexidade computacional do algoritmo da Figura 1.9 é

$$\text{Adições: } \sum_{i=1}^m 2(m-1) + 1 = 2m^2 - 2m + m = 2(n+1)^2 - (n+1) = 2n^2 + 3n + 1;$$

$$\text{Multiplicações: } \sum_{i=1}^m (m-1) = m^2 - m = (n+1)^2 - (n+1) = n^2 + n;$$

$$\text{Divisões: } \sum_{i=1}^m (m-1) = m^2 - m = (n+1)^2 - (n+1) = n^2 + n.$$

Estes resultados estão resumidos na Tabela 1.6.

Tabela 1.6 Complexidade da interpolação de Lagrange via Expressão 1.

(n : grau do polinômio interpolador.)

| Operações | Complexidade |
|----------------|-----------------|
| adições | $2n^2 + 3n + 1$ |
| multiplicações | $n^2 + n$ |
| divisões | $n^2 + n$ |

O polinômio de Lagrange também pode ser expandido de modo a resultar a Expressão 2

$$\begin{aligned} L_n(x) &= y_0 \times \frac{(x - x_1) \times (x - x_2) \times \dots \times (x - x_n)}{(x_0 - x_1) \times (x_0 - x_2) \times \dots \times (x_0 - x_n)} \\ &+ y_1 \times \frac{(x - x_0) \times (x - x_2) \times \dots \times (x - x_n)}{(x_1 - x_0) \times (x_1 - x_2) \times \dots \times (x_1 - x_n)} \\ &\dots + y_n \times \frac{(x - x_0) \times (x - x_1) \times \dots \times (x - x_{n-1})}{(x_n - x_0) \times (x_n - x_1) \times \dots \times (x_n - x_{n-1})}. \end{aligned}$$

O algoritmo desta expressão é mostrado na Figura 3.2, na página 132, e a sua complexidade computacional é compilada na Tabela 3.2, na página 132. Comparando os resultados das Tabelas 1.6 e 3.2, nota-se que o número de adições é o mesmo e o de multiplicações é da mesma ordem (n^2). No entanto, o número de divisões utilizado pela Expressão 2 é de uma ordem de grandeza a menos.

O polinômio de Lagrange serve para exemplificar que uma mesma notação matemática pode resultar em algoritmos de diferentes complexidades. Isto deve ser lembrado ao se elaborar um algoritmo.

1.5 Implementação de algoritmos

Uma das quatro etapas na solução de um problema é a solução numérica, a qual pode ser subdividida em três fases: elaboração do algoritmo, codificação do programa e processamento do programa. Nas seções anteriores foi proposta uma notação algorítmica e mostrado como elaborar um algoritmo a partir de uma formulação matemática. A próxima fase é a codificação do programa na linguagem escolhida. Nesta seção serão mostrados três exemplos de implementações dos algoritmos nas linguagens de programação FORTRAN, Pascal e MATLAB descritas nos Apêndices A, B e C, respectivamente.

Exemplo 1.21 Implementar em FORTRAN o algoritmo do Exemplo 1.10 usando variável de ponto flutuante de 8 bytes.

```
program PreMaq
c     Programa para determinar a precisao da maquina
c     para variavel real de 8 bytes
real*8 Epsilon
Epsilon = 1.0d0
10 continue
    Epsilon = Epsilon / 2.0d0
    if( Epsilon+1.0d0.ne.1.0d0 ) go to 10
    write(*,16) Epsilon
    stop
16 format('Precisao da maquina:',1pd15.8)
end
```

A execução do programa fornece o resultado

Precisao da maquina: 1.11022302E-16

que é igual a 2^{-53} . Se for utilizada variável real de 4 bytes, o resultado será

Precisao da maquina: 5.96046448E-08

sendo igual a 2^{-24} . Estes resultados mostram que se qualquer número menor ou igual à precisão da máquina for somado a 1, o resultado será 1. ■

Exemplo 1.22 Implementar em Pascal o algoritmo da Figura 1.1.

```
program Media_desvio;
type vetor = array[1..100] of real;
var n: integer;
    Media, DesvioPadrao: real;
    x: vetor;
{   Calculo da media aritmetica e desvio padrao }
procedure MediaDesvioPadrao(n:integer;x:vetor;var Media,DesvioPadrao:real);
var i: integer;
    Soma, Soma2: real;
begin
    Soma := 0;
    Soma2 := 0;
    for i := 1 to n do begin
        Soma := Soma + x[i];
        Soma2 := Soma2 + sqr(x[i]);
    end;
    Media := Soma / n;
    DesvioPadrao := sqrt((Soma2-sqr(Soma)/n)/(n-1));
end; { procedure MediaDesvioPadrao }
begin
var i: integer;
writeln('Numero de elementos: ');
readln(n);
writeln('Elementos: ');
for i := 1 to n do
    read(x[i]);
MediaDesvioPadrao(n, x, Media, DesvioPadrao);
writeln('Media =',Media:10:5,' Desvio padrao =',DesvioPadrao:10:5);
end.
```

Exemplo 1.23 Implementar em MATLAB o algoritmo da Figura 1.3.

```
% Calculo de pi com uma dada precisao
function Pi = Calcular_pi(Precisao)
Soma = 1;
Sinal = -1;
Denominador = 3;
while 1
    Soma = Soma + Sinal / Denominador;
    if 1 / Denominador < Precisao
        break
    end
    Sinal = -Sinal;
    Denominador = Denominador + 2;
end
Pi = 4 * Soma;
```

1.6 Tipos de erros

Durante as etapas de solução de um problema, surgem erros de várias fontes que podem alterar profundamente os resultados obtidos. É de importância fundamental conhecer as causas desses erros para minimizar as suas consequências.

Erro de truncamento

O erro de truncamento é devido à aproximação de uma fórmula por outra. É sabido que, para avaliar uma função matemática no computador, somente as operações aritméticas e lógicas podem ser requeridas, por serem as operações que ele é capaz de efetuar. Por exemplo, para avaliar $f(x) = \sin(x)$ esta tem que ser aproximada por uma série, tal como

$$\sin(x) = \sum_{n=0}^{\infty} (-1)^n \frac{x^{2n+1}}{(2n+1)!} = x - \frac{x^3}{6} + \frac{x^5}{120} - \frac{x^7}{5040} + \dots, \quad 0 \leq x \leq \frac{\pi}{4}.$$

À medida que n aumenta, mais o valor da série se aproxima do valor real. A Tabela 1.7 mostra a diferença entre o valor obtido pela série de $\sin(x)$ e um valor mais exato, para n até 2, 3 e 4. Quando n aumenta, o erro de truncamento diminui, ficando claro que estes erros são devidos aos truncamentos da série.

Tabela 1.7 Efeito do erro de truncamento no cálculo de $\sin(x)$.

| x | $t = 2$ | $t = 3$ | $t = 4$ |
|----------|----------------------|----------------------|-----------------------|
| 0 | 0 | 0 | 0 |
| $\pi/16$ | $2,4 \times 10^{-6}$ | $2,2 \times 10^{-9}$ | $1,2 \times 10^{-12}$ |
| $\pi/8$ | $7,8 \times 10^{-5}$ | $2,9 \times 10^{-7}$ | $6,1 \times 10^{-10}$ |
| $\pi/6$ | $3,3 \times 10^{-4}$ | $2,1 \times 10^{-6}$ | $8,1 \times 10^{-9}$ |
| $\pi/4$ | $2,5 \times 10^{-3}$ | $3,6 \times 10^{-5}$ | $3,1 \times 10^{-7}$ |

Erro absoluto e relativo

O erro cometido na computação de um resultado pode ser medido de duas maneiras. A primeira é o erro absoluto definido como

$$\text{erro absoluto} = \text{valor real} - \text{valor aproximado}.$$

O tamanho do erro absoluto é mais grave quando o valor verdadeiro for pequeno. Por exemplo, $1711,321 \pm 0,030$ é exato com cinco dígitos significativos, enquanto $0,001 \pm 0,030$ tem pouco significado. A outra maneira é o erro relativo definido como

$$\text{erro relativo} = \frac{\text{valor real} - \text{valor aproximado}}{\text{valor real}},$$

sendo indefinido para valor real nulo. A sua vantagem sobre o erro absoluto é a independência da magnitude dos valores.

Erro na modelagem

Na etapa da modelagem matemática de um problema real, muitas vezes se faz necessário o uso de dados obtidos por medidas experimentais. Pode ocorrer tanto uma modelagem incorreta na qual a expressão matemática não reflete perfeitamente o fenômeno físico quanto os dados terem sido obtidos com pouca exatidão. Nesses casos, é necessária a realização de testes para verificar o quanto os resultados são sensíveis às alterações dos dados fornecidos.

Mudanças grandes nos resultados devido a pequenas variações nos dados são sintomas de um malcondicionamento do modelo proposto, sendo uma nova modelagem do fenômeno a tentativa de cura do problema.

Erro grosso

A possibilidade de um computador cometer um erro é muito pequena; no entanto, podem ser cometidos erros na elaboração do algoritmo, na sua implementação e mesmo na digitação de dados. Executar o programa, cujo resultado seja conhecido, ajuda a remover erros, mas demonstra, apenas, que o programa está correto para aquela massa de dados! A solução seria elaborar uma *prova de correção de programa* que é uma tarefa não trivial.

Erro de arredondamento

Um número decimal qualquer, por exemplo $0,4_{10}$ (0,4 na base 10), não pode ser representado exatamente em um computador porque ele tem que ser convertido para a base 2 e armazenado em um número finito de bits. O erro causado por esta imperfeição na representação de um número é chamado de erro de arredondamento. As causas e consequências desse tipo de erro serão abordadas introdutoriamente na Seção 1.7.

Para uma análise mais detalhada sobre os efeitos do erro de arredondamento deve ser consultado um texto mais específico, como, por exemplo, um livro clássico de James Hardy Wilkinson [39].

1.7 Aritmética de ponto flutuante

Para uma melhor compreensão das causas do erro de arredondamento, se faz necessário conhecer como os números são armazenados em um computador. Um número pode ser representado com ponto fixo, por exemplo, 12,34 ou com ponto flutuante² 0,1234×10². Neste texto, será abordada apenas a aritmética de ponto flutuante. A forma geral de representação de um número de ponto flutuante é similar à notação científica

$$\pm.d_1d_2d_3\dots d_p \times B^e,$$

onde os d_i 's são os dígitos da parte fracionária, tais que $0 \leq d_i \leq B-1$, $d_1 \neq 0$, B é o valor da base (geralmente 2, 10 ou 16), p é o número de dígitos e e é um expoente inteiro. Deste modo, um número de ponto flutuante tem três partes: o sinal, a parte fracionária chamada de significando ou mantissa e o expoente. Estas três partes têm um comprimento total fixo que depende do computador e do tipo de número: precisão simples, dupla ou estendida, conforme será visto mais adiante. Com o intuito de mostrar a representação de um número de ponto flutuante, seja um computador hipotético com dois dígitos ($p=2$), base $B=2$ e expoente na faixa $-1 \leq e \leq 2$. Como o número é normalizado, isto é, $d_1 \neq 0$, eles serão da forma

$$\pm.10_2 \times 2^e \text{ ou } \pm.11_2 \times 2^e, e = -1, \dots, 2.$$

Considerando a conversão de binário para decimal de um número menor que 1,

$$.10_2 = 1 \times 2^{-1} + 0 \times 2^{-2} = 1/2 \text{ e}$$

$$.11_2 = 1 \times 2^{-1} + 1 \times 2^{-2} = 3/4,$$

então os únicos números positivos representáveis neste computador são

| | |
|---|---|
| $.10_2 \times 2^{-1} = 1/2 \times 2^{-1} = 1/4$ | $.11_2 \times 2^{-1} = 3/4 \times 2^{-1} = 3/8$ |
| $.10_2 \times 2^0 = 1/2 \times 1 = 1/2$ | $.11_2 \times 2^0 = 3/4 \times 1 = 3/4$ |
| $.10_2 \times 2^1 = 1/2 \times 2 = 1$ | $.11_2 \times 2^1 = 3/4 \times 2 = 3/2$ |
| $.10_2 \times 2^2 = 1/2 \times 4 = 2$ | $.11_2 \times 2^2 = 3/4 \times 4 = 3$ |

O zero é representado de uma forma especial: todos os dígitos d_i do significando e do expoente são nulos. O mais importante a ser observado acerca dos números de ponto flutuante é que eles são discretos e não contínuos como um *número real* definido na Matemática, conforme pode ser visto na Figura 1.10.

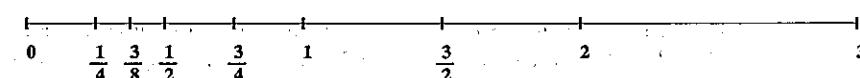


Figura 1.10 Valores discretos dos números de ponto flutuante.

²Os nomes corretos desses formatos em português deveriam ser vírgula fixa e vírgula flutuante, pois em nossa língua é a vírgula que separa a parte inteira de um número da sua parte decimal. No entanto, como a terminologia ponto fixo e ponto flutuante já está consagrada na literatura, optamos por adotá-la.

O conceito de sempre existir um número real entre dois números reais quaisquer não é válido para os números de ponto flutuante. A falha deste conceito tem consequência desastrosa, conforme será mostrado a seguir. Considere a representação binária

$$0,6_{10} = 0,100110011001\ldots_2 \text{ e } 0,7_{10} = 0,1011001100110\ldots_2.$$

Se estes dois números forem armazenados naquele computador hipotético, eles serão representados igualmente como $.10_2 \times 2^0$. Isto significa que tanto $0,6_{10}$ quanto $0,7_{10}$ serão vistos como $0,5_{10}$ por aquele computador. Esta é uma grande causa de erro de arredondamento nos processos numéricos.

A forma de representação de um número de ponto flutuante depende do fabricante do computador, portanto um mesmo programa implementado em computadores que utilizam formatos diferentes pode fornecer resultados diferentes.

O formato utilizado pela maioria dos microcomputadores e estações de trabalho é aquele proposto pelo IEEE (Institute of Electrical and Electronics Engineers), o qual é mostrado na Tabela 1.8 [25].

Tabela 1.8 Formato IEEE de ponto flutuante.

| Propriedade | Precisão | | |
|-------------------|--------------------------------|---------------------------------|----------------------------------|
| | Simples | Dupla | Estendida |
| comprimento total | 32 | 64 | 80 |
| bits na mantissa | 23 | 52 | 64 |
| bits no expoente | 8 | 11 | 15 |
| base | 2 | 2 | 2 |
| expoente máximo | 127 | 1023 | 16383 |
| expoente mínimo | -126 | -1022 | -16382 |
| maior número | $\approx 3,40 \times 10^{38}$ | $\approx 1,80 \times 10^{308}$ | $\approx 1,19 \times 10^{4932}$ |
| menor número | $\approx 1,18 \times 10^{-38}$ | $\approx 2,23 \times 10^{-308}$ | $\approx 3,36 \times 10^{-4932}$ |
| dígitos decimais | 7 | 16 | 19 |

Se uma operação aritmética resultar em um número que seja maior em módulo que o maior número representável, ocorrerá um *overflow*. Se, por outro lado, resultar em um número que em módulo seja menor que o menor número representável diferente de zero, ocorrerá um *underflow*. O modo de tratar *overflow* e *underflow* dependerá do compilador utilizado para gerar o programa.

Será mostrada, a seguir, a precisão das operações numéricas envolvendo números de ponto flutuante. Para tal, será utilizado um outro computador hipotético com dois dígitos ($p = 2$), base $B = 10$ para facilitar o entendimento e expoente $e = -5, \dots, 5; \pm d_1 d_2 \times 10^e$.

Quando dois números são somados ou subtraídos, os dígitos do número de expoente menor devem ser deslocados de modo a alinhar as casas decimais. O resultado é, então, arredondado para dois dígitos para caber na mantissa de tamanho $p = 2$. Isto feito, o expoente é ajustado de forma a normalizar a mantissa ($d_1 \neq 0$).

Exemplo 1.24 Somar 4,32 e 0,064.

Os números são armazenados no formato especificado, as casas decimais são alinhadas e a operação de adição é efetuada. O resultado é arredondado para dois dígitos

$$\begin{aligned} 4,32 + 0,064 &= .43 \times 10^1 + .064 \times 10^{-1} = & .43 & \times 10^1 \\ &+ .0064 & & \times 10^1 \\ &= .4364 & & \times 10^1 \\ &\rightarrow .44 & & \times 10^1. \end{aligned}$$

O resultado da adição foi 4,4 em vez de 4,384.

Exemplo 1.25 Subtrair 371 de 372.

Os números são armazenados no formato especificado, resultando em um mesmo valor no caso e a operação de subtração é efetuada. O resultado é convertido para zero

$$\begin{aligned} 372 - 371 &= .37 \times 10^3 - .37 \times 10^3 = & .37 & \times 10^3 \\ &- .37 & & \times 10^3 \\ &= .00 & & \times 10^3 \\ &\rightarrow .00 & & \times 10^0. \end{aligned}$$

A subtração deu 0 em vez de 1. A perda de precisão quando dois números aproximadamente iguais são subtraídos é a maior fonte de erro nas operações de ponto flutuante.

Exemplo 1.26 Somar 691 e 2,71.

Os números são armazenados no formato especificado, as casas decimais são alinhadas e a operação de adição é efetuada. O resultado é arredondado para dois dígitos

$$\begin{aligned} 691 + 2,71 &= .69 \times 10^3 + .27 \times 10^1 = & .69 & \times 10^3 \\ &+ .0027 & & \times 10^3 \\ &= .6927 & & \times 10^3 \\ &\rightarrow .69 & & \times 10^3. \end{aligned}$$

A adição resultou em 690 em vez de 693,71. O deslocamento das casas decimais de 2,71 causou uma perda total dos seus dígitos durante a operação.

Exemplo 1.27 Multiplicar 1234 por 0,016.

Os números são armazenados no formato definido e a operação de multiplicação é efetuada utilizando-se $2p = 4$ dígitos na mantissa. O resultado é arredondado para dois dígitos e normalizado

$$\begin{aligned} 1234 \times 0,016 &= .12 \times 10^4 \times .16 \times 10^{-1} = & .12 & \times 10^4 \\ &\times .16 & & \times 10^{-1} \\ &= .0192 & & \times 10^3 \\ &\rightarrow .19 & & \times 10^2. \end{aligned}$$

O resultado da multiplicação foi 19 em vez de 19,744.

Exemplo 1.28 Multiplicar 875 por 3172.

Os números são armazenados no formato indicado e a operação de multiplicação é efetuada utilizando-se $2p = 4$ dígitos. O resultado é arredondado, normalizado, e, como o expoente $e = 7 > 5$ (máximo definido para este computador hipotético), então ocorre um *overflow*

$$\begin{aligned} 875 \times 3172 &= .88 \times 10^3 \times .32 \times 10^4 = & .88 & \times 10^3 \\ &\quad \times .32 & \times 10^4 \\ &= .2816 & \times 10^7 \\ &\rightarrow \text{overflow}. \end{aligned}$$

A multiplicação resultou em um valor maior do que este computador pode representar. ■

Exemplo 1.29 Dividir 0,00183 por 492.

Os números são armazenados no formato especificado e a operação de divisão é efetuada utilizando $2p = 4$ dígitos na mantissa. O resultado é arredondado para dois dígitos e normalizado

$$\begin{aligned} 0,00183 \div 492 &= .18 \times 10^{-2} \div .49 \times 10^3 = & .18 & \times 10^{-2} \\ &\div .49 & \times 10^3 \\ &= .3673 & \times 10^{-5} \\ &\rightarrow .37 & \times 10^{-5}. \end{aligned}$$

O erro relativo desse resultado foi de aproximadamente 0,52%. ■

Exemplo 1.30 Dividir 0,0064 por 7312.

Os números são armazenados no formato definido e a divisão é efetuada utilizando-se $2p = 4$ dígitos na mantissa. O resultado é arredondado, normalizado, e, sendo o expoente $e = -6 < -5$, então ocorre um *underflow*.

$$\begin{aligned} 0,0064 \div 7312 &= .64 \times 10^{-2} \div .73 \times 10^4 = & .64 & \times 10^{-2} \\ &\div .73 & \times 10^4 \\ &= .8767 & \times 10^{-6} \\ &\rightarrow \text{underflow}. \end{aligned}$$

O resultado da divisão foi um valor menor que este computador pode armazenar, sem considerar o zero, que tem uma representação especial. ■

Além dos erros descritos na Seção 1.6, uma outra causa de erros quando se usa um computador é devida à conversão de base. Geralmente, um número é fornecido ao computador na base 10 e, no entanto, ele é armazenado na base 2.

Quando os números forem inteiros, a representação binária é exata, como, por exemplo, $44_{10} = 101100_2$. No entanto, um número com decimais pode resultar em um número binário com infinitos dígitos ($0,4_{10} = 0,01100110\dots_2$) que têm que ser arredondados para o armazenamento em um formato de ponto flutuante.

1.8 Exercícios

Seção 1.1

1.1. Ler o Exemplo de aplicação 2.10.1 sobre tensões em circuito elétrico, na página 117.

1.2. Ler o Exemplo de aplicação 5.8.1 sobre distribuição de probabilidade, na página 264.

1.3. Ler o Exemplo de aplicação 6.7.1 sobre juros de financiamento, na página 316.

1.4. Definir um problema e fazer a modelagem matemática que envolva a solução de um sistema de equações lineares.

1.5. Escolher um problema e modelar de modo a recair no cálculo da raiz de uma equação.

Seção 1.2

1.6. Pela regra de De Morgan para expressões lógicas

$\neg(a \wedge b) = \neg(a) \vee \neg(b)$,

$\neg(a \vee b) = \neg(a) \wedge \neg(b)$.

Com base nas regras, simplificar as expressões

$\neg(\neg(c > 5) \wedge j = 1)$ e

$\neg(d \geq 0 \vee m \neq 3)$.

1.7. Qual a diferença entre a estrutura condicional simples e a composta?

1.8. Propor um algoritmo que faça uso de uma estrutura de repetição com um número indefinido de repetições.

1.9. Elaborar um algoritmo que utilize uma estrutura de repetição com um número definido de repetições.

1.10. Qual a diferença entre os dois comandos: *interrompa* e *abandone*?

Seção 1.3

Passar as notações matemáticas dadas a seguir para uma notação algorítmica

1.11. O traço de uma matriz A é a soma dos elementos da sua diagonal principal

$$\text{traço}(A) = \sum_{i=1}^n a_{ii}.$$

1.12. O produto externo entre um vetor x de tamanho n e outro vetor y de tamanho m resulta em uma matriz $M = xy^T$ de dimensão $(n \times m)$ tal que

$$m_{ij} = x_i y_j \begin{cases} i = 1, 2, \dots, n, \\ j = 1, 2, \dots, m, \end{cases}$$

ou

$$M = \begin{bmatrix} x_1 y_1 & \cdots & x_1 y_m \\ \vdots & \cdots & \vdots \\ x_n y_1 & \cdots & x_n y_m \end{bmatrix}.$$

1.13. A norma matricial de soma máxima de coluna é dada por

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}|.$$

1.14. A norma de Frobenius é definida por

$$\|A\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^n |a_{ij}|^2}.$$

1.15. O produto de uma matriz A de dimensão $(n \times p)$ por outra matriz B de dimensão $(p \times m)$ é a matriz $C = AB$ $(n \times m)$, sendo que

$$c_{ij} = \sum_{k=1}^p a_{ik} b_{kj} \begin{cases} i = 1, 2, \dots, n \\ j = 1, 2, \dots, m \end{cases}$$

Seção 1.4

Fazer a análise de complexidade dos algoritmos dados a seguir em relação a operação de adição.

1.16. Algoritmo do Exercício 1.11.

1.17. Algoritmo do Exercício 1.14.

1.18. Algoritmo do Exercício 1.15, para o produto de duas matrizes quadradas de ordem n .

2.1.1 Alguns tipos de matrizes

Matrizes com determinados formatos e elementos aparecem com tanta freqüência que merecem nomes especiais.

Coluna e linha

Uma matriz de tamanho $n \times 1$ é dita uma matriz coluna ou um vetor coluna de tamanho n . Por outro lado, uma matriz de tamanho $1 \times m$ é uma matriz linha ou um vetor linha de tamanho m .

Nula

Se todos os elementos de uma matriz A forem iguais a zero, então ela é uma matriz nula, ou seja, $a_{ij} = 0, \forall i, j$.

Diagonal

Uma matriz quadrada é dita diagonal se todos os elementos fora da diagonal principal forem nulos, isto é, se $d_{ij} = 0, \forall i \neq j$.

Identidade

Matriz identidade é uma matriz diagonal com os elementos da diagonal principal iguais a 1, ou seja, $e_{ij} = 1, \forall i = j$ e $e_{ij} = 0, \forall i \neq j$.

Triangular

Existem dois tipos de matrizes triangulares: inferior e superior. Uma matriz triangular inferior é uma matriz com todos os elementos acima da diagonal principal iguais a zero, isto é, $b_{ij} = 0, \forall i < j$

$$B = \begin{bmatrix} b_{11} & 0 & 0 & \cdots & 0 \\ b_{21} & b_{22} & 0 & \cdots & 0 \\ b_{31} & b_{32} & b_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & b_{m3} & \cdots & b_{mm} \end{bmatrix}$$

De modo similar, uma matriz triangular superior é uma matriz com todos os elementos abaixo da diagonal principal sendo nulos, ou seja, $c_{ij} = 0, \forall i > j$

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} & \cdots & c_{1m} \\ 0 & c_{22} & c_{23} & \cdots & c_{2m} \\ 0 & 0 & c_{33} & \cdots & c_{3m} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & c_{mm} \end{bmatrix}$$

Densa e esparsa

Uma matriz é densa quando a maior parte de seus elementos forem não nulos e é esparsa quando a maioria for igual a zero. Muitas vezes, as matrizes provenientes da solução de problemas reais são esparsas e de alta ordem, existindo esquemas especiais para representá-las e manipulá-las [14].

Simétrica

Uma matriz é dita simétrica se houver uma simetria dos elementos em relação à diagonal principal, isto é, $m_{ij} = m_{ji}, \forall i, j$. Deste modo, uma matriz é simétrica se ela for igual à sua transposta, ou seja, $M = M^T$ (ver Seção 2.1.2).

Exemplo 2.1 A matriz simétrica M e sua transposta M^T

$$M = \begin{bmatrix} 3 & 1 & 0 & 4 \\ 1 & 9 & 5 & 2 \\ 0 & 5 & 8 & 6 \\ 4 & 2 & 6 & 7 \end{bmatrix} \quad \text{e} \quad M^T = \begin{bmatrix} 3 & 1 & 0 & 4 \\ 1 & 9 & 5 & 2 \\ 0 & 5 & 8 & 6 \\ 4 & 2 & 6 & 7 \end{bmatrix}$$

2.1.2 Operações matriciais

Operações envolvendo matrizes são de grande importância na solução de problemas. Algumas delas são descritas a seguir.

Transposição

A transposta de uma matriz A , representada por A^T , é uma matriz obtida trocando-se as suas linhas pelas colunas, de modo que a linha i torna-se a coluna i e a coluna j transforma-se na linha j .

Exemplo 2.2 A transposição da matriz A

$$A = \begin{bmatrix} 5 & 2 & 0 \\ 6 & 9 & 1 \\ 3 & 4 & 2 \\ 7 & 0 & 8 \\ 1 & 5 & 6 \end{bmatrix} \quad \text{e} \quad A^T = \begin{bmatrix} 5 & 6 & 3 & 7 & 1 \\ 2 & 9 & 4 & 0 & 5 \\ 0 & 1 & 2 & 8 & 6 \end{bmatrix}$$

Adição e subtração

Se A e B forem matrizes de dimensão $n \times m$, então $C = A + B$ também será $n \times m$, tal que $c_{ij} = a_{ij} + b_{ij}, \forall i, j$. É claro que apenas matrizes de mesmo tamanho podem ser somadas ou subtraídas.

Exemplo 2.3 As operações de adição e subtração das matrizes A e B

$$A = \begin{bmatrix} 8 & 6 \\ 1 & 4 \\ 5 & 7 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 2 \\ 0 & 3 \\ 2 & 1 \end{bmatrix}, \quad C = A + B = \begin{bmatrix} 9 & 8 \\ 1 & 7 \\ 7 & 8 \end{bmatrix} \quad \text{e} \quad D = A - B = \begin{bmatrix} 7 & 4 \\ 1 & 1 \\ 3 & 6 \end{bmatrix}$$

Multiplicação

O produto de uma matriz A de dimensão $n \times m$ por um escalar k resulta em uma matriz $B = kA$ de mesma dimensão $n \times m$, tal que $b_{ij} = ka_{ij}, \forall i, j$.

Exemplo 2.4 O produto de matriz por escalar

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \text{ e } B = 2A = \begin{bmatrix} 2 & 4 & 6 \\ 8 & 10 & 12 \end{bmatrix}.$$

O produto de uma matriz A ($n \times m$) por um vetor v ($m \times 1$) resulta em um vetor x ($n \times 1$), de forma que

$$x_i = \sum_{j=1}^m a_{ij}v_j, \quad i = 1, 2, \dots, n.$$

Exemplo 2.5 O produto de matriz por vetor

$$A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{bmatrix}, \quad v = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \rightarrow x = Av = \begin{bmatrix} 5 \\ 11 \\ 17 \end{bmatrix}.$$

O produto de uma matriz A ($n \times p$) por uma matriz B ($p \times m$) é uma matriz $C = AB$ ($n \times m$), tal que

$$c_{ij} = \sum_{k=1}^p a_{ik}b_{kj}, \quad i = 1, 2, \dots, n \text{ e } j = 1, 2, \dots, m.$$

Exemplo 2.6 O produto de matriz por matriz

$$A = \begin{bmatrix} 2 & 1 & 0 \\ 3 & 5 & 6 \end{bmatrix}, \quad B = \begin{bmatrix} 1 & 6 \\ 4 & 0 \\ 3 & 5 \end{bmatrix} \rightarrow C = AB = \begin{bmatrix} 6 & 12 \\ 41 & 48 \end{bmatrix}.$$

O elemento c_{ij} é obtido pela soma dos produtos dos elementos da linha i de A pelos correspondentes elementos da coluna j de B . Desse modo, duas matrizes podem ser multiplicadas se, e somente se, o número de colunas da primeira for igual ao número de linhas da segunda.

A pré-multiplicação de uma matriz A por uma matriz diagonal D tem o efeito de multiplicar cada linha de A pelo correspondente elemento de D

$$\begin{bmatrix} d_{11} & & \\ & d_{22} & \\ & & d_{33} \end{bmatrix} \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} d_{11}a_{11} & d_{11}a_{12} & d_{11}a_{13} \\ d_{22}a_{21} & d_{22}a_{22} & d_{22}a_{23} \\ d_{33}a_{31} & d_{33}a_{32} & d_{33}a_{33} \end{bmatrix}.$$

Por sua vez, a pós-multiplicação de uma matriz A por uma matriz diagonal D faz com que cada coluna de A seja multiplicada pelo correspondente elemento de D

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} d_{11} & & \\ & d_{22} & \\ & & d_{33} \end{bmatrix} = \begin{bmatrix} a_{11}d_{11} & a_{12}d_{22} & a_{13}d_{33} \\ a_{21}d_{11} & a_{22}d_{22} & a_{23}d_{33} \\ a_{31}d_{11} & a_{32}d_{22} & a_{33}d_{33} \end{bmatrix}.$$

Pelas expressões anteriores, pode-se verificar que a multiplicação matricial não segue a lei comutativa, ou seja, geralmente $AB \neq BA$.

Produto interno e externo

O produto escalar ou interno entre um vetor x ($n \times 1$) e um vetor y ($n \times 1$) é um escalar k obtido por

$$k = x^T y = x_1y_1 + x_2y_2 + x_3y_3 + \cdots + x_ny_n = \sum_{i=1}^n x_iy_i.$$

Por outro lado, o produto externo entre um vetor x ($n \times 1$) e outro vetor y ($m \times 1$) resulta em uma matriz M ($n \times m$), de forma que

$$M = xy^T, \text{ com } m_{ij} = x_iy_j, \quad i = 1, 2, \dots, n \text{ e } j = 1, 2, \dots, m,$$

isto é,

$$M = \begin{bmatrix} x_1y_1 & \cdots & x_1y_m \\ \vdots & \ddots & \vdots \\ x_ny_1 & \cdots & x_ny_m \end{bmatrix}.$$

Exemplo 2.7 O produto interno e externo de dois vetores

$$x = \begin{bmatrix} 5 \\ -1 \\ 2 \end{bmatrix}, \quad y = \begin{bmatrix} 1 \\ 3 \\ 4 \end{bmatrix} \rightarrow k = x^T y = 10 \text{ e } M = xy^T = \begin{bmatrix} 5 & 15 & 20 \\ -1 & -3 & -4 \\ 2 & 6 & 8 \end{bmatrix}.$$

Determinante

Uma matriz quadrada A de ordem n tem um número associado denominado determinante, cujo valor pode ser obtido pela fórmula de recorrência

$$\det(A) = a_{11} \det(M_{11}) - a_{12} \det(M_{12}) + \cdots + (-1)^{n+1} a_{1n} \det(M_{1n}),$$

onde M_{ij} é a matriz de ordem $n - 1$ resultante da remoção da linha i e coluna j de A e sendo o determinante de uma matriz (1×1) igual a esse único elemento. Particularmente,

$$A = [a_{11}] \rightarrow \det(A) = a_{11},$$

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \rightarrow \det(A) = a_{11}a_{22} - a_{21}a_{12} \text{ e}$$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \rightarrow$$

$$\det(A) = a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{21}a_{33} - a_{31}a_{23}) + a_{13}(a_{21}a_{32} - a_{31}a_{22}). \quad (2.1)$$

Um modo mais eficiente de calcular o determinante de uma matriz será mostrado mais adiante. Uma matriz A com $\det(A) = 0$ é dita singular e com $\det(A) \neq 0$ é não singular.

Exemplo 2.8 O determinante de uma matriz de ordem 2

$$A = \begin{bmatrix} 2 & 1 \\ 4 & 5 \end{bmatrix} \rightarrow \det(A) = 2 \times 5 - 4 \times 1 = 6.$$

Posto

Uma seqüência de vetores $\{v_1, v_2, \dots, v_p\}$ é dita linearmente dependente se existirem escalares $\alpha_1, \alpha_2, \dots, \alpha_p$, não todos nulos, tais que

$$\alpha_1 v_1 + \alpha_2 v_2 + \dots + \alpha_p v_p = 0.$$

Neste caso, diz-se também que os vetores v_1, v_2, \dots, v_p são linearmente dependentes. Se a igualdade acima só se verificar com os α_i , $i = 1, 2, \dots, p$ iguais a zero, diz-se que os vetores v_1, v_2, \dots, v_p são linearmente independentes. Agora, pode-se definir posto de uma matriz A ($m \times n$) como o número máximo de vetores linhas ou de vetores colunas de A que são linearmente independentes, sendo $\text{posto}(A) \leq \min(m, n)$.

Exemplo 2.9 Seja a matriz

$$A = \begin{bmatrix} 2 & -3 & 4 & 1 & 0 & -2 \\ 7 & -2 & 4 & 3 & 3 & -1 \\ 5 & 1 & 0 & 2 & 3 & 1 \\ -1 & -7 & 8 & 0 & -3 & -5 \\ 1 & 2 & 2 & -1 & 5 & 2 \end{bmatrix}.$$

Considerando que as linhas 2 e 4 são obtidas pela combinação linear das linhas 1 e 3, pois

$$\text{linha } 2 = \text{linha } 1 + \text{linha } 3$$

$$\text{linha } 4 = 2(\text{linha } 1) - \text{linha } 3$$

e como as linhas 1, 3 e 5 são linearmente independentes, então tem-se $\text{posto}(A) = 3$.

Traço

O traço de uma matriz quadrada A é a soma dos elementos da sua diagonal principal

$$\text{traço}(A) = \sum_{i=1}^n a_{ii}.$$

$$\text{Exemplo 2.10} \quad \text{A matriz } M = \begin{bmatrix} 5 & -1 & 3 \\ 2 & 3 & 1 \\ 0 & 8 & 9 \end{bmatrix}$$

$$\text{tem } \text{traço}(M) = 5 + 3 + 9 = 17.$$

Inversa

A inversa de uma matriz quadrada A de ordem n é representada por A^{-1} e definida tal que

$$AA^{-1} = A^{-1}A = I_n,$$

onde I_n é a matriz identidade de ordem n . Assim, a lei comutativa existe para o produto de uma matriz por sua inversa. Um método para calcular a matriz inversa será apresentado na Seção 2.7.2.

Exemplo 2.11 Uma matriz A e sua inversa A^{-1}

$$A = \begin{bmatrix} 1 & 0 & 2 & 1 \\ 1 & 1 & 1 & 2 \\ -1 & -1 & 0 & -2 \\ 2 & 0 & 4 & 3 \end{bmatrix}, \quad A^{-1} = \begin{bmatrix} 3 & -2 & -2 & -1 \\ 1 & 2 & 1 & -1 \\ 0 & 1 & 1 & 0 \\ -2 & 0 & 0 & 1 \end{bmatrix} \text{ e } AA^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Operações com transposta e inversa

$$\text{a)} (A^T)^T = A.$$

$$\text{b)} (A^{-1})^{-1} = A.$$

$$\text{c)} (A^{-1})^T = (A^T)^{-1} = A^{-T}.$$

$$\text{d)} \text{Se } A = BCD, \text{ então } A^T = D^T C^T B^T \text{ e } A^{-1} = D^{-1} C^{-1} B^{-1}.$$

$$\text{e)} (A + B)^T = A^T + B^T.$$

$$\text{f)} (A + B)^{-1} \neq A^{-1} + B^{-1}.$$

2.1.3 Noções sobre autovalores e autovetores

Seja a matriz

$$A = \begin{bmatrix} 10 & -4 \\ 12 & -4 \end{bmatrix}$$

que apresenta a propriedade

$$\begin{bmatrix} 10 & -4 \\ 12 & -4 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} = 2 \begin{bmatrix} 1 \\ 2 \end{bmatrix}.$$

É dito, então, que a matriz A possui um autovalor $\lambda = 2$ e um correspondente autovetor $v = [1 \ 2]^T$. O mesmo também é verdade para $\lambda = 4$ e $v = [2 \ 3]^T$

$$\begin{bmatrix} 10 & -4 \\ 12 & -4 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix} = 4 \begin{bmatrix} 2 \\ 3 \end{bmatrix}.$$

Desta forma, pode-se definir a relação fundamental de uma matriz A de ordem n com seus n autovalores λ e os correspondentes autovetores v

$$Av = \lambda v. \quad (2.2)$$

O problema do autovalor consiste em encontrar a solução não trivial (ou não nula) do sistema homogêneo

$$(A - \lambda I)v = 0.$$

Pelo Teorema 2.1 [29] e sabendo que uma matriz singular tem determinante nulo, então

$$\det(A - \lambda I) = 0. \quad (2.3)$$

Teorema 2.1 Se M for uma matriz de ordem n , então o sistema homogêneo $My = 0$ tem solução não trivial se, e somente se, M for singular.

Exemplo 2.12 Para a matriz A mostrada acima

$$\det(A - \lambda I) = \det \begin{pmatrix} 10 - \lambda & -4 \\ 12 & -4 - \lambda \end{pmatrix} = 0, \text{ isto é,}$$

$$(10 - \lambda)(-4 - \lambda) - 12(-4) = 0 \rightarrow \lambda^2 - 6\lambda + 8 = (\lambda - 2)(\lambda - 4) = 0.$$

Portanto, os valores de λ para os quais $\det(A - \lambda I) = 0$ são os chamados autovalores $\lambda_1 = 2$ e $\lambda_2 = 4$ daquela matriz A . É fácil ver que para $\lambda = 2$

$$\det(A - \lambda I) = \det(A - 2I) = \det \begin{pmatrix} 10 - 2 & -4 \\ 12 & -4 - 2 \end{pmatrix}$$

$$\det(A - 2I) = 8 \times -6 - 12 \times -4 = 0.$$

Também para $\lambda = 4$

$$\det(A - \lambda I) = \det(A - 4I) = \det \begin{pmatrix} 10 - 4 & -4 \\ 12 & -4 - 4 \end{pmatrix}$$

$$\det(A - 4I) = 6 \times -8 - 12 \times -4 = 0.$$

Já para $\lambda \neq 2$ ou $\lambda \neq 4$, por exemplo, $\lambda = 1$

$$\det(A - \lambda I) = \det(A - I) = \det \begin{pmatrix} 10 - 1 & -4 \\ 12 & -4 - 1 \end{pmatrix}$$

$$\det(A - I) = 9 \times -5 - 12 \times -4 = 3 \neq 0.$$

Polinômio característico

O determinante (2.3) é da forma

$$D_n(\lambda) = \det(A - \lambda I),$$

$$D_n(\lambda) = \det \begin{pmatrix} a_{11} - \lambda & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} - \lambda & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} - \lambda & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} - \lambda \end{pmatrix}. \quad (2.4)$$

O polinômio $D_n(\lambda)$ de grau n é chamado de polinômio característico de A e seus n zeros λ_i são os autovalores de A . Por exemplo, expandindo o determinante dado por (2.4), para $n = 3$, obtém-se o seguinte polinômio característico

$$D_3(\lambda) = -\lambda^3 + [a_{11} + a_{22} + a_{33}] \lambda^2 -$$

$$[(a_{11}a_{22} - a_{12}a_{21}) + (a_{22}a_{33} - a_{23}a_{32}) + (a_{11}a_{33} - a_{13}a_{31})] \lambda +$$

$$[a_{11}(a_{22}a_{33} - a_{23}a_{32}) - a_{12}(a_{11}a_{33} - a_{13}a_{31}) + a_{13}(a_{11}a_{22} - a_{12}a_{21})],$$

que é da forma $D_3(\lambda) = d_3\lambda^3 + d_2\lambda^2 + d_1\lambda + d_0$. Deve ser observado que o coeficiente $d_{n-1} = d_2$ do polinômio é igual a $\sum_{i=1}^n a_{ii}$ = traço(A) e que $d_0 = \det(A)$, conforme (2.1). Pelas relações entre raízes e coeficientes de uma equação algébrica (relações de Girard), mostradas na Seção 6.1.1, tem-se que

$$\lambda_1 + \lambda_2 + \lambda_3 + \cdots + \lambda_n = -\frac{d_{n-1}}{d_n} \text{ e}$$

$$\lambda_1 \lambda_2 \lambda_3 \dots \lambda_n = \prod_{i=1}^n \lambda_i = (-1)^n \frac{d_0}{d_n}.$$

Conseqüentemente, duas importantes propriedades são obtidas

$$\text{traço}(A) = -\frac{d_{n-1}}{d_n} \rightarrow \sum_{i=1}^n a_{ii} = \sum_{i=1}^n \lambda_i,$$

ou seja, a soma dos elementos da diagonal principal de uma matriz (traço) é igual à soma dos seus autovalores e, além disso,

$$\det(A) = (-1)^n \frac{d_0}{d_n} \rightarrow \det(A) = \prod_{i=1}^n \lambda_i,$$

isto é, o determinante de uma matriz é igual ao produto dos seus autovalores. Isso mostra que uma matriz singular tem, no mínimo, um autovalor nulo.

Exemplo 2.13 Para a matriz

$$A = \begin{bmatrix} 10 & -4 \\ 12 & -4 \end{bmatrix}$$

tem-se que

$$\text{traço}(A) = 10 + (-4) = 6 = \lambda_1 + \lambda_2 = 2 + 4 \text{ e}$$

$$\det(A) = 10(-4) - 12(-4) = 8 = \lambda_1 \lambda_2 = 2 \times 4.$$

Por (2.4), uma matriz com elementos reais tem seu polinômio característico com coeficientes reais. Mas, pelo Teorema 6.2, se os coeficientes de uma equação algébrica forem reais, então suas raízes complexas serão complexos conjugados em pares, ou seja, uma matriz com elementos reais tem autovalores reais e/ou complexos conjugados em pares.

Exemplo 2.14 Calcular os autovalores da matriz

$$A = \begin{bmatrix} 2 & 2 \\ 2 & -1 \end{bmatrix}.$$

Por (2.4), o polinômio característico é

$$D_2(\lambda) = \det(A - \lambda I) = \det \begin{pmatrix} 2 - \lambda & 2 \\ 2 & -1 - \lambda \end{pmatrix} = (2 - \lambda)(-1 - \lambda) - 2 \times 2 \rightsquigarrow$$

$$D_2(\lambda) = \lambda^2 - \lambda - 6.$$

Como os autovalores λ_i de A são os zeros do polinômio característico $D_2(\lambda)$, então

$$\lambda = \frac{1 \pm \sqrt{(-1)^2 - 4 \times 1 \times (-6)}}{2} \rightarrow \begin{cases} \lambda_1 = 3 \\ \lambda_2 = -2. \end{cases}$$

É fácil verificar que

$$\text{traço}(A) = 2 + (-1) = 1 = \sum_{i=1}^2 \lambda_i = 3 + (-2) \quad \text{e}$$

$$\det(A) = 2(-1) - 2 \times 2 = -6 = \prod_{i=1}^2 \lambda_i = 3 \times (-2).$$

O processo de calcular os autovalores por intermédio da determinação dos zeros do polinômio característico, apesar de ser esquematicamente simples, é ineficiente em termos computacionais e, portanto, é um processo não recomendado. O estudo de métodos mais eficientes, como aqueles baseados em transformações ortogonais [20, 34], está além do escopo deste texto.

Forma quadrática

Seja A uma matriz simétrica de ordem n com autovalores λ_i , $i = 1, 2, \dots, n$ e v um vetor qualquer não nulo de tamanho n . A forma quadrática de uma matriz é definida pelo escalar

$$q = v^T A v, \forall v \neq 0.$$

A Tabela 2.1 mostra que, dependendo do valor da forma quadrática de A , a matriz pode ter diferentes nomes. Ela apresenta também a relação entre a forma quadrática de A e seus autovalores λ_i .

Tabela 2.1 Valores da forma quadrática.

| Forma quadrática | Nome de A | Autovalores de A |
|------------------|-----------------------|--------------------|
| $v^T A v > 0$ | definida positiva | $\lambda_i > 0$ |
| $v^T A v \geq 0$ | semidefinida positiva | $\lambda_i \geq 0$ |
| $v^T A v < 0$ | definida negativa | $\lambda_i < 0$ |
| $v^T A v \leq 0$ | semidefinida negativa | $\lambda_i \leq 0$ |

Quando uma matriz A não for enquadrada em nenhum dos nomes da Tabela 2.1 é dita indefinida, pois seus autovalores podem ser negativos, nulos e positivos.

Outras propriedades dos autovalores

a) Considerando que $\det(A) = \det(A^T)$, então os autovalores λ de A , representados por $\lambda(A)$, são iguais a $\lambda(A^T)$.

b) Se A for uma matriz triangular de ordem n , então, por (2.4)

$$\det(A - \lambda I) = (a_{11} - \lambda)(a_{22} - \lambda) \cdots (a_{nn} - \lambda) = 0.$$

Esta equação se anula para $\lambda_i = a_{ii}$, $i = 1, 2, \dots, n$, significando que os autovalores de uma matriz triangular ou diagonal são iguais aos elementos da diagonal principal.

c) O posto de uma matriz quadrada é igual ao número de autovalores não nulos.

d) Se λ_i são os autovalores de A , então λ_i^{-1} são os autovalores de A^{-1} . Multiplicando ambos os lados da igualdade $Av = \lambda v$ por A^{-1} , tem-se

$$A^{-1}Av = \lambda A^{-1}v,$$

$$Iv = \lambda A^{-1}v \longrightarrow A^{-1}v = \frac{1}{\lambda}v.$$

Exemplo 2.15 Seja a matriz $A = \begin{bmatrix} 2 & -1 & 4 \\ 0 & 1 & 3 \\ 0 & 0 & 5 \end{bmatrix}$.

Então

Autovalores de A : $\lambda_1 = 2$, $\lambda_2 = 1$, $\lambda_3 = 5$,

posto(A) = 3 e

Autovalores de A^{-1} : $\tau_1 = \lambda_1^{-1} = 0,5$; $\tau_2 = \lambda_2^{-1} = 1$; $\tau_3 = \lambda_3^{-1} = 0,2$; sendo

$$A^{-1} = \begin{bmatrix} 0,5 & 0,5 & -0,7 \\ 0 & 1 & -0,6 \\ 0 & 0 & 0,2 \end{bmatrix}.$$

2.1.4 Normas

O modo usual de expressar a magnitude de um vetor ou de uma matriz é por meio de um escalar denominado norma. No caso de um vetor x de comprimento n , as normas são definidas em termos da norma- p

$$\|x\|_p = \sqrt[p]{\sum_{i=1}^n |x_i|^p}.$$

A partir desta equação geral, obtém-se as três normas vetoriais mais comuns

$$\|x\|_1 = \sum_{i=1}^n |x_i| \quad (\text{norma de soma de magnitudes}),$$

$$\|x\|_2 = \sqrt{\sum_{i=1}^n |x_i|^2} \quad (\text{norma Euclidiana}) \text{ e}$$

$$\|x\|_\infty = \lim_{p \rightarrow \infty} \sqrt[p]{\sum_{i=1}^n |x_i|^p} = \max_{1 \leq i \leq n} |x_i| \quad (\text{norma de máxima magnitude}).$$

Uma norma vetorial é uma função $\|\cdot\| : \mathbb{C}^n \rightarrow \mathbb{R}$, que associa um número real a cada vetor e que satisfaz às condições

$$\|x\| \geq 0 \text{ e } \|x\| = 0 \text{ se, e somente se, } x = 0,$$

$$\|x + y\| \leq \|x\| + \|y\| \text{ e}$$

$$\|kx\| = |k|\|x\|,$$

onde $x, y \in \mathbb{C}^n$ são vetores e $k \in \mathbb{C}$ é um escalar.

Exemplo 2.16 Calcular as normas 1, 2 e ∞ do vetor $x = [3 \ -5 \ 1]^T$.

$$\|x\|_1 = |3| + |-5| + |1| \sim \|x\|_1 = 9,$$

$$\|x\|_2 = \sqrt{|3|^2 + |-5|^2 + |1|^2} \sim \|x\|_2 = \sqrt{35} \approx 5,9161 \text{ e}$$

$$\|x\|_\infty = \max(|3|, |-5|, |1|) \sim \|x\|_\infty = 5.$$

Exemplo 2.17 Calcular as normas 1, 2 e ∞ do vetor $v = [1 \ -3 \ 4+3i \ 4-3i]^T$.

$$\|v\|_1 = |1| + |-3| + |4+3i| + |4-3i| = 1 + 3 + 5 + 5 \sim \|v\|_1 = 14,$$

$$\|v\|_2 = \sqrt{|1|^2 + |-3|^2 + |4+3i|^2 + |4-3i|^2} \sim \|v\|_2 = \sqrt{60} \approx 7,7460 \text{ e}$$

$$\|v\|_\infty = \max(|1|, |-3|, |4+3i|, |4-3i|) \sim \|v\|_\infty = 5.$$

Quanto às normas de matrizes, elas satisfazem às condições

$$\|A\| \geq 0 \text{ e } \|A\| = 0 \text{ se, e somente se, } A = 0,$$

$$\|A + B\| \leq \|A\| + \|B\| \text{ e}$$

$$\|kA\| = |k|\|A\|,$$

onde A e $B \in \mathbb{C}^n$ são matrizes de mesma ordem e $k \in \mathbb{C}^n$ é um escalar. As normas matriciais mais comuns são

$$\|A\|_1 = \max_{1 \leq j \leq n} \sum_{i=1}^n |a_{ij}| \quad (\text{norma de soma máxima de coluna}),$$

$$\|A\|_\infty = \max_{1 \leq i \leq n} \sum_{j=1}^n |a_{ij}| \quad (\text{norma de soma máxima de linha}),$$

$$\|A\|_F = \sqrt{\sum_{i=1}^n \sum_{j=1}^n |a_{ij}|^2} \quad (\text{norma de Frobenius}) \text{ e}$$

$$\|A\|_2 = \begin{cases} \lambda_{\max} & \text{se } A = A^T \\ \sigma_{\max} & \text{se } A \neq A^T \end{cases} \quad (\text{norma espectral}), \quad (2.5)$$

onde λ_{\max} é o maior autovalor de A em módulo e σ_{\max} é o maior valor singular de A , sendo $\sigma_{\max} = \sqrt{\lambda_{\max}(A^T A)}$ (raiz quadrada do maior autovalor em módulo da matriz $A^T A$).

Uma norma matricial $\|A\|$ é dita consistente com uma norma vetorial $\|x\|$ se, para qualquer matriz A ($n \times n$) e vetor x ($n \times 1$), tem-se que $\|Ax\| \leq \|A\|\|x\|$. Uma norma matricial consistente $\|A\|$ é dita subordinada a uma norma vetorial $\|y\|$ se para qualquer matriz A ($n \times n$) existe um vetor y ($n \times 1$), $y \neq 0$, tal que $\|Ay\| = \|A\|\|y\|$. Se a norma for subordinada, então $\|AB\| \leq \|A\|\|B\|$. As normas matriciais 1, 2 e ∞ são consistentes e subordinadas às respectivas normas vetoriais. No entanto, a norma de Frobenius é consistente, mas não subordinada à norma-2 vetorial [3].

Exemplo 2.18 Calcular as normas 1, ∞ , F e 2 da matriz $A = \begin{bmatrix} 2 & -1 \\ 3 & 5 \end{bmatrix}$.

$$\|A\|_1 = \max(|2| + |3|, |-1| + |5|) = \max(5, 6) \sim \|A\|_1 = 6,$$

$$\|A\|_\infty = \max(|2| + |-1|, |3| + |5|) = \max(3, 8) \sim \|A\|_\infty = 8,$$

$$\|A\|_F = \sqrt{|2|^2 + |-1|^2 + |3|^2 + |5|^2} \sim \|A\|_F = \sqrt{39} \approx 6,2450 \text{ e}$$

$$\|A\|_2 = \max(\sqrt{\lambda(A^T A)}) = \max(2,2284; 5,8339) \sim \|A\|_2 = 5,8339.$$

Exemplo 2.19 Calcular as normas 1, ∞ , F e 2 da matriz $B = \begin{bmatrix} 3+4i & -2i \\ 3-4i & 9 \end{bmatrix}$.

$$\|B\|_1 = \max(|3+4i| + |3-4i|, |-2i| + |9|) = \max(10, 11) \sim \|B\|_1 = 11,$$

$$\|B\|_\infty = \max(|3+4i| + |-2i|, |3-4i| + |9|) = \max(7, 14) \sim \|B\|_\infty = 14,$$

$$\|B\|_F = \sqrt{|3+4i|^2 + |-2i|^2 + |3-4i|^2 + |9|^2} \sim \|B\|_F = \sqrt{135} \approx 11,6190 \text{ e}$$

$$\|B\|_2 = \max(\sqrt{\lambda(B^T B)}) = \max(5,2831; 10,3484) \sim \|B\|_2 = 10,3484.$$

2.1.5 Sistemas de equações lineares

Um sistema de equações algébricas lineares consiste em um conjunto de m equações polinomiais com n variáveis x_i de grau 1

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots + a_{3n}x_n &= b_3 \\ &\vdots && \vdots \\ a_{m1}x_1 + a_{m2}x_2 + a_{m3}x_3 + \cdots + a_{mn}x_n &= b_m \end{aligned}$$

que pode ser representado na forma matricial por

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_m \end{bmatrix} \quad (2.6)$$

ou simplesmente $Ax = b$, onde A é chamada de matriz dos coeficientes, x é o vetor solução e b é o vetor dos termos independentes. Se A for uma matriz quadrada ($n \times n$) não singular, então

$$Ax = b \rightarrow A^{-1}Ax = A^{-1}b \rightarrow x = A^{-1}b,$$

ou seja, o vetor solução pode ser obtido pelo produto da inversa da matriz dos coeficientes pelo vetor de termos independentes. Resolver um sistema de equações lineares consiste em encontrar um vetor x ($n \times 1$) que satisfaça simultaneamente às n equações.

2.1.6 Classificação de sistemas

Um sistema linear pode ser classificado sob vários critérios, como, por exemplo, a forma da matriz dos coeficientes e o número de soluções.

Forma da matriz

Quando a matriz dos coeficientes A ($m \times n$) possuir $m \geq n$ e $\text{posto}(A) = n$, têm-se mais equações do que incógnitas e o sistema (2.6) é dito sobre determinado. Este é um problema de quadrados mínimos lineares

$$\underset{x}{\text{minimize}} \quad \|b - Ax\|_2$$

o qual possui uma única solução, chamada de solução de quadrados mínimos.

Por outro lado, quando $m < n$ e $\text{posto}(A) = m$, diz-se que o sistema é subdeterminado, significando que existem mais incógnitas do que equações. Neste caso, ou o sistema não tem solução ou existe um número infinito de soluções que satisfazem $b - Ax = 0$. Nesta situação, é muitas vezes útil encontrar a solução única x que minimiza $\|x\|_2$ e o problema é chamado de determinar a solução de norma mínima do sistema linear (2.6).

A situação mais comum é quando $m = n$, caso em que a matriz dos coeficientes é quadrada. Resolver um sistema de ordem n significa encontrar as n incógnitas x_i que satisfaçam, simultaneamente, às n equações algébricas lineares (2.6). Esta é a situação que será abordada neste capítulo.

Número de soluções

O número de soluções de um sistema de equações lineares depende do valor do determinante da matriz dos coeficientes. Há três situações possíveis:

a) Única solução

$$\begin{aligned} x_1 + x_2 &= 3 \\ x_1 - x_2 &= -1 \end{aligned} \iff \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 3 \\ -1 \end{bmatrix} \sim \det(A) \neq 0 \text{ e } x = [1 \ 2]^T.$$

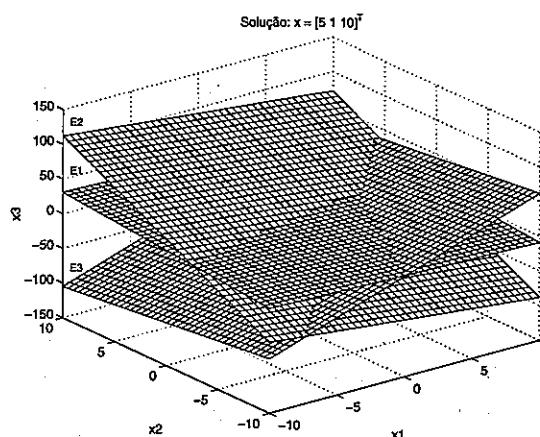
O fato de $\det(A) \neq 0$ resulta que o sistema admite uma única solução. Geometricamente, a solução de um sistema linear de ordem n é um ponto no \mathbb{C}^n comum aos n hiperplanos descritos por cada uma das n equações, ou seja, o ponto que satisfaz simultaneamente a todas as equações. Por exemplo, a solução de

$$\begin{bmatrix} 1 & -3 & 2 \\ -2 & 8 & -1 \\ -20 & 5 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 22 \\ -12 \\ -65 \end{bmatrix}$$

é o vetor $x = [5 \ 1 \ 10]^T$. O vetor solução x é a interseção dos três planos descritos por cada uma das três equações E1, E2 e E3 acima, conforme mostrado na Figura 2.1. Neste caso, com $\det(A) = 251 \neq 0$, os três planos se interceptam em um único ponto no \mathbb{R}^3 .

b) Infinitas soluções

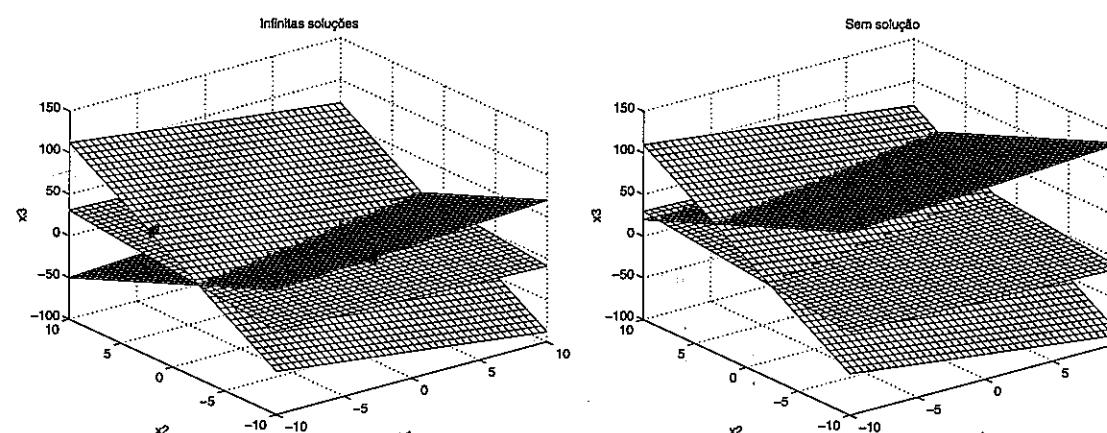
$$\begin{aligned} x_1 + x_2 &= 2 \\ 2x_1 + 2x_2 &= 4 \end{aligned} \iff \begin{bmatrix} 1 & 1 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix} \sim \det(A) = 0 \text{ e } x = [\theta \ 2-\theta]^T.$$

Figura 2.1 Interpretação geométrica da solução de $Ax = b$.

Como $\det(A) = 0$, o sistema admite infinitas soluções, uma para cada valor de θ . A Figura 2.2(a) exibe a representação geométrica dos planos descritos por

$$\begin{bmatrix} 1 & -3 & 2 \\ -2 & 8 & -1 \\ -1 & 5 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 22 \\ -12 \\ 10 \end{bmatrix}.$$

Neste sistema, com $\det(A) = 0$, os três planos se interceptam em uma linha reta descrita por $x = [70 - 6,5\theta \ 16 - 1,5\theta \ \theta]^T$, conforme será apresentado no Exemplo 2.32. Assim, para cada valor de θ ter-se-á uma solução do sistema linear.



(a) Sistema com infinitas soluções.

(b) Sistema sem solução.

Figura 2.2 Interpretação geométrica de sistema linear com matriz singular.

c) Sem solução

$$\begin{aligned} x_1 + x_2 &= 1 \\ x_1 + x_2 &= -1 \end{aligned} \Leftrightarrow \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 1 \\ -1 \end{bmatrix} \Leftrightarrow \det(A) = 0 \text{ e } \nexists x.$$

Se $\det(A) = 0$, o sistema pode também não ter solução. A Figura 2.2(b) mostra que os planos descritos por

$$\begin{bmatrix} 1 & -3 & 2 \\ -2 & 8 & -1 \\ -1 & 5 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 20 \\ -10 \\ 80 \end{bmatrix}$$

(com $\det(A) = 0$) não têm nenhum ponto em comum, ou seja, o sistema acima não admite solução (ver Exemplo 2.32).

Deste modo, conclui-se que, se $\det(A) \neq 0$, o sistema possui uma única solução, e, se $\det(A) = 0$, ou o sistema tem infinitas soluções ou nenhuma. Sistema linear com matriz dos coeficientes singular será abordado na Seção 2.4.4.

2.2 Sistemas triangulares

Os sistemas triangulares são amplamente utilizados porque as suas soluções são facilmente obtidas. Há dois tipos de sistemas triangulares: inferiores e superiores.

2.2.1 Sistema triangular inferior

Um sistema triangular inferior de ordem n apresenta a forma

$$\begin{bmatrix} l_{11} & 0 & 0 & \cdots & 0 \\ l_{21} & l_{22} & 0 & \cdots & 0 \\ l_{31} & l_{32} & l_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 \\ l_{n1} & l_{n2} & l_{n3} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{bmatrix}.$$

A solução de um sistema triangular inferior é calculada pelas substituições sucessivas

$$\begin{aligned} l_{11}x_1 &= c_1 \rightsquigarrow x_1 = \frac{c_1}{l_{11}}, \\ l_{21}x_1 + l_{22}x_2 &= c_2 \rightsquigarrow x_2 = \frac{c_2 - l_{21}x_1}{l_{22}}, \\ l_{31}x_1 + l_{32}x_2 + l_{33}x_3 &= c_3 \rightsquigarrow x_3 = \frac{c_3 - l_{31}x_1 - l_{32}x_2}{l_{33}}, \\ &\dots \\ l_{n1}x_1 + l_{n2}x_2 + \cdots + l_{n,n-1}x_{n-1} + l_{nn}x_n &= c_n \rightsquigarrow \\ x_n &= \frac{c_n - l_{n1}x_1 - l_{n2}x_2 - \cdots - l_{n,n-1}x_{n-1}}{l_{nn}}. \end{aligned}$$

As substituições sucessivas podem ser representadas por

$$x_i = \frac{c_i - \sum_{j=1}^{i-1} l_{ij}x_j}{l_{ii}}, \quad i = 1, 2, \dots, n.$$

(2.7)

Exemplo 2.20 Calcular a solução do sistema triangular inferior usando as substituições sucessivas (2.7)

$$\begin{bmatrix} 2 & 0 & 0 & 0 \\ 3 & 5 & 0 & 0 \\ 1 & -6 & 8 & 0 \\ -1 & 4 & -3 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ 48 \\ 6 \end{bmatrix}$$

$$2x_1 = 4, x_1 = \frac{4}{2} \rightsquigarrow x_1 = 2,$$

$$3x_1 + 5x_2 = 1, x_2 = \frac{1 - 3(2)}{5} \rightsquigarrow x_2 = -1,$$

$$x_1 - 6x_2 + 8x_3 = 48, x_3 = \frac{48 - (2) + 6(-1)}{8} \rightsquigarrow x_3 = 5 \text{ e}$$

$$-x_1 + 4x_2 - 3x_3 + 9x_4 = 6, x_4 = \frac{6 + (2) - 4(-1) + 3(5)}{9} \rightsquigarrow x_4 = 3.$$

Conseqüentemente, a solução do sistema triangular inferior é $x = [2 \ -1 \ 5 \ 3]^T$. A Figura 2.3 mostra um algoritmo para resolver um sistema triangular inferior usando as substituições sucessivas (2.7). Os parâmetros de entrada são a ordem n do sistema, a matriz triangular inferior L e o vetor de termos independentes c . O parâmetro de saída é o vetor solução x .

```

Algoritmo Substituições Sucessivas
{ Objetivo: Resolver o sistema triangular inferior  $Lx = c$  }
{ pelas substituições sucessivas }
parâmetros de entrada  $n, L, c$ 
{ ordem, matriz triangular inferior e vetor independente }
parâmetros de saída  $x$  { solução do sistema triangular inferior }
   $x(1) \leftarrow c(1)/L(1, 1)$ 
  para  $i \leftarrow 2$  até  $n$  faça
    Soma  $\leftarrow 0$ 
    para  $j \leftarrow 1$  até  $i - 1$  faça
      Soma  $\leftarrow Soma + L(i, j) \cdot x(j)$ 
    fimpara
     $x(i) \leftarrow (c(i) - Soma)/L(i, i)$ 
  fimpara
fimalgoritmo

```

Figura 2.3 Solução de sistema triangular inferior pelas substituições sucessivas.

Exemplo 2.21 Resolver o sistema triangular inferior do Exemplo 2.20 usando o algoritmo da Figura 2.3.

% Os valores de entrada

$$\begin{aligned} n &= 4 \\ L &= \begin{bmatrix} 2 & 0 & 0 & 0 \\ 3 & 5 & 0 & 0 \\ 1 & -6 & 8 & 0 \\ -1 & 4 & -3 & 9 \end{bmatrix} \\ c &= \begin{bmatrix} 4 \\ 1 \\ 48 \\ 6 \end{bmatrix} \end{aligned}$$

% produzem o resultado

$$\begin{aligned} x &= \begin{bmatrix} 2 \\ -1 \\ 5 \\ 3 \end{bmatrix} \end{aligned}$$

2.2.2 Sistema triangular superior

Um sistema triangular superior de ordem n apresenta a forma

$$\begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1,n-1} & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & u_{2,n-1} & u_{2n} \\ 0 & 0 & u_{33} & \cdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & u_{n-1,n-1} & u_{n-1,n} \\ 0 & 0 & 0 & \cdots & 0 & u_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{n-1} \\ d_n \end{bmatrix}.$$

O vetor solução de um sistema triangular superior é obtido pelas substituições retroativas

$$u_{nn}x_n = d_n \rightsquigarrow x_n = \frac{d_n}{u_{nn}},$$

$$u_{n-1,n-1}x_{n-1} + u_{n-1,n}x_n = d_{n-1} \rightsquigarrow x_{n-1} = \frac{d_{n-1} - u_{n-1,n}x_n}{u_{n-1,n-1}},$$

$$u_{22}x_2 + u_{23}x_3 + \cdots + u_{2n}x_n = d_2 \rightsquigarrow x_2 = \frac{d_2 - u_{23}x_3 - \cdots - u_{2n}x_n}{u_{22}} \text{ e}$$

$$u_{11}x_1 + u_{12}x_2 + u_{13}x_3 + \cdots + u_{1n}x_n = d_1 \rightsquigarrow x_1 = \frac{d_1 - u_{12}x_2 - u_{13}x_3 - \cdots - u_{1n}x_n}{u_{11}}.$$

As substituições retroativas podem ser representadas por

$$x_i = \frac{d_i - \sum_{j=i+1}^n u_{ij}x_j}{u_{ii}}, \quad i = n, n-1, \dots, 1. \quad (2.8)$$

Exemplo 2.22 Determinar a solução do sistema triangular superior utilizando as substituições retroativas (2.8)

$$\begin{bmatrix} 5 & -2 & 6 & 1 \\ 0 & 3 & 7 & -4 \\ 0 & 0 & 4 & 5 \\ 0 & 0 & 0 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ 28 \\ 8 \end{bmatrix}.$$

$$2x_4 = 8, x_4 = \frac{8}{2} \rightsquigarrow x_4 = 4,$$

$$4x_3 + 5x_4 = 28, x_3 = \frac{28 - 5(4)}{4} \rightsquigarrow x_3 = 2,$$

$$3x_2 + 7x_3 - 4x_4 = -2, x_2 = \frac{-2 - 7(2) + 4(4)}{3} \rightsquigarrow x_2 = 0 \text{ e}$$

$$5x_1 - 2x_2 + 6x_3 + x_4 = 1, x_1 = \frac{1 + 2(0) - 6(2) - (4)}{5} \rightsquigarrow x_1 = -3.$$

O vetor solução do sistema triangular superior é $x = [-3 \ 0 \ 2 \ 4]^T$. Deve ser observado que os elementos de x são obtidos em ordem reversa. A Figura 2.4 exibe um algoritmo para resolver um sistema triangular superior pelas substituições retroativas (2.8). Os parâmetros de entrada são a ordem n do sistema, a matriz triangular superior U e o vetor de termos independentes d . O parâmetro de saída é o vetor solução x .

```

Algoritmo Substituições Retroativas
{ Objetivo: Resolver o sistema triangular superior  $Ux = d$  }
{ pelas substituições retroativas }
parâmetros de entrada  $n, U, d$ 
{ ordem, matriz triangular superior e vetor independente }
parâmetros de saída  $x$  { solução do sistema triangular superior }
 $x(n) \leftarrow d(n)/U(n, n)$ 
para  $i \leftarrow n-1$  até 1 passo -1 faça
    Soma  $\leftarrow 0$ 
    para  $j \leftarrow i+1$  até  $n$  faça
        Soma  $\leftarrow Soma + U(i, j) * x(j)$ 
    fim para
     $x(i) \leftarrow (d(i) - Soma)/U(i, i)$ 
fim para
finalgoritmo

```

Figura 2.4 Solução de sistema triangular superior pelas substituições retroativas.

Exemplo 2.23 Resolver o sistema triangular superior do Exemplo 2.22 usando o algoritmo da Figura 2.4.

% Os valores de entrada

$n = 4$
 $U =$
 $\begin{bmatrix} 5 & -2 & 6 & 1 \\ 0 & 3 & 7 & -4 \\ 0 & 0 & 4 & 5 \\ 0 & 0 & 0 & 2 \end{bmatrix}$
 $d =$
 $\begin{bmatrix} 1 \\ -2 \\ 28 \\ 8 \end{bmatrix}$

% produzem o resultado

$x =$
 $\begin{bmatrix} -3 \\ 0 \\ 2 \\ 4 \end{bmatrix}$

2.2.3 Complexidade computacional

Considerando que $\sum_{i=1}^n i = \frac{n(n+1)}{2}$, a complexidade computacional do algoritmo de substituições sucessivas mostrado na Figura 2.3 é

$$\text{Adições: } \sum_{i=2}^n [(i-1)+1] = \frac{n(n+1)}{2} - 1 = \frac{n^2}{2} + \frac{n}{2} - 1;$$

$$\text{Multiplicações: } \sum_{i=2}^n (i-1) = \frac{n(n+1)}{2} - 1 - (n-1) = \frac{n^2}{2} - \frac{n}{2};$$

$$\text{Divisões: } 1 + \sum_{i=2}^n 1 = 1 + n - 1 = n.$$

Os resultados acima estão resumidos na Tabela 2.2.

Tabela 2.2 Complexidade das substituições sucessivas para sistema de ordem n .

| Operações | Complexidade |
|----------------|-------------------------------------|
| adições | $\frac{1}{2}n^2 + \frac{1}{2}n - 1$ |
| multiplicações | $\frac{1}{2}n^2 - \frac{1}{2}n$ |
| divisões | n |

De modo similar, a complexidade computacional do algoritmo de substituições retroativas da Figura 2.4 é

$$\text{Adições: } \sum_{i=1}^{n-1} [(n-i)+1] = (n-1)n - \frac{(n-1)n}{2} + n - 1 = \frac{n^2}{2} + \frac{n}{2} - 1;$$

$$\text{Multiplicações: } \sum_{i=1}^{n-1} (n-i) = (n-1)n - \frac{(n-1)n}{2} = \frac{n^2}{2} - \frac{n}{2};$$

$$\text{Divisões: } 1 + \sum_{i=1}^{n-1} 1 = 1 + n - 1 = n.$$

Estes resultados estão resumidos na Tabela 2.3, os quais são idênticos aos mostrados na Tabela 2.2.

Tabela 2.3 Complexidade das substituições retroativas para sistema de ordem n .

| Operações | Complexidade |
|----------------|-------------------------------------|
| adições | $\frac{1}{2}n^2 + \frac{1}{2}n - 1$ |
| multiplicações | $\frac{1}{2}n^2 - \frac{1}{2}n$ |
| divisões | n |

Como o número de operações aritméticas das substituições sucessivas e retroativas é descrito por polinômios, diz-se que estes algoritmos são polinomiais.

2.3 Eliminação de Gauss

Existem duas grandes classes de métodos para resolução de sistemas de equações lineares: métodos diretos e iterativos. Os métodos diretos são aqueles em que a solução exata do sistema é obtida, teoricamente, com um número finito de operações aritméticas. Na prática, os erros de arredondamento devidos à aritmética de ponto flutuante interferem no resultado verdadeiro. Por outro lado, os métodos iterativos obtêm a solução exata somente com um número infinito de operações. Em cada passo desses métodos a solução é calculada com um nível de exatidão crescente, o qual é limitado pelo número finito de bytes utilizados para armazenar as variáveis do programa que implementa o método iterativo. Os métodos de eliminação de Gauss, decomposição LU, de Cholesky e LDL^T , que serão abordados neste capítulo, são exemplos de métodos diretos.

2.3.1 Sistemas equivalentes

Antes de ser apresentado o método de eliminação de Gauss, faz-se necessário definir o conceito de equivalência de sistemas. Dois sistemas de equações lineares são ditos equivalentes quando possuem o mesmo vetor solução. Por exemplo,

$$A \left\{ \begin{array}{l} 2x_1 + 3x_2 = 8 \\ x_1 - x_2 = -1 \end{array} \right. \text{ e } B \left\{ \begin{array}{l} 2x_1 - 2x_2 = -2 \\ x_1 + 4x_2 = 9 \end{array} \right. \Rightarrow x^A = x^B = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \Rightarrow A \sim B,$$

onde o símbolo \sim significa equivalência.

2.3.2 Operações l-elementares

Um sistema de equações lineares pode ser transformado em um outro sistema equivalente utilizando as três operações l-elementares (operações de linha)

a) Trocar a ordem de duas equações

$$B \left\{ \begin{array}{l} 2x_1 - 2x_2 = -2 \\ x_1 + 4x_2 = 9 \end{array} \right. \text{ e } C \left\{ \begin{array}{l} x_1 + 4x_2 = 9 \\ 2x_1 - 2x_2 = -2 \end{array} \right. \Rightarrow x^B = x^C = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \Rightarrow B \sim C.$$

b) Multiplicar uma equação por uma constante não nula

$$C \left\{ \begin{array}{l} x_1 + 4x_2 = 9 \\ 2x_1 - 2x_2 = -2 \end{array} \right. \text{ e } D \left\{ \begin{array}{l} x_1 + 4x_2 = 9 \\ x_1 - x_2 = -1 \end{array} \right. \Rightarrow x^C = x^D = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \Rightarrow C \sim D.$$

c) Somar uma equação à outra

$$D \left\{ \begin{array}{l} x_1 + 4x_2 = 9 \\ x_1 - x_2 = -1 \end{array} \right. \text{ e } E \left\{ \begin{array}{l} 2x_1 + 3x_2 = 8 \\ x_1 - x_2 = -1 \end{array} \right. \Rightarrow x^D = x^E = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \Rightarrow D \sim E.$$

Pode ser observado que $A \sim B \sim C \sim D \sim E = A$. Portanto, pelo uso das operações l-elementares torna-se possível transformar um sistema linear em um outro sistema equivalente de solução mais fácil, como, por exemplo, um sistema triangular.

2.3.3 Sistema triangular equivalente

O método de eliminação de Gauss¹ consiste em transformar um sistema de equações lineares em um sistema triangular superior equivalente por intermédio das operações l-elementares, ou seja,

$$\left[\begin{array}{cccc|c} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} & x_1 \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} & x_2 \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} & x_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} & x_n \end{array} \right] \left[\begin{array}{c} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{array} \right] \sim \left[\begin{array}{cccc|c} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} & x_1 \\ 0 & u_{22} & u_{23} & \cdots & u_{2n} & x_2 \\ 0 & 0 & u_{33} & \cdots & u_{3n} & x_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & u_{nn} & x_n \end{array} \right] \left[\begin{array}{c} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{array} \right].$$

A transformação também pode ser representada por $Ax = b \sim Ux = d$. Assim, a solução do sistema triangular superior $Ux = d$ é obtida pelas substituições retroativas (2.8). A exatidão da solução pode ser verificada pelo cálculo do vetor resíduo $r = b - Ax$, de modo que se $r = 0$, então a solução é exata.

Exemplo 2.24 Resolver o sistema abaixo pelo método de eliminação de Gauss e verificar a exatidão da solução

$$(8.3) \quad \left[\begin{array}{ccc|c} 1 & -3 & 2 & 11 \\ -2 & 8 & -1 & -15 \\ 4 & -6 & 5 & 29 \end{array} \right] \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \end{array} \right] = \left[\begin{array}{c} 11 \\ -15 \\ 29 \end{array} \right].$$

Res...

¹O nome do método foi uma homenagem a Gauss. O processo aparece no livro chinês *Nove capítulos sobre a arte matemática*, escrito por volta de 250 a.C.

Os elementos da primeira coluna abaixo da diagonal devem ser eliminados, baseando-se no elemento da diagonal da primeira linha $a_{11} = 1$. Por esta razão, a_{11} é chamado de elemento pivô e a linha que o contém é a linha pivotal. Para eliminar $a_{21} = -2$, a primeira linha deve ser multiplicada por um fator $-m_{21}$ e somada à segunda linha. Este fator é tal que $-m_{21}a_{11} + a_{21} = 0 \rightarrow m_{21} = a_{21}/a_{11} = (-2)/1 = -2$. A nova linha 2 será $L'_2 = 2L_1 + L_2$. Obviamente, esta operação l-elementar deve ser efetuada nos dois lados da igualdade.

Do mesmo modo, para eliminar $a_{31} = 4$ deve-se multiplicar a primeira linha por $-m_{31}$ e somar à terceira linha, com $-m_{31}a_{11} + a_{31} = 0 \rightarrow m_{31} = a_{31}/a_{11} = 4/1 = 4$, ou seja, $L'_3 = -4L_1 + L_3$. Após estas duas operações l-elementares, o sistema equivalente intermediário terá os dois elementos abaixo da diagonal iguais a zero

$$\begin{bmatrix} 1 & -3 & 2 \\ 0 & 2 & 3 \\ 0 & 6 & -3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 7 \\ -15 \end{bmatrix}. \quad (2.9)$$

Agora, para eliminar o elemento da segunda coluna abaixo da diagonal, deve-se usar $a'_{22} = 2$ como elemento pivô e a segunda linha como pivotal. Se a primeira linha continuar sendo a pivotal, então o elemento $a'_{31} = 0$ terá um indesejável valor diferente de zero. A segunda linha é multiplicada pelo fator $-m_{32}$ e somada à terceira linha, com $-m_{32}a'_{22} + a'_{32} = 0 \rightarrow m_{32} = a'_{32}/a'_{22} = 6/2 = 3$. A nova linha 3 será $L''_3 = -3L'_2 + L'_3$, resultando finalmente no sistema triangular superior equivalente

$$\begin{bmatrix} 1 & -3 & 2 \\ 0 & 2 & 3 \\ 0 & 0 & -12 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 7 \\ -36 \end{bmatrix}. \quad (2.10)$$

Esta etapa de obtenção do sistema triangular superior equivalente pode ser summarizada no seguinte dispositivo prático

| <i>L</i> | Multiplicador | <i>A</i> | <i>b</i> | Operações |
|----------|------------------------|----------|----------|---------------|
| 1 | | 1 -3 2 | 11 | |
| 2 | $m_{21} = (-2)/1 = -2$ | -2 8 -1 | -15 | |
| 3 | $m_{31} = 4/1 = 4$ | 4 -6 5 | 29 | |
| 4 | | 0 2 3 | 7 | $2L_1 + L_2$ |
| 5 | $m_{32} = 6/2 = 3$ | 0 6 -3 | -15 | $-4L_1 + L_3$ |
| 6 | | 0 0 -12 | -36 | $-3L_4 + L_5$ |

O sistema intermediário equivalente (2.9) é composto por L_1 (linha pivotal), L_4 e L_5 . O sistema triangular superior equivalente (2.10) é composto por L_1 (primeira linha pivotal), L_4 (segunda linha pivotal) e L_6 (última linha), e os elementos pivôs estão sublinhados. O sistema triangular superior (2.10) pode ser resolvido pelas substituições retroativas (2.8)

$$-12x_3 = -36, x_3 = \frac{-36}{-12} \sim x_3 = 3,$$

$$2x_2 + 3x_3 = 7, x_2 = \frac{7 - 3(3)}{2} \sim x_2 = -1 \text{ e}$$

$$x_1 - 3x_2 + 2x_3 = 11, x_1 = \frac{11 + 3(-1) - 2(3)}{1} \sim x_1 = 2.$$

Conseqüentemente, o vetor solução do sistema é $x = [2 \ -1 \ 3]^T$. O vetor resíduo $r = b - Ax$ deve ser utilizado para verificar a exatidão da solução obtida

$$r = \begin{bmatrix} 11 \\ -15 \\ 29 \end{bmatrix} - \begin{bmatrix} 1 & -3 & 2 \\ -2 & 8 & -1 \\ 4 & -6 & 5 \end{bmatrix} \begin{bmatrix} 2 \\ -1 \\ 3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Como o vetor resíduo é nulo, a solução é exata. Deve ser usado o sistema original $Ax = b$ para calcular o resíduo r e não o triangular equivalente $Ux = d$. Desta forma, poderá ser detectado um possível erro cometido ao se obter o sistema triangular equivalente. ■

Exemplo 2.25 Resolver o sistema abaixo pelo método de eliminação de Gauss e verificar a exatidão da solução

$$\begin{bmatrix} 1 & 6 & 2 & 4 \\ 3 & 19 & 4 & 15 \\ 1 & 4 & 8 & -12 \\ 5 & 33 & 9 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 8 \\ 25 \\ 18 \\ 72 \end{bmatrix}.$$

Usando o dispositivo prático, tem-se

| <i>L</i> | Multiplicador | <i>A</i> | <i>b</i> | Operações |
|----------|------------------------|------------|----------|-----------------|
| 1 | | 1 6 2 4 | 8 | |
| 2 | $m_{21} = 3/1 = 3$ | 3 19 4 15 | 25 | |
| 3 | $m_{31} = 1/1 = 1$ | 1 4 8 -12 | 18 | |
| 4 | $m_{41} = 5/1 = 5$ | 5 33 9 3 | 72 | |
| 5 | | 0 1 -2 3 | 1 | $-3L_1 + L_2$ |
| 6 | $m_{32} = (-2)/1 = -2$ | 0 -2 6 -16 | 10 | $-L_1 + L_3$ |
| 7 | $m_{42} = 3/1 = 3$ | 0 3 -1 -17 | 32 | $-5L_1 + L_4$ |
| 8 | | 0 0 2 -10 | 12 | $2L_5 + L_6$ |
| 9 | $m_{43} = 5/2 = 2,5$ | 0 0 5 -26 | 29 | $-3L_5 + L_7$ |
| 10 | | 0 0 0 -1 | -1 | $-2,5L_8 + L_9$ |

O sistema triangular superior equivalente é composto pelas linhas pivotais (com os pivôs sublinhados), ou seja, L_1 , L_5 , L_8 e L_{10}

$$\begin{bmatrix} 1 & 6 & 2 & 4 \\ 0 & 1 & -2 & 3 \\ 0 & 0 & 2 & -10 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 8 \\ 1 \\ 12 \\ -1 \end{bmatrix}.$$

Resolvendo pelas substituições retroativas (2.8)

$$-1x_4 = -1, x_4 = \frac{-1}{-1} \sim x_4 = 1,$$

$$2x_3 - 10x_4 = 12, \quad x_3 = \frac{12 + 10(1)}{2} \rightsquigarrow x_3 = 11,$$

$$x_2 - 2x_3 + 3x_4 = 1, \quad x_2 = \frac{1 + 2(11) - 3(1)}{1} \rightsquigarrow x_2 = 20 \text{ e}$$

$$x_1 + 6x_2 + 2x_3 + 4x_4 = 8, \quad x_1 = \frac{8 - 6(20) - 2(11) - 4(1)}{1} \rightsquigarrow x_1 = -138.$$

Assim, o vetor solução é $x = [-138 \ 20 \ 11 \ 1]^T$, e o resíduo

$$r = \begin{bmatrix} 8 \\ 25 \\ 18 \\ 72 \end{bmatrix} - \begin{bmatrix} 1 & 6 & 2 & 4 \\ 3 & 19 & 4 & 15 \\ 1 & 4 & 8 & -12 \\ 5 & 33 & 9 & 3 \end{bmatrix} \begin{bmatrix} -138 \\ 20 \\ 11 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

mostra que a solução é exata. ■

2.3.4 Cálculo do determinante

O valor do determinante da matriz dos coeficientes pode ser obtido como um subproduto do método de eliminação de Gauss. Para tanto, basta saber quais as relações entre os determinantes das matrizes dos sistemas equivalentes intermediários obtidos pelas operações l-elementares durante o processo de eliminação.

a) Se duas linhas quaisquer de uma matriz A forem trocadas, então o determinante da nova matriz B será

$$\det(B) = -\det(A).$$

$$A = \begin{bmatrix} 2 & -2 \\ 1 & 4 \end{bmatrix} \rightarrow \det(A) = 10 \text{ e } B = \begin{bmatrix} 1 & 4 \\ 2 & -2 \end{bmatrix} \rightarrow \det(B) = -10.$$

b) Se todos os elementos de uma linha de A forem multiplicados por uma constante k , então o determinante da matriz resultante B será

$$\det(B) = k \det(A).$$

$$A = \begin{bmatrix} 1 & 4 \\ 2 & -2 \end{bmatrix} \rightarrow \det(A) = -10 \text{ e } B = \begin{bmatrix} 1 & 4 \\ 1 & -1 \end{bmatrix} \rightarrow \det(B) = -5.$$

c) Se um múltiplo escalar de uma linha de A for somado a outra linha, então o determinante da nova matriz B será

$$\det(B) = \det(A).$$

$$A = \begin{bmatrix} 1 & 4 \\ 1 & -1 \end{bmatrix} \rightarrow \det(A) = -5 \text{ e } B = \begin{bmatrix} 1 & 4 \\ 0 & -5 \end{bmatrix} \rightarrow \det(B) = -5.$$

d) Se A for uma matriz triangular ou diagonal de ordem n , então o seu determinante será igual ao produto dos elementos da diagonal principal, ou seja,

$$\det(A) = a_{11}a_{22}a_{33}\dots a_{nn} = \prod_{i=1}^n a_{ii}.$$

$$A = \begin{bmatrix} 2 & 3 \\ 0 & -1 \end{bmatrix} \rightarrow \det(A) = -2 \text{ e } B = \begin{bmatrix} 3 & 0 & 0 \\ 0 & -5 & 0 \\ 0 & 0 & -1 \end{bmatrix} \rightarrow \det(B) = 15.$$

e) Se uma matriz A for multiplicada por uma matriz B , o determinante da matriz resultante C será

$$\det(C) = \det(A) \det(B).$$

$$A = \begin{bmatrix} 1 & -2 \\ 3 & 4 \end{bmatrix} \rightarrow \det(A) = 10, \quad B = \begin{bmatrix} 3 & 0 \\ 1 & 1 \end{bmatrix} \rightarrow \det(B) = 3 \text{ e }$$

$$C = \begin{bmatrix} 1 & -2 \\ 13 & 4 \end{bmatrix} \rightarrow \det(C) = 30.$$

Exemplo 2.26 Calcular o determinante da matriz do Exemplo 2.24

$$\begin{bmatrix} 1 & -3 & 2 \\ -2 & 8 & -1 \\ 4 & -6 & 5 \end{bmatrix}.$$

A seqüência de matrizes obtidas pelas operações l-elementares foram as dos sistemas (2.9) e (2.10).

$$\begin{bmatrix} 1 & -3 & 2 \\ -2 & 8 & -1 \\ 4 & -6 & 5 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & -3 & 2 \\ 0 & 2 & 3 \\ 0 & 6 & -3 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & -3 & 2 \\ 0 & 2 & 3 \\ 0 & 0 & -12 \end{bmatrix}.$$

Considerando que (2.9) e (2.10) foram obtidas somente por intermédio de combinações lineares das linhas, então estas três matrizes possuem determinantes com o mesmo valor (propriedade c). Deste modo, como (2.10) é uma matriz triangular, então, pela propriedade d, o determinante será igual ao produto dos elementos da diagonal, ou seja, o determinante será simplesmente o produto dos pivôs

$$\det(A) = 1 \times 2 \times -12 = -24.$$

2.3.5 Pivotação parcial

O método de Gauss irá falhar quando um pivô for nulo, pois, neste caso, não será possível calcular os multiplicadores m_{ij} utilizados na eliminação. Este sério problema pode ser evitado pelo uso da estratégia da pivotação parcial, que consiste em escolher como pivô o maior elemento em módulo da coluna, cujos elementos serão eliminados. A pivotação parcial garante que o pivô seja não nulo, exceto quando a matriz for singular. Outra vantagem é que todos os multiplicadores satisfazem $-1 \leq m_{ij} \leq 1$, evitando, assim, que eles sejam muito grandes. Multiplicadores grandes podem ampliar os erros de arredondamento de modo a comprometer a solução do sistema.

A única diferença entre o método de eliminação de Gauss com e sem pivotação parcial está no modo de escolha do elemento pivô. Nos dois métodos, após o pivô ter sido escolhido, devem ser eliminados os demais elementos da coluna que contém o pivô, exceto aqueles elementos que pertençam a alguma linha pivotal.

Exemplo 2.27 Resolver o sistema do Exemplo 2.24 pelo método de Gauss com pivotação parcial

$$\begin{bmatrix} 1 & -3 & 2 \\ -2 & 8 & -1 \\ 4 & -6 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ -15 \\ 29 \end{bmatrix}.$$

O primeiro elemento pivô escolhido é $a_{31} = 4$ porque ele é o maior elemento em módulo da primeira coluna. Para eliminar os demais elementos da coluna (a_{11} e a_{21}) efetua-se $L'_1 = -m_{11}L_3 + L_1$, com a condição $-m_{11}a_{31} + a_{11} = 0 \rightarrow m_{11} = a_{11}/a_{31} = 1/4 = 0,25$ e $L'_2 = -m_{21}L_3 + L_2$, com $-m_{21}a_{31} + a_{21} = 0 \rightarrow m_{21} = a_{21}/a_{31} = (-2)/4 = -0,5$. Efetuando-se essas duas operações l-elementares, o sistema equivalente intermediário conterá dois elementos nulos na primeira coluna

$$\begin{bmatrix} 0 & -1,5 & 0,75 \\ 0 & 5 & 1,5 \\ 4 & -6 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3,75 \\ -0,5 \\ 29 \end{bmatrix}.$$

Para eliminar o elemento da segunda coluna, escolhe-se o maior elemento em módulo desta coluna, sem considerar o elemento da linha pivotal L_3 . Portanto, o elemento pivô será $a'_{22} = 5$. Para eliminar a'_{12} , basta $L''_1 = -m_{12}L'_2 + L'_1$, com $-m_{12}a'_{22} + a'_{12} = 0 \rightarrow m_{12} = a'_{12}/a'_{22} = (-1,5)/5 = -0,3$. O sistema equivalente é

$$\begin{bmatrix} 0 & 0 & 1,2 \\ 0 & 5 & 1,5 \\ 4 & -6 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3,6 \\ -0,5 \\ 29 \end{bmatrix},$$

o qual pode ser transformado em um sistema triangular superior efetuando-se a operação l-elementar de trocar a ordem de duas linhas, no caso, a primeira e a terceira linhas

$$\begin{bmatrix} 4 & -6 & 5 \\ 0 & 5 & 1,5 \\ 0 & 0 & 1,2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 29 \\ -0,5 \\ 3,6 \end{bmatrix}.$$

Estas operações l-elementares estão representadas no seguinte dispositivo prático

| L | Multiplicador | A | b | Operações |
|-----|----------------------------|-----|------|------------------|
| 1 | $m_{11} = 1/4 = 0,25$ | 1 | -3 | 2 |
| 2 | $m_{21} = (-2)/4 = -0,5$ | -2 | 8 | -1 |
| 3 | | 4 | -6 | 5 |
| 4 | $m_{12} = (-1,5)/5 = -0,3$ | 0 | -1,5 | 0,75 |
| 5 | | 0 | 5 | 1,5 |
| 6 | | 0 | 0 | 1,2 |
| | | | 11 | |
| | | | -15 | |
| | | | 29 | |
| | | | 3,75 | $-0,25L_3 + L_1$ |
| | | | -0,5 | $0,5L_3 + L_2$ |
| | | | 3,6 | $0,3L_5 + L_4$ |

O sistema triangular superior é constituído pelas linhas pivotais L_3 , L_5 e L_6 , ou seja, as linhas que contêm os elementos pivôs sublinhados. Este sistema é resolvido pelas substituições retroativas (2.8)

$$1,2x_3 = 3,6; x_3 = \frac{3,6}{1,2} \sim x_3 = 3,$$

$$5x_2 + 1,5x_3 = -0,5; x_2 = \frac{-0,5 - 1,5(3)}{5} \sim x_2 = -1 \text{ e}$$

$$4x_1 - 6x_2 + 5x_3 = 29, x_1 = \frac{29 + 6(-1) - 5(3)}{4} \sim x_1 = 2.$$

Por isso, $x = [2 \ -1 \ 3]^T$.

2.4 Decomposição LU

Uma matriz quadrada pode ser escrita como o produto de duas matrizes, por exemplo,

$$\begin{bmatrix} 4 & 3 \\ 8 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 4 & 3 \\ 0 & -1 \end{bmatrix}.$$

Assim, uma matriz A foi fatorada tal que $A = LU$, onde L é uma matriz triangular inferior unitária ($l_{ii} = 1, \forall i$) e U é uma matriz triangular superior. Deste modo, para resolver o sistema $Ax = b$, usa-se a matriz A na forma decomposta

$$Ax = b \longrightarrow LUx = b.$$

Fazendo

$$Ux = y, \text{ então } Ly = b.$$

A solução y do sistema triangular inferior $Ly = b$ é obtida pelas substituições sucessivas (2.7) com $l_{ii} = 1$ porque L é uma matriz unitária. O vetor y é usado como termo independente no sistema triangular superior $Ux = y$, cuja solução x é calculada pelas substituições retroativas (2.8).

2.4.1 Cálculo dos fatores

Uma matriz A pode ser fatorada usando-se o método de eliminação de Gauss. A matriz triangular superior U é a mesma do método de Gauss e a matriz triangular inferior unitária L , além de $l_{ii} = 1$, $l_{ij} = 0, i < j$, possui $l_{ij} = m_{ij}, i > j$, sendo m_{ij} os multiplicadores usados no processo de eliminação de Gauss.

Exemplo 2.28 Resolver o sistema do Exemplo 2.24 usando a decomposição LU

$$\begin{bmatrix} 1 & -3 & 2 \\ -2 & 8 & -1 \\ 4 & -6 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ -15 \\ 29 \end{bmatrix}.$$

Utilizando um dispositivo prático similar ao da eliminação de Gauss, tem-se

| L | m | A | Operações |
|-----|------------------------|---------|---------------|
| 1 | | 1 -3 2 | |
| 2 | $m_{21} = (-2)/1 = -2$ | -2 8 -1 | |
| 3 | $m_{31} = 4/1 = 4$ | 4 -6 5 | |
| 4 | | 0 2 3 | $2L_1 + L_2$ |
| 5 | $m_{32} = 6/2 = 3$ | 0 6 -3 | $-4L_1 + L_3$ |
| 6 | | 0 0 -12 | $-3L_4 + L_5$ |

A partir do dispositivo, obtém-se as duas matrizes

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 4 & 3 & 1 \end{bmatrix} \quad \text{e} \quad U = \begin{bmatrix} 1 & -3 & 2 \\ 0 & 2 & 3 \\ 0 & 0 & -12 \end{bmatrix}.$$

Pode ser verificada a igualdade $A = LU$

$$\begin{bmatrix} 1 & -3 & 2 \\ -2 & 8 & -1 \\ 4 & -6 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 4 & 3 & 1 \end{bmatrix} \begin{bmatrix} 1 & -3 & 2 \\ 0 & 2 & 3 \\ 0 & 0 & -12 \end{bmatrix}.$$

A solução do sistema $Ly = b$ é calculada pelas substituições sucessivas (2.7)

$$\begin{bmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ 4 & 3 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 11 \\ -15 \\ 29 \end{bmatrix},$$

$$y_1 = 11,$$

$$-2y_1 + y_2 = -15, \quad y_2 = -15 + 2(11) \rightsquigarrow y_2 = 7 \text{ e}$$

$$4y_1 + 3y_2 + y_3 = 29, \quad y_3 = 29 - 4(11) - 3(7) \rightsquigarrow y_3 = -36.$$

Assim, $y = [11 \ 7 \ -36]^T$. A solução do sistema $Ux = y$, idêntico ao (2.10), é obtida pelas substituições retroativas (2.8)

$$\begin{bmatrix} 1 & -3 & 2 \\ 0 & 2 & 3 \\ 0 & 0 & -12 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ 7 \\ -36 \end{bmatrix},$$

$$-12x_3 = -36, \quad x_3 = \frac{-36}{-12} \rightsquigarrow x_3 = 3,$$

$$2x_2 + 3x_3 = 7, \quad x_2 = \frac{7 - 3(3)}{2} \rightsquigarrow x_2 = -1 \text{ e}$$

$$x_1 - 3x_2 + 2x_3 = 11, \quad x_1 = \frac{11 + 3(-1) - 2(3)}{1} \rightsquigarrow x_1 = 2.$$

O vetor solução do sistema é $x = [2 \ -1 \ 3]^T$, que obviamente é o mesmo obtido pelo método de eliminação de Gauss (ver Exemplo 2.24). ■

A única diferença entre os dispositivos práticos da eliminação de Gauss e da decomposição LU é a ausência da coluna relativa ao vetor b de termos independentes na LU . Na realidade, efetuar as substituições sucessivas para resolver $Ly = b$ na decomposição LU é o mesmo que fazer as operações l-elementares em b na eliminação de Gauss. Desta forma, a solução de $Ly = b$ funciona como uma memória de cálculo para ser efetuada sobre o vetor b . Isto facilita resolver vários sistemas lineares com a mesma matriz dos coeficientes, pois a fatoração da matriz é feita uma única vez. Alguns exemplos serão mostrados na Seção 2.7.

2.4.2 Pivotação parcial

De modo similar ao método de eliminação de Gauss, a estratégia da pivotação parcial deve ser usada na decomposição LU para evitar um pivô nulo e que os multiplicadores m_{ij} tenham valores muito grandes. Quando a pivotação parcial for usada, a decomposição será da forma

$$PA = LU,$$

onde P é uma matriz de permutações, que será constituída das linhas de uma matriz identidade I , colocadas na mesma ordem das linhas pivotais que geram a matriz triangular superior U . A matriz triangular inferior unitária L é formada pelos multiplicadores m_{ij} utilizados na eliminação. A ordem em que os multiplicadores são atribuídos a cada linha de L é dada pelos índices das linhas pivotais. Para resolver o sistema $Ax = b$, tem-se

$$Ax = b \longrightarrow PAx = Pb \longrightarrow LUx = Pb.$$

Fazendo

$$Ux = y, \quad \text{então } Ly = Pb. \quad (2.11)$$

A solução y do sistema triangular inferior $Ly = Pb$ é calculada pelas substituições sucessivas. Por sua vez, o vetor y é utilizado como termo independente no sistema triangular superior $Ux = y$ para obter a solução x pelas substituições retroativas.

Exemplo 2.29 Resolver o sistema do Exemplo 2.27 pela decomposição *LU* usando pivotação parcial

$$\begin{bmatrix} 1 & -3 & 2 \\ -2 & 8 & -1 \\ 4 & -6 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 11 \\ -15 \\ 29 \end{bmatrix}.$$

Utilizando o dispositivo prático, tem-se

| <i>L</i> | <i>m</i> | <i>A</i> | Operações | <i>p</i> |
|----------|----------------------------|---------------|------------------|----------|
| 1 | $m_{11} = 1/4 = 0,25$ | 1 -3 2 | | 1 |
| 2 | $m_{21} = (-2)/4 = -0,5$ | -2 8 -1 | | 2 |
| 3 | | <u>4</u> -6 5 | | <u>3</u> |
| 4 | $m_{12} = (-1,5)/5 = -0,3$ | 0 -1,5 0,75 | $-0,25L_3 + L_1$ | 1 |
| 5 | | 0 5 1,5 | $0,5L_3 + L_2$ | 2 |
| 6 | | 0 0 1,2 | $0,3L_5 + L_4$ | <u>1</u> |

Neste dispositivo, o multiplicador m_{ij} é utilizado para eliminar o elemento da posição (i, j) , e a coluna *p* indica quais são as linhas pivotais determinadas durante a pivotação parcial. A linha pivotal escolhida para eliminar uma coluna é sublinhada e as remanescentes são listadas a seguir para que se faça a escolha da próxima linha pivotal. Este processo se repete até restar apenas uma única linha.

Considerando que *L* é uma matriz triangular inferior unitária, então ela possui todos os elementos da diagonal principal iguais a 1 e todos os elementos acima da diagonal iguais a 0. Cada linha de *L* é constituída pelos multiplicadores relativos a cada uma das linhas pivotais. Mas deve ser notado que não existe multiplicador associado às linhas sublinhadas porque uma linha pivotal não é mais transformada.

Os índices das linhas pivotais estão no vetor $\underline{p} = [3 \ 2 \ 1]$, cujos elementos informam como montar as linhas da matriz *L* a partir dos multiplicadores m_{ij} . A linha 1 de *L* não utiliza multiplicador porque $l_{11} = 1$ e $l_{1j} = 0 \forall j > 1$. A linha 2 de *L* é construída com o multiplicador m_{21} , sendo $i = \underline{p}(2) = 2$, ou seja, $l_{21} = m_{21} = -0,5$. A linha 3 de *L* usa os multiplicadores m_{31} e m_{32} , com $i = \underline{p}(3) = 1$, consequentemente, $l_{31} = m_{31} = 0,25$ e $l_{32} = m_{32} = -0,3$.

$$L = \begin{bmatrix} 1 & 0 & 0 \\ m_{21} & 1 & 0 \\ m_{31} & m_{32} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -0,5 & 1 & 0 \\ 0,25 & -0,3 & 1 \end{bmatrix}.$$

Por sua vez, a matriz *U* é formada simplesmente pelas linhas pivotais. A matriz *P* possui as linhas de uma matriz identidade na ordem das linhas pivotais de $\underline{p} = [3 \ 2 \ 1]$, ou *P* pode ser vista como uma matriz similar à identidade com as linhas colocadas de modo que os elementos iguais a 1 estejam nas colunas relativas aos índices das linhas pivotais

$$U = \begin{bmatrix} 4 & -6 & 5 \\ 0 & 5 & 1,5 \\ 0 & 0 & 1,2 \end{bmatrix} \quad \text{e} \quad P = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

O vetor obtido do produto *Pb* é formado pelos elementos de *b* dispostos na ordem das linhas pivotais contidas em \underline{p} . A solução do sistema $Ly = Pb$ é conseguida pelas substituições sucessivas (2.7)

$$\begin{bmatrix} 1 & 0 & 0 \\ -0,5 & 1 & 0 \\ 0,25 & -0,3 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 29 \\ -15 \\ 11 \end{bmatrix},$$

$$y_1 = 29,$$

$$-0,5y_1 + y_2 = -15, \quad y_2 = -15 + 0,5(29) \rightsquigarrow y_2 = -0,5 \text{ e}$$

$$0,25y_1 - 0,3y_2 + y_3 = 11, \quad y_3 = 11 - 0,25(29) + 0,3(-0,5) \rightsquigarrow y_3 = 3,6.$$

Assim, $y = [29 \ -0,5 \ 3,6]^T$. A solução do sistema $Ux = y$ é obtida pelas substituições retroativas (2.8)

$$\begin{bmatrix} 4 & -6 & 5 \\ 0 & 5 & 1,5 \\ 0 & 0 & 1,2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 29 \\ -0,5 \\ 3,6 \end{bmatrix},$$

$$1,2x_3 = 3,6; \quad x_3 = \frac{3,6}{1,2} \rightsquigarrow x_3 = 3,$$

$$5x_2 + 1,5x_3 = -0,5; \quad x_2 = \frac{-0,5 - 1,5(3)}{5} \rightsquigarrow x_2 = -1 \text{ e}$$

$$4x_1 - 6x_2 + 5x_3 = 29, \quad x_1 = \frac{29 + 6(-1) - 5(3)}{4} \rightsquigarrow x_1 = 2,$$

ou seja, o vetor solução do sistema é $x = [2 \ -1 \ 3]^T$. O vetor resíduo é

$$r = b - Ax = \begin{bmatrix} 11 \\ -15 \\ 29 \end{bmatrix} - \begin{bmatrix} 1 & -3 & 2 \\ -2 & 8 & -1 \\ 4 & -6 & 5 \end{bmatrix} \begin{bmatrix} 2 \\ -1 \\ 3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$$

indicando que a solução *x* é exata.

2.4.3 Cálculo do determinante

Considerando que

$$PA = LU \longrightarrow \det(PA) = \det(LU),$$

então, pela propriedade e dos determinantes vista na Seção 2.3.4,

$$\det(A) = \frac{\det(L) \det(U)}{\det(P)}.$$

Pela propriedade d,

$$\det(L) = \prod_{i=1}^n l_{ii} = 1 \quad \text{e} \quad \det(U) = \prod_{i=1}^n u_{ii} \quad (\text{produto dos pivôs})$$

e, pela propriedade a,

$$\det(P) = (-1)^t,$$

onde t é o número de trocas de linhas necessárias para transformar a matriz de permutações P em uma matriz identidade. Conseqüentemente,

$$\boxed{\det(A) = (-1)^t \prod_{i=1}^n u_{ii}} \quad (2.12)$$

Exemplo 2.30 Calcular o determinante da matriz do Exemplo 2.29

$$A = \begin{bmatrix} 1 & -3 & 2 \\ -2 & 8 & -1 \\ 4 & -6 & 5 \end{bmatrix}.$$

Para calcular o determinante de A por (2.12), deve ser encontrado o valor de t , isto é, o número de trocas de linhas necessárias para transformar a matriz P em uma matriz identidade. Para isso, basta contar quantas permutações precisam ser feitas para colocar os índices das linhas pivotais em ordem crescente

| t | Linhas pivotais | Comentário |
|-----|-----------------|-----------------|
| 0 | 3 2 1 | trocar 3 com 1 |
| 1 | 1 2 3 | ordem crescente |

Deste modo, $t = 1$ e o determinante, dado por (2.12), será

$$\det(A) = (-1)^t \prod_{i=1}^3 u_{ii} = (-1)^1 \times 4 \times 5 \times 1,2 \rightsquigarrow \det(A) = -24.$$

Exemplo 2.31 Resolver o sistema abaixo pela decomposição LU , usando pivotação parcial, e verificar a exatidão e a unicidade da solução

$$\begin{bmatrix} 4 & -1 & 0 & -1 \\ 1 & -2 & 1 & 0 \\ 0 & 4 & -4 & 1 \\ 5 & 0 & 5 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 1 \\ -2 \\ -3 \\ 4 \end{bmatrix}$$

Construindo o dispositivo prático, tem-se

| L | m | A | Operações | p |
|-----|----------------------------|--------------------|-----------------|-----|
| 1 | $m_{11} = 4/5 = 0,8$ | 4 -1 0 -1 | | 1 |
| 2 | $m_{21} = 1/5 = 0,2$ | 1 -2 1 0 | | 2 |
| 3 | $m_{31} = 0/5 = 0$ | 0 4 -4 1 | | 3 |
| 4 | | 5 0 5 -1 | | 4 |
| 5 | $m_{12} = (-1)/4 = -0,25$ | 0 -1 -4 -0,2 | $-0,8L_4 + L_1$ | 1 |
| 6 | $m_{22} = (-2)/4 = -0,5$ | 0 -2 0 0,2 | $-0,2L_4 + L_2$ | 2 |
| 7 | | 0 <u>4</u> -4 1 | $0L_4 + L_3$ | 3 |
| 8 | | 0 0 <u>-5</u> 0,05 | $0,25L_7 + L_5$ | 1 |
| 9 | $m_{23} = (-2)/(-5) = 0,4$ | 0 0 -2 0,7 | $0,5L_7 + L_6$ | 2 |
| 10 | | 0 0 0 0,68 | $-0,4L_8 + L_9$ | 2 |

O vetor p é formado pelos índices das linhas pivotais, isto é, $p = [4 \ 3 \ 1 \ 2]$. A linha 1 de L já está pronta, pois $l_{11} = 1$ e $l_{1j} = 0 \forall j > 1$. A linha 2 de L usa o multiplicador m_{21} , sendo $i = p(2) = 3$, portanto, $l_{21} = m_{31} = 0$. A linha 3 de L é construída com os multiplicadores m_{31} e m_{32} , com $i = p(3) = 1$, assim, $l_{31} = m_{11} = 0,8$ e $l_{32} = m_{12} = -0,25$. Finalmente, a linha 4 de L utiliza os multiplicadores m_{41} , m_{42} e m_{43} , sendo $i = p(4) = 2$, ou seja, $l_{41} = m_{21} = 0,2$, $l_{42} = m_{22} = -0,5$ e $l_{43} = m_{23} = 0,4$.

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 \\ m_{31} & 1 & 0 & 0 \\ m_{11} & m_{12} & 1 & 0 \\ m_{21} & m_{22} & m_{23} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0,8 & -0,25 & 1 & 0 \\ 0,2 & -0,5 & 0,4 & 1 \end{bmatrix}.$$

A matriz U é obtida pelas linhas pivotais (sublinhadas), e a matriz P possui elementos iguais a 1 nas colunas com índices das linhas pivotais contidos em p

$$U = \begin{bmatrix} 5 & 0 & 5 & -1 \\ 0 & 4 & -4 & 1 \\ 0 & 0 & -5 & 0,05 \\ 0 & 0 & 0 & 0,68 \end{bmatrix} \quad \text{e} \quad P = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

O produto da matriz P pelo vetor b é equivalente ao vetor obtido pelos elementos de b dispostos na ordem das linhas pivotais dada em p . A solução do sistema $Ly = Pb$ é dada pelas substituições sucessivas (2.7)

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0,8 & -0,25 & 1 & 0 \\ 0,2 & -0,5 & 0,4 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 4 \\ -3 \\ 1 \\ -2 \end{bmatrix} \rightsquigarrow y = \begin{bmatrix} 4 \\ -3 \\ -2,95 \\ -3,12 \end{bmatrix}.$$

A solução do sistema $Ux = y$ é obtida pelas substituições retroativas (2.8)

$$\begin{bmatrix} 5 & 0 & 5 & -1 \\ 0 & 4 & -4 & 1 \\ 0 & 0 & -5 & 0,05 \\ 0 & 0 & 0 & 0,68 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 4 \\ -3 \\ -2,95 \\ -3,12 \end{bmatrix} \rightsquigarrow x = \begin{bmatrix} -0,6617 \\ 0,9412 \\ 0,5441 \\ -4,5882 \end{bmatrix}.$$

A quase exatidão da solução é verificada pelo vetor resíduo $r = b - Ax$ que para este caso $r = [-0,0002 \ 0,0000 \ -0,0002 \ -0,0002]^T$. A unicidade da solução é confirmada por intermédio do cálculo do determinante. Para isto, é necessário encontrar t , contando quantas permutações precisam ser feitas para colocar os índices das linhas pivotais em ordem crescente

| t | Linhas pivotais | | | | Comentário |
|-----|-----------------|---|---|---|-----------------|
| 0 | 4 | 3 | 1 | 2 | trocar 4 com 1 |
| 1 | 1 | 3 | 4 | 2 | trocar 3 com 2 |
| 2 | 1 | 2 | 4 | 3 | trocar 4 com 3 |
| 3 | 1 | 2 | 3 | 4 | ordem crescente |

Assim, $t = 3$ e o determinante, dado por (2.12), será

$$\det(A) = (-1)^t \prod_{i=1}^4 u_{ii},$$

$$\det(A) = (-1)^3 \times 5 \times 4 \times -5 \times 0,68 = 68 \neq 0 \rightarrow \text{solução única.}$$

2.4.4 Sistema com matriz singular

Conforme visto na Seção 2.1.6, quando a matriz dos coeficientes A de um sistema linear $Ax = b$ for singular, isto é, $\det(A) = 0$, então este sistema pode ter infinitas soluções ou não ter solução. Será mostrado, por um exemplo, como diferenciar essas duas situações.

Exemplo 2.32 Resolver os sistemas $Ax = b$ e $Ax = c$ usando decomposição LU com pivotação parcial, sendo

$$A = \begin{bmatrix} 1 & -3 & 2 \\ -2 & 8 & -1 \\ -1 & 5 & 1 \end{bmatrix}, b = \begin{bmatrix} 22 \\ -12 \\ 10 \end{bmatrix} \quad \text{e} \quad c = \begin{bmatrix} 20 \\ -10 \\ 80 \end{bmatrix}.$$

As representações gráficas dos sistemas $Ax = b$ e $Ax = c$ são exibidas nas Figuras 2.2(a) e (b) na página 44. Utilizando o dispositivo prático, tem-se

| L | m | A | Operações | p |
|-----|----------------------------|---------------------------|-----------------|-----|
| 1 | $m_{11} = 1/(-2) = -0,5$ | $1 \ -3 \ 2$ | | 1 |
| 2 | | $\underline{-2} \ 8 \ -1$ | | 2 |
| 3 | $m_{31} = (-1)/(-2) = 0,5$ | $-1 \ 5 \ 1$ | | 3 |
| 4 | | $0 \ 1 \ 1,5$ | $0,5L_2 + L_1$ | 1 |
| 5 | $m_{32} = 1/1 = 1$ | $0 \ 1 \ 1,5$ | $-0,5L_2 + L_3$ | 3 |
| 6 | | $0 \ 0 \ 0$ | $-L_4 + L_5$ | 3 |

Assim, os três fatores são

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -0,5 & 1 & 0 \\ 0,5 & 1 & 1 \end{bmatrix}, U = \begin{bmatrix} -2 & 8 & -1 \\ 0 & 1 & 1,5 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{e} \quad P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

A solução do sistema $Ly = Pb$ é obtida pelas substituições sucessivas (2.7)

$$\begin{bmatrix} 1 & 0 & 0 \\ -0,5 & 1 & 0 \\ 0,5 & 1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} -12 \\ 22 \\ 10 \end{bmatrix} \rightsquigarrow y = \begin{bmatrix} -12 \\ 16 \\ 0 \end{bmatrix}.$$

A solução do sistema $Ux = y$ é obtida pelas substituições retroativas (2.8)

$$\begin{bmatrix} -2 & 8 & -1 \\ 0 & 1 & 1,5 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -12 \\ 16 \\ 0 \end{bmatrix},$$

$$0x_3 = 0 \rightsquigarrow x_3 = \theta \quad (\text{qualquer valor de } x_3 \text{ é solução}),$$

$$x_2 + 1,5x_3 = 16 \rightsquigarrow x_2 = 16 - 1,5\theta \text{ e}$$

$$-2x_1 + 8x_2 - x_3 = -12, \quad x_1 = \frac{-12 - 8(16 - 1,5\theta) + \theta}{-2} \rightsquigarrow x_1 = 70 - 6,5\theta.$$

Conseqüentemente, o vetor solução do sistema é $x = [70 - 6,5\theta \ 16 - 1,5\theta \ \theta]^T$, ou seja, o sistema $Ax = b$ apresenta infinitas soluções, uma para cada valor de θ .

Para $Ax = c$, a solução de $Ly = Pc$ é

$$\begin{bmatrix} 1 & 0 & 0 \\ -0,5 & 1 & 0 \\ 0,5 & 1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} -10 \\ 20 \\ 80 \end{bmatrix} \rightsquigarrow y = \begin{bmatrix} -10 \\ 15 \\ 70 \end{bmatrix}$$

e a solução de $Ux = y$ é

$$\begin{bmatrix} -2 & 8 & -1 \\ 0 & 1 & 1,5 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -10 \\ 15 \\ 70 \end{bmatrix}$$

$$0x_3 = 70 \rightarrow \nexists x_3 \rightsquigarrow \nexists x.$$

Deste modo, o sistema $Ax = c$ não tem solução porque $\nexists x_3$ tal que $0x_3 \neq 0$.

2.4.5 Algoritmos e complexidade

A Figura 2.5 apresenta um algoritmo para a decomposição LU de uma matriz A , via método de eliminação de Gauss com pivotação parcial. Os parâmetros de entrada são a ordem n e a matriz A . Os parâmetros de saída são as matrizes $L \setminus U$ escritas sobre A , o determinante Det de A e o vetor $Pivot$ contendo as linhas pivotais.

```

Algoritmo Decomposição LU
{ Objetivo: Fazer a decomposição LU de uma matriz A }
parâmetros de entrada n, A { ordem e matriz a ser decomposta }
parâmetros de saída A, Det, Pivot
{ matriz decomposta A = U + L = I, determinante, pivôs }
para i ← 1 até n faça Pivot(i) ← i, fim para; Det ← 1
para j ← 1 até n - 1 faça
{ escolha do elemento pivô }
p ← j; Amax ← abs(A(j,j))
para k ← j + 1 até n faça
se abs(A(k,j)) > Amax então
Amax ← abs(A(k,j)); p ← k
fim se
fim para
se p ≠ j então
{ troca de linhas }
para k ← 1 até n faça
t ← A(j,k); A(j,k) ← A(p,k); A(p,k) ← t
fim para
m ← Pivot(j); Pivot(j) ← Pivot(p); Pivot(p) ← m
Det ← -Det
fim se
Det ← Det * A(j,j)
se abs(A(j,j)) ≠ 0 então
{ eliminação de Gauss }
r ← 1/A(j,j)
para j ← j + 1 até n faça
Mult ← A(i,j) * r; A(i,j) ← Mult
para k ← j + 1 até n faça
A(i,k) ← A(i,k) - Mult * A(j,k)
fim para
fim para
fim se
fim para
Det ← Det * A(n,n)
finalgoritmo

```

Figura 2.5 Decomposição LU por eliminação de Gauss com pivotação parcial.
(Ver significado da função abs na Tabela 1.1, na página 6.)

Neste algoritmo, as matrizes triangulares L e U são escritas sobre a matriz original A

A , antes da decomposição → A , após a decomposição

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & a_{nn} \end{bmatrix} \rightarrow \begin{bmatrix} u_{11} & u_{12} & u_{13} & \cdots & u_{1n} \\ l_{21} & u_{22} & u_{23} & \cdots & u_{2n} \\ l_{31} & l_{32} & u_{33} & \cdots & u_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & l_{n3} & \cdots & u_{nn} \end{bmatrix}.$$

Como a matriz A é transformada em $L \setminus U$, se A não puder ser destruída durante a decomposição, então ela deve ser previamente copiada em uma outra matriz. Na decomposição LU , a matriz L é triangular inferior unitária, ou seja, os elementos da diagonal principal são iguais a 1.

Além disso, como o esquema da pivotação parcial é utilizado, então o algoritmo das substituições sucessivas para solução de $Ly = b$ deve ser modificado para resolver $Ly = Pb$ de (2.11), tal como o algoritmo apresentado na Figura 2.6. Os parâmetros de entrada são a ordem n do sistema, a matriz triangular inferior L , o vetor de termos independentes b e o vetor $Pivot$ contendo as linhas pivotais. O parâmetro de saída é o vetor solução y .

Algoritmo Substituições_Sucessivas_Pivotal

```

{ Objetivo: Resolver o sistema triangular inferior Ly = Pb }
{ pelas substituições sucessivas, com a matriz L }
{ obtida de decomposição LU com pivotação parcial }
parâmetros de entrada n, L, b, Pivot
{ ordem, matriz triangular inferior unitária, }
{ vetor independente e posição dos pivôs }
parâmetros de saída y { solução do sistema triangular inferior }
k ← Pivot(1); y(1) ← b(k)
para i ← 2 até n faça
Soma ← 0
para j ← 1 até i - 1 faça
Soma ← Soma + L(i,j) * y(j)
fim para
k ← Pivot(i); y(i) ← b(k) - Soma
fim para
finalgoritmo

```

Figura 2.6 Solução do sistema triangular inferior $Ly = Pb$, de LU com pivotação parcial.

A solução de $Ux = y$ de (2.11) é obtida pelas substituições retroativas, usando o algoritmo da Figura 2.4.

A Tabela 2.4 mostra a complexidade computacional do algoritmo da Figura 2.5. Deve ser mencionado que não foram consideradas as operações de trocas de sinal e multiplicações necessárias para o cálculo do determinante.

Tabela 2.4 Complexidade da decomposição LU de uma matriz de ordem n .
(Desconsiderando operações para o cálculo do determinante.)

| Operações | Complexidade |
|----------------|--|
| adições | $\frac{1}{3}n^3 - \frac{1}{2}n^2 + \frac{1}{6}n$ |
| multiplicações | $\frac{1}{3}n^3 - \frac{1}{3}n$ |
| divisões | $n - 1$ |

A complexidade computacional do algoritmo de substituições sucessivas pivotal difere daquele apresentado na Figura 2.3 somente quanto ao número de divisões, que é nulo, pois, como a matriz L neste caso é unitária, não há necessidade de divisão.

Exemplo 2.33 Resolver o sistema do Exemplo 2.31 usando os algoritmos da Figura 2.5, Figura 2.6 e Figura 2.4. Comparar os resultados intermediários com os valores obtidos naquele exemplo: matriz $L \setminus U$ em A, matriz P em Pivot, determinante em Det, vetor y em y e solução x em x.

```
% Os valores de entrada
n = 4
A =
 4   -1   0   -1
 1   -2   1   0
 0   4   -4   1
 5   0   5   -1
% produzem os resultados pela decomposicao LU
A =
 5.0000      0   5.0000  -1.0000
      0   4.0000  -4.0000   1.0000
  0.8000  -0.2500  -5.0000   0.0500
  0.2000  -0.5000   0.4000   0.6800
Det = 68.0000
Pivot = 4   3   1   2
% vetor de termos independentes
b =
 1
-2
-3
 4
% As substituições sucessivas pivotal produzem
y =
 4.0000
-3.0000
-2.9500
```

```
-3.1200
% As substituições retroativas resultam em
x =
-0.6618
 0.9412
 0.5441
-4.5882
```

2.4.6 Sistemas lineares complexos

Os sistemas de equações lineares que envolvam números complexos podem ser solucionados pelos algoritmos apresentados neste capítulo. Eles são resolvidos tanto pelos algoritmos implementados em uma linguagem de programação que suporta aritmética complexa quanto pelos algoritmos implementados com aritmética real. Mas, para este caso, o sistema complexo deve ser previamente transformado em um sistema real.

Para resolver um sistema complexo usando aritmética real, faz-se necessária uma transformação. Seja o sistema complexo $Ax = b$. Fazendo $A = A_r + iA_i$, $x = x_r + ix_i$, $b = b_r + ib_i$ e substituindo na equação acima, tem-se

$$(A_r + iA_i)(x_r + ix_i) = b_r + ib_i,$$

$$A_r x_r - A_i x_i + i(A_i x_r + A_r x_i) = b_r + ib_i.$$

Na forma matricial

$$\begin{bmatrix} A_r & -A_i \\ A_i & A_r \end{bmatrix} \begin{bmatrix} x_r \\ x_i \end{bmatrix} = \begin{bmatrix} b_r \\ b_i \end{bmatrix}. \quad (2.13)$$

Exemplo 2.34 Resolver o sistema abaixo, utilizando os algoritmos da Figura 2.4, Figura 2.5 e Figura 2.6 implementados em uma linguagem com aritmética complexa.

$$\begin{bmatrix} 1+2i & -3i & 5 \\ 2+3i & 1+i & 1-i \\ 4 & 2i & 3-2i \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 10-16i \\ -5+12i \\ 13+2i \end{bmatrix}.$$

```
% Os valores de entrada
n = 3
A =
 1.0000+ 2.0000i      0- 3.0000i      5.0000
 2.0000+ 3.0000i     1.0000+ 1.0000i  1.0000- 1.0000i
 4.0000                  0+ 2.0000i     3.0000- 2.0000i
% Produzem os resultados pela decomposicao LU
A =
 4.0000                  0+ 2.0000i     3.0000- 2.0000i
 0.2500+ 0.5000i     1.0000- 3.5000i  3.2500- 1.0000i
 0.5000+ 0.7500i     0.1887+ 0.6604i -3.2736- 4.2075i
Det =
-72.0000+29.0000i
Pivot =
```

```

    3   1   2
% vetor de termos independentes
b =
10.0000-16.0000i
-5.0000+12.0000i
13.0000+ 2.0000i
% As substituições sucessivas pivotal produzem
y =
13.0000+ 2.0000i
7.7500-23.0000i
-26.6509+ 0.4717i
% As substituições retroativas resultam em
x =
3.0000+ 4.0000i
2.0000+ 0.0000i
3.0000- 4.0000i

```

O vetor solução é

$$x = \begin{bmatrix} 3+4i \\ 2 \\ 3-4i \end{bmatrix}.$$

Exemplo 2.35 Resolver o sistema do Exemplo 2.34, utilizando os algoritmos da Figura 2.4, Figura 2.5 e Figura 2.6 implementados em uma linguagem que não tem aritmética complexa.

Por (2.13), o sistema complexo pode ser resolvido por meio do sistema real

$$\left[\begin{array}{cccccc} 1 & 0 & 5 & -2 & 3 & 0 \\ 2 & 1 & 1 & -3 & -1 & 1 \\ 4 & 0 & 3 & 0 & -2 & 2 \\ 2 & -3 & 0 & 1 & 0 & 5 \\ 3 & 1 & -1 & 2 & 1 & 1 \\ 0 & 2 & -2 & 4 & 0 & 3 \end{array} \right] \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \right] = \left[\begin{array}{c} 10 \\ -5 \\ 13 \\ -16 \\ 12 \\ 2 \end{array} \right]$$

% Os valores de entrada

```

n = 6
A =
1   0   5   -2   3   0
2   1   1   -3   -1   1
4   0   3   0   -2   2
2   -3   0   1   0   5
3   1   -1   2   1   1
0   2   -2   4   0   3
% produzem os resultados pela decomposição LU
A =
4.0000      0   3.0000      0   -2.0000   2.0000
0.5000  -3.0000  -1.5000  1.0000   1.0000   4.0000
0.2500      0   4.2500  -2.0000   3.5000  -0.5000
0   -0.6667  -0.7059  3.2549   3.1373   5.3137
0.7500  -0.3333  -0.8824  0.1747   5.3735  -0.5361
0.5000  -0.3333  -0.2353  -0.9639   0.7780   6.7545

```

```

Det =
6.0250e+03
Pivot =
3   4   1   6   5   2
% vetor de termos independentes
b =
10
-5
13
-16
12
2
% As substituições sucessivas pivotal produzem
y =
13.0000
-22.5000
6.7500
-8.2353
2.1446
-27.0179
% As substituições retroativas resultam em
x =
3.0000
2.0000
3.0000
4.0000
0.0000
-4.0000

```

Portanto, o vetor solução é

$$x = \begin{bmatrix} 3+4i \\ 2 \\ 3-4i \end{bmatrix}.$$

2.5 Decomposição de Cholesky e LDL^T

Quando a matriz dos coeficientes A for simétrica e definida positiva, ou seja, sua forma quadrática $v^T A v > 0$, $\forall v \neq 0$ (ver Tabela 2.1), então A pode ser decomposta tal que

$$A = LL^T,$$

onde L é uma matriz triangular inferior e, consequentemente, L^T será triangular superior. A existência da decomposição de Cholesky é garantida pelo Teorema 2.2 [20, Teorema 4.2.5].

Teorema 2.2 (Cholesky) Se A for uma matriz simétrica e definida positiva, então existe uma única matriz triangular L com elementos da diagonal positivos tal que $A = LL^T$.

2.5.1 Cálculo do fator

Seja o produto $LL^T = A$ de uma matriz, por exemplo, de ordem 4

$$\begin{bmatrix} l_{11} & 0 & 0 & 0 \\ l_{21} & l_{22} & 0 & 0 \\ l_{31} & l_{32} & l_{33} & 0 \\ l_{41} & l_{42} & l_{43} & l_{44} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & l_{31} & l_{41} \\ 0 & l_{22} & l_{32} & l_{42} \\ 0 & 0 & l_{33} & l_{43} \\ 0 & 0 & 0 & l_{44} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix}$$

O elemento l_{44} da diagonal é obtido por

$$l_{41}^2 + l_{42}^2 + l_{43}^2 + l_{44}^2 = a_{44} \rightarrow l_{44} = \sqrt{a_{44} - (l_{41}^2 + l_{42}^2 + l_{43}^2)} \sim l_{44} = \sqrt{a_{44} - \sum_{k=1}^3 l_{4k}^2}$$

Esta expressão pode ser generalizada para um elemento qualquer da diagonal de L

$$l_{jj} = \sqrt{a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2}, \quad j = 1, 2, \dots, n. \quad (2.14)$$

O elemento l_{43} abaixo da diagonal é calculado por intermédio de

$$l_{41}l_{31} + l_{42}l_{32} + l_{43}l_{33} = a_{43} \rightarrow l_{43} = \frac{a_{43} - (l_{41}l_{31} + l_{42}l_{32})}{l_{33}} \sim l_{43} = \frac{a_{43} - \sum_{k=1}^2 l_{4k}l_{3k}}{l_{33}}$$

Para obter um elemento genérico abaixo da diagonal principal

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik}l_{jk}}{l_{ii}}, \quad j = 1, 2, \dots, n-1 \text{ e } i = j+1, j+2, \dots, n. \quad (2.15)$$

Os elementos de L são computados por (2.14) e (2.15), com as operações podendo ser efetuadas ou por linha ou por coluna. Um elemento da diagonal de L é, por (2.14), a raiz quadrada da diferença entre o correspondente elemento da diagonal de A e o somatório dos quadrados dos elementos da mesma linha, à qual pertence o elemento da diagonal de L que se deseja obter. Para calcular um elemento l_{ij} por (2.15), toma-se o somatório do produto de dois a dois dos elementos pertencentes às linhas i e j de L que estejam da coluna 1 até a coluna $j-1$. Em seguida, calcula-se a diferença do correspondente elemento de A e aquele somatório. E, finalmente, obtém-se a razão entre esta diferença e o elemento situado na diagonal de L .

Devido à estabilidade numérica da decomposição de uma matriz simétrica definida positiva, não se faz necessário o uso da pivotação parcial na decomposição de Cholesky. De maneira similar à decomposição LU , o sistema de equações lineares $Ax = b$ é resolvido por

$$Ax = b \rightarrow LL^T x = b.$$

Fazendo

$$L^T x = y \text{ então } Ly = b. \quad (2.16)$$

O sistema triangular inferior $Ly = b$ é resolvido pelas substituições sucessivas na forma

$$y_i = \frac{b_i - \sum_{j=1}^{i-1} l_{ij}y_j}{l_{ii}}, \quad i = 1, 2, \dots, n. \quad (2.17)$$

O vetor solução do sistema triangular superior $L^T x = y$ é obtido pelas substituições retroativas na forma

$$x_i = \frac{y_i - \sum_{j=i+1}^n l_{ji}x_j}{l_{ii}}, \quad i = n, n-1, \dots, 1. \quad (2.18)$$

2.5.2 Cálculo do determinante

Para calcular o determinante pela decomposição de Cholesky basta considerar as propriedades d e e da Seção 2.3.4

$$\det(A) = \det(L) \det(L^T) \rightarrow \det(A) = \left(\prod_{i=1}^n l_{ii} \right)^2. \quad (2.19)$$

Exemplo 2.36 Resolver o sistema abaixo usando a decomposição de Cholesky e verificar a exatidão e unicidade da solução

$$\begin{bmatrix} 4 & -2 & 2 \\ -2 & 10 & -7 \\ 2 & -7 & 30 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 11 \\ -31 \end{bmatrix}.$$

Pelo uso de (2.14) e (2.15), obtém-se

Coluna 1:

$$l_{11} = \sqrt{a_{11}} = \sqrt{4} = 2, \quad l_{21} = \frac{a_{21}}{l_{11}} = \frac{-2}{2} = -1, \quad l_{31} = \frac{a_{31}}{l_{11}} = \frac{2}{2} = 1.$$

Coluna 2:

$$l_{22} = \sqrt{a_{22} - l_{21}^2} = \sqrt{10 - (-1)^2} = 3, \quad l_{32} = \frac{a_{32} - l_{31}l_{21}}{l_{22}} = \frac{-7 - (1)(-1)}{3} = -2.$$

Coluna 3:

$$l_{33} = \sqrt{a_{33} - (l_{31}^2 + l_{32}^2)} = \sqrt{30 - ((1)^2 + (-2)^2)} = 5.$$

Estes resultados podem ser sumarizados no dispositivo prático

| A | | | L | | | | |
|-----|----|----|----|-----|----|----|---|
| i\j | 1 | 2 | 3 | i\j | 1 | 2 | 3 |
| 1 | 4 | | | 1 | 2 | | |
| 2 | -2 | 10 | | 2 | -1 | 3 | |
| 3 | 2 | -7 | 30 | 3 | 1 | -2 | 5 |

Deve ser observado que $LL^T = A$

$$\begin{bmatrix} 2 & 0 & 0 \\ -1 & 3 & 0 \\ 1 & -2 & 5 \end{bmatrix} \begin{bmatrix} 2 & -1 & 1 \\ 0 & 3 & -2 \\ 0 & 0 & 5 \end{bmatrix} = \begin{bmatrix} 4 & -2 & 2 \\ -2 & 10 & -7 \\ 2 & -7 & 30 \end{bmatrix}.$$

Resolvendo o sistema $Ly = b$ pelas substituições sucessivas (2.17)

$$\begin{bmatrix} 2 & 0 & 0 \\ -1 & 3 & 0 \\ 1 & -2 & 5 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 8 \\ 11 \\ -31 \end{bmatrix} \rightsquigarrow y = \begin{bmatrix} 4 \\ 5 \\ -5 \end{bmatrix}.$$

Resolvendo o sistema $L^T x = y$ pelas substituições retroativas (2.18)

$$\begin{bmatrix} 2 & -1 & 1 \\ 0 & 3 & -2 \\ 0 & 0 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 5 \\ -5 \end{bmatrix} \rightsquigarrow x = \begin{bmatrix} 3 \\ 1 \\ -1 \end{bmatrix}.$$

A exatidão da solução é verificada pelo vetor resíduo $r = b - Ax$

$$r = \begin{bmatrix} 8 \\ 11 \\ -31 \end{bmatrix} - \begin{bmatrix} 4 & -2 & 2 \\ -2 & 10 & -7 \\ 2 & -7 & 30 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

A unicidade da solução é confirmada por intermédio do cálculo do determinante por (2.19)

$$\det(A) = \left(\prod_{i=1}^3 l_{ii} \right)^2 = ((2)(3)(5))^2 = 900 \neq 0 \rightarrow \text{solução única.}$$

Exemplo 2.37 Resolver o sistema abaixo usando a decomposição de Cholesky e verificar a exatidão e unicidade da solução

$$\begin{bmatrix} 9 & 6 & -3 & 3 \\ 6 & 20 & 2 & 22 \\ -3 & 2 & 6 & 2 \\ 3 & 22 & 2 & 28 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 64 \\ 4 \\ 82 \end{bmatrix}.$$

Por (2.14) e (2.15), têm-se

Coluna 1:

$$l_{11} = \sqrt{a_{11}} = \sqrt{9} = 3, \quad l_{21} = \frac{a_{21}}{l_{11}} = \frac{6}{3} = 2, \quad l_{31} = \frac{a_{31}}{l_{11}} = \frac{-3}{3} = -1,$$

$$l_{41} = \frac{a_{41}}{l_{11}} = \frac{3}{3} = 1.$$

Coluna 2:

$$l_{22} = \sqrt{a_{22} - l_{21}^2} = \sqrt{20 - (2)^2} = 4, \quad l_{32} = \frac{a_{32} - l_{31}l_{21}}{l_{22}} = \frac{2 - (-1)(2)}{4} = 1,$$

$$l_{42} = \frac{a_{42} - l_{41}l_{21}}{l_{22}} = \frac{22 - (1)(2)}{4} = 5.$$

Coluna 3:

$$l_{33} = \sqrt{a_{33} - (l_{31}^2 + l_{32}^2)} = \sqrt{6 - ((-1)^2 + (1)^2)} = 2,$$

$$l_{43} = \frac{a_{43} - (l_{41}l_{31} + l_{42}l_{32})}{l_{33}} = \frac{2 - ((1)(-1) + (5)(1))}{2} = -1.$$

Coluna 4:

$$l_{44} = \sqrt{a_{44} - (l_{41}^2 + l_{42}^2 + l_{43}^2)} = \sqrt{28 - ((1)^2 + (5)^2 + (-1)^2)} = 1.$$

Os resultados acima estão compilados no dispositivo prático

| A | | | | L | | | | | |
|-----|----|----|---|----|-----|----|---|----|---|
| i\j | 1 | 2 | 3 | 4 | i\j | 1 | 2 | 3 | 4 |
| 1 | 9 | | | | 1 | 3 | | | |
| 2 | 6 | 20 | | | 2 | 2 | 4 | | |
| 3 | -3 | 2 | 6 | | 3 | -1 | 1 | 2 | |
| 4 | 3 | 22 | 2 | 28 | 4 | 1 | 5 | -1 | 1 |

Resolvendo o sistema $Ly = b$ pelas substituições sucessivas (2.17)

$$\begin{bmatrix} 3 & 0 & 0 & 0 \\ 2 & 4 & 0 & 0 \\ -1 & 1 & 2 & 0 \\ 1 & 5 & -1 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} 12 \\ 64 \\ 4 \\ 82 \end{bmatrix} \rightsquigarrow y = \begin{bmatrix} 4 \\ 14 \\ -3 \\ 5 \end{bmatrix}.$$

Resolvendo o sistema $L^T x = y$ pelas substituições retroativas (2.18)

$$\begin{bmatrix} 3 & 2 & -1 & 1 \\ 0 & 4 & 1 & 5 \\ 0 & 0 & 2 & -1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 4 \\ 14 \\ -3 \\ 5 \end{bmatrix} \rightsquigarrow x = \begin{bmatrix} 2 \\ -3 \\ 1 \\ 5 \end{bmatrix}.$$

A solução é exata, pois o vetor resíduo $r = b - Ax = [0 \ 0 \ 0 \ 0]^T$. Quanto à unicidade, por (2.19)

$$\det(A) = \left(\prod_{i=1}^4 l_{ii} \right)^2 = ((3)(4)(2)(1))^2 = 576 \neq 0 \rightarrow \text{solução única.}$$

Exemplo 2.38 Resolver o sistema abaixo usando a decomposição de Cholesky e verificar a exatidão e unicidade da solução

$$\begin{bmatrix} 5 & -1 & 2 \\ -1 & 8 & 4 \\ 2 & 4 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 21 \\ 10 \\ 50 \end{bmatrix}.$$

Pelo uso de (2.14) e (2.15), obtém-se

Coluna 1:

$$l_{11} = \sqrt{a_{11}} = \sqrt{5} = 2,2361;$$

$$l_{21} = \frac{a_{21}}{l_{11}} = \frac{-1}{2,2361} = -0,4472; l_{31} = \frac{a_{31}}{l_{11}} = \frac{2}{2,2361} = 0,8944.$$

Coluna 2:

$$l_{22} = \sqrt{a_{22} - l_{21}^2} = \sqrt{8 - (-0,4472)^2} = 2,7929;$$

$$l_{32} = \frac{a_{32} - l_{31}l_{21}}{l_{22}} = \frac{4 - (0,8944)(-0,4472)}{2,7929} = 1,5754.$$

Coluna 3:

$$l_{33} = \sqrt{a_{33} - (l_{31}^2 + l_{32}^2)} = \sqrt{10 - ((0,8944)^2 + (1,5754)^2)} = 2,5919.$$

Esses resultados podem ser listados no dispositivo prático

| A | | | L | | | | |
|-----|----|---|----|-----|---------|--------|--------|
| i\j | 1 | 2 | 3 | i\j | 1 | 2 | 3 |
| 1 | 5 | | | 1 | 2,2361 | | |
| 2 | -1 | 8 | | 2 | -0,4472 | 2,7929 | |
| 3 | 2 | 4 | 10 | 3 | 0,8944 | 1,5754 | 2,5919 |

Resolvendo o sistema $Ly = b$ pelas substituições sucessivas (2.17)

$$\begin{bmatrix} 2,2361 & 0 & 0 \\ -0,4472 & 2,7929 & 0 \\ 0,8944 & 1,5754 & 2,5919 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 21 \\ 10 \\ 50 \end{bmatrix} \rightsquigarrow y = \begin{bmatrix} 9,3914 \\ 5,0843 \\ 12,9598 \end{bmatrix}.$$

Resolvendo o sistema $L^T x = y$ pelas substituições retroativas (2.18)

$$\begin{bmatrix} 2,2361 & -0,4472 & 0,8944 \\ 0 & 2,7929 & 1,5754 \\ 0 & 0 & 2,5919 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 9,3914 \\ 5,0843 \\ 12,9598 \end{bmatrix} \rightsquigarrow x = \begin{bmatrix} 2,0000 \\ -1,0000 \\ 5,0001 \end{bmatrix}.$$

A exatidão da solução é verificada pelo vetor resíduo $r = b - Ax$

$$r = \begin{bmatrix} 21 \\ 10 \\ 50 \end{bmatrix} - \begin{bmatrix} 5 & -1 & 2 \\ -1 & 8 & 4 \\ 2 & 4 & 10 \end{bmatrix} \begin{bmatrix} 2,0000 \\ -1,0000 \\ 5,0001 \end{bmatrix} = \begin{bmatrix} -0,0002 \\ -0,0004 \\ -0,0010 \end{bmatrix},$$

significando que a solução não é exata devido aos erros de arredondamento. A unicidade da solução é verificada por meio do cálculo do determinante por (2.19)

$$\det(A) = \left(\prod_{i=1}^3 l_{ii} \right)^2 \approx ((2,2361)(2,7929)(2,5919))^2,$$

$$\det(A) \approx 262,0171 \neq 0 \rightarrow \text{solução única.}$$

2.5.3 Algoritmo e complexidade

A Figura 2.7 apresenta um algoritmo para fatorar uma matriz A simétrica definida positiva, tal que $A = LL^T$, usando a decomposição de Cholesky. Os parâmetros de entrada são a ordem n e a parte triangular inferior da matriz A . Os parâmetros de saída são o fator L escrito sobre A , o determinante Det e a condição de erro $CondErro$. A condição $CondErro = 0$ significa que a decomposição foi realizada com sucesso, enquanto $CondErro = 1$ significa que não foi porque a matriz não é definida positiva. As substituições sucessivas e retroativas necessárias para resolver os sistemas triangulares $Ly = b$ e $L^T x = y$ de (2.16) podem ser computadas pelos algoritmos mostrados nas Figuras 2.3 e 2.4.

Exemplo 2.39 Resolver o sistema do Exemplo 2.37 usando os algoritmos da Figura 2.7, Figura 2.3 e Figura 2.4. Observar que a condição $CondErro = 0$ indica que não houve erro na decomposição, pois a matriz A é, de fato, definida positiva.

```
% Os valores de entrada
n = 4
A =
  9
  6  20
 -3  2   6
  3  22   2   28
% produzem os resultados pela decomposicao de Cholesky
A =
  3
  2   4
 -1   1   2
  1   5   -1   1
Det = 576
CondErro = 0
% vetor de termos independentes
b =
  12
  64
   4
  82
% As substituicoes sucessivas resultam em
y =
   4
  14
  -3
```

```

5
% As substituições retroativas produzem
x =
2
-3
1
5

```

Algoritmo Cholesky{ Objetivo: Fazer a decomposição LL^T de uma matriz A }

{ simétrica e definida positiva }

parâmetros de entrada n, A { ordem e matriz a ser decomposta }parâmetros de saída $A, Det, Cond\text{erro}$ { fator L escrito sobre A , determinante e condição de erro } $Cond\text{erro} \leftarrow 0; Det \leftarrow 1$ para $j \leftarrow 1$ até n faça Soma $\leftarrow 0$ para $k \leftarrow 1$ até $j - 1$ faça Soma $\leftarrow Soma + A(j, k)^2$

fim para

 $t \leftarrow A(j, j) - Soma$ se $t > 0$ então $A(j, j) \leftarrow \text{raiz}_2(t); r \leftarrow 1/A(j, j); Det \leftarrow Det * t$

senão

 $Cond\text{erro} \leftarrow 1$, escreva "a matriz não é definida positiva", abandone

fim se

 para $i \leftarrow j + 1$ até n faça Soma $\leftarrow 0$ para $k \leftarrow 1$ até $j - 1$ faça Soma $\leftarrow Soma + A(i, k) * A(j, k)$

fim para

 $A(i, j) \leftarrow (A(i, j) - Soma) * r$

fim para

fim para

finalgoritmo

Figura 2.7 Decomposição de Cholesky da matriz A simétrica definida positiva.(Ver significado da função raiz_2 na Tabela 1.1, na página 6.)

A complexidade computacional do algoritmo da Figura 2.7 é mostrada na Tabela 2.5, sen-
do a operação de potenciação computada como uma multiplicação e não incluindo as n
multiplicações efetuadas para o cálculo do determinante.

Tabela 2.5 Complexidade da decomposição de Cholesky de matriz de ordem n .

(Desconsiderando as multiplicações para o cálculo do determinante.)

| Operações | Complexidade |
|------------------|--|
| adições | $\frac{1}{6}n^3 + \frac{1}{2}n^2 + \frac{1}{3}n$ |
| multiplicações | $\frac{1}{6}n^3 + \frac{1}{2}n^2 - \frac{2}{3}n$ |
| divisões | n |
| raízes quadradas | n |

2.5.4 Fatoração LDL^T

Uma matriz A simétrica pode ser decomposta, tal que $A = LDL^T$, onde L é uma matriz triangular inferior unitária ($l_{jj} = 1, \forall j$) e D é uma matriz diagonal. A matriz D é computada por

$$d_{jj} = a_{jj} - \sum_{k=1}^{j-1} l_{jk}^2 d_{kk}, \quad j = 1, 2, \dots, n. \quad (2.20)$$

Por sua vez, a matriz unitária L é obtida por

$$l_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} l_{ik} d_{kk} l_{jk}}{d_{jj}}, \quad j = 1, 2, \dots, n-1 \text{ e } i = j+1, j+2, \dots, n. \quad (2.21)$$

A solução do sistema de equações lineares $Ax = b$ é obtida por

$$Ax = b \longrightarrow LDL^T x = b.$$

Fazendo

$$L^T x = t \text{ e } Dt = y \text{ então } Ly = b.$$

O sistema triangular inferior unitário $Ly = b$ é resolvido pelas substituições sucessivas. A solução do sistema diagonal $Dt = y$ é, simplesmente, $t_i = y_i/D_{ii}$, e o vetor solução do sistema triangular superior unitário $L^T x = t$ é obtido pelas substituições retroativas.

Pelas propriedades d e e da Seção 2.3.4 o determinante é dado por

$$\det(A) = \det(L) \det(D) \det(L^T) \longrightarrow \det(A) = \prod_{i=1}^n d_{ii}. \quad (2.22)$$

Exemplo 2.40 Resolver o sistema do Exemplo 2.38 usando a decomposição de LDL^T e verificar a exatidão e unicidade da solução

$$\begin{bmatrix} 5 & -1 & 2 \\ -1 & 8 & 4 \\ 2 & 4 & 10 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 21 \\ 10 \\ 50 \end{bmatrix}.$$

Pelo uso de (2.20) e (2.21), obtém-se

Coluna 1:

$$d_{11} = a_{11} = 5, \quad l_{21} = \frac{a_{21}}{d_{11}} = \frac{-1}{5} = -0,2; \quad l_{31} = \frac{a_{31}}{d_{11}} = \frac{2}{5} = 0,4.$$

Coluna 2:

$$d_{22} = a_{22} - l_{21}^2 d_{11} = 8 - (-0,2)^2(5) = 7,8;$$

$$l_{32} = \frac{a_{32} - l_{31} d_{11} l_{21}}{d_{22}} = \frac{4 - (0,4)(5)(-0,2)}{7,8} = 0,5641.$$

Coluna 3:

$$d_{33} = a_{33} - (l_{31}^2 d_{11} + l_{32}^2 d_{22}) = 10 - ((0,4)^2(5) + (0,5641)^2(7,8)) = 6,7180.$$

É fácil observar que $A = LDL^T$

$$\begin{bmatrix} 5 & -1 & 2 \\ -1 & 8 & 4 \\ 2 & 4 & 10 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -0,2 & 1 & 0 \\ 0,4 & 0,5641 & 1 \end{bmatrix} \begin{bmatrix} 5 & 0 & 0 \\ 0 & 7,8 & 0 \\ 0 & 0 & 6,7180 \end{bmatrix} \begin{bmatrix} 1 & -0,2 & 0,4 \\ 0 & 1 & 0,5641 \\ 0 & 0 & 1 \end{bmatrix}.$$

Resolvendo o sistema $Ly = b$ pelas substituições sucessivas (2.17)

$$\begin{bmatrix} 1 & 0 & 0 \\ -0,2 & 1 & 0 \\ 0,4 & 0,5641 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 21 \\ 10 \\ 50 \end{bmatrix} \rightsquigarrow y = \begin{bmatrix} 21 \\ 14,2 \\ 33,5898 \end{bmatrix}.$$

Resolvendo o sistema $Dt = y$

$$\begin{bmatrix} 5 & 0 & 0 \\ 0 & 7,8 & 0 \\ 0 & 0 & 6,7180 \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix} = \begin{bmatrix} 21 \\ 14,2 \\ 33,5898 \end{bmatrix} \rightsquigarrow t = \begin{bmatrix} 4,2 \\ 1,8205 \\ 5,0000 \end{bmatrix}.$$

Resolvendo o sistema $L^T x = t$ pelas substituições retroativas (2.18)

$$\begin{bmatrix} 1 & -0,2 & 0,4 \\ 0 & 1 & 0,5641 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4,2 \\ 1,8205 \\ 5,0000 \end{bmatrix} \rightsquigarrow x = \begin{bmatrix} 2,0000 \\ -1,0000 \\ 5,0000 \end{bmatrix}.$$

A exatidão da solução é verificada pelo vetor resíduo $r = b - Ax$

$$r = \begin{bmatrix} 21 \\ 10 \\ 50 \end{bmatrix} - \begin{bmatrix} 5 & -1 & 2 \\ -1 & 8 & 4 \\ 2 & 4 & 10 \end{bmatrix} \begin{bmatrix} 2,0000 \\ -1,0000 \\ 5,0000 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix},$$

ou seja, a solução é exata com quatro decimais. A unicidade da solução é verificada por meio do cálculo do determinante dado por (2.22)

$$\det(A) = \prod_{i=1}^3 d_{ii} \approx (5)(7,8)(6,7180) = 262,0020 \neq 0 \rightarrow \text{solução única.}$$

Um algoritmo para fatorar uma matriz A simétrica definida positiva tal que $A = LDL^T$ é mostrado na Figura 2.8. Os parâmetros de entrada são a ordem n e a parte triangular inferior da matriz A . Os parâmetros de saída são A (contendo em sua diagonal a diagonal de D e na parte triangular inferior o fator L sem os elementos unitários) e o determinante Det .

As soluções dos sistemas $Ly = b$, $Dt = y$ e $L^T x = t$ são obtidas após pequenas modificações dos algoritmos das Figuras 2.3 e 2.4.

```

Algoritmo Decomposição  $LDL^T$ 
{ Objetivo: Fazer a decomposição  $LDL^T$  de uma matriz  $A$  }
{ simétrica e definida positiva }
parâmetros de entrada  $n$ ,  $A$  { ordem e matriz a ser decomposta }
parâmetros de saída  $A$ ,  $\text{Det}$ 
{ matriz decomposta  $A = L - I + D$  e determinante }
 $\text{Det} \leftarrow 1$ 
para  $j \leftarrow 1$  até  $n$  faça
   $Soma \leftarrow 0$ 
  para  $k \leftarrow 1$  até  $j - 1$  faça
     $Soma \leftarrow Soma + A(j, k)^2 * A(k, k)$ 
  fimpara
   $A(j, j) \leftarrow A(j, j) - Soma$ 
   $r = 1/A(j, j); \text{Det} \leftarrow \text{Det} * A(j, j)$ 
  para  $i \leftarrow j + 1$  até  $n$  faça
     $Soma \leftarrow 0$ 
    para  $k \leftarrow 1$  até  $j - 1$  faça
       $Soma \leftarrow Soma + A(i, k) * A(k, k) * A(j, k)$ 
    fimpara
     $A(i, j) \leftarrow (A(i, j) - Soma) * r$ 
  fimpara
fimpara
finalgoritmo

```

Figura 2.8 Decomposição LDL^T da matriz A simétrica definida positiva.

Exemplo 2.41 Decompor a matriz do sistema do Exemplo 2.40 utilizando o algoritmo da Figura 2.8.

```
% Os valores de entrada
n = 3
A =
  5
 -1   8
  2   4   10
% produzem os resultados pela decomposicao LDLt
A =
  5.0000
 -0.2000  7.8000
  0.4000  0.5641  6.7179
Det = 262.0000
```

A complexidade computacional do algoritmo da Figura 2.8 é apresentada na Tabela 2.6, sen-
do a operação de potenciação contada como multiplicação e não incluindo as n multiplicações
efetuadas para o cálculo do determinante.

Tabela 2.6 Complexidade da decomposição LDL^T de matriz de ordem n .
(Desconsiderando as multiplicações para o cálculo do determinante.)

| Operações | Complexidade |
|----------------|--|
| adições | $\frac{1}{6}n^3 + \frac{1}{2}n^2 + \frac{1}{3}n$ |
| multiplicações | $\frac{1}{3}n^3 + \frac{1}{2}n^2 - \frac{5}{6}n$ |
| divisões | n |

A vantagem da fatoração LDL^T é evitar o cálculo de raiz quadrada. Assim, poder-se-ia pensar em usá-la em matriz simétrica que não seja definida positiva. No entanto, neste caso esta decomposição não é estável, sendo recomendado o uso de outros métodos, como o de Aasen [20].

2.6 Decomposição espectral

Considerando que uma matriz A de ordem n possui autovalores λ_i , $i = 1, 2, \dots, n$, e que cada autovalor tem um autovetor correspondente, então a relação (2.2) torna-se

$$Av_i = \lambda_i v_i, \quad i = 1, 2, \dots, n, \text{ ou}$$

$$AV = V\Lambda,$$

onde $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$ é uma matriz diagonal contendo os autovalores λ_i e V é a matriz, cujas colunas são os autovetores v_i . Pós-multiplicando a equação acima por V^{-1} , tem-se a matriz A decomposta em termos de seus autovalores e autovetores, a chamada decomposição espectral de A

$$A = V\Lambda V^{-1}. \quad (2.23)$$

2.6.1 Cálculo dos autovetores

Da relação fundamental (2.2), tem-se que

$$(A - \lambda_i I)v_i = 0. \quad (2.24)$$

No entanto, como a matriz $(A - \lambda_i I)$ é singular ($\det(A - \lambda_i I) = 0$) e o sistema (2.24) é homogêneo, então ele apresenta infinitas soluções v_i e não nenhuma solução. Atribuindo um valor arbitrário a um elemento de v_i , por exemplo $v_{i1} = 1$, podem-se obter os demais elementos do autovetor pela solução do sistema resultante de ordem $n - 1$.

Exemplo 2.42 Fazer a decomposição espectral da matriz

$$A = \begin{bmatrix} 7 & 14 & -2 \\ -3 & -10 & 2 \\ -12 & -28 & 5 \end{bmatrix}.$$

O polinômio característico é

$$\Lambda \quad D_3(\lambda) = \det(A - \lambda I) = \det \left(\begin{bmatrix} 7-\lambda & 14 & -2 \\ -3 & -10-\lambda & 2 \\ -12 & -28 & 5-\lambda \end{bmatrix} \right).$$

Desenvolvendo o determinante, tem-se que

$$D_3(\lambda) = -\lambda^3 + 2\lambda^2 + 11\lambda - 12.$$

Os três zeros do polinômio característico são os três autovalores de A , a saber, $\lambda_1 = 4$, $\lambda_2 = 1$ e $\lambda_3 = -3$. Portanto, a matriz Λ contendo os autovalores é

$$\Lambda = \begin{bmatrix} 4 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -3 \end{bmatrix}.$$

O cálculo das raízes de equações algébricas será abordado no Capítulo 6. Para calcular o autovetor v correspondente ao autovalor $\lambda_1 = 4$, utiliza-se (2.24)

$$\begin{bmatrix} 3 & 14 & -2 \\ -3 & -14 & 2 \\ -12 & -28 & 1 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Como as equações 1 e 2 são redundantes, elimina-se a segunda e faz-se $v_1 = 1$. Deste modo,

$$\begin{bmatrix} 14 & -2 \\ -28 & 1 \end{bmatrix} \begin{bmatrix} v_2 \\ v_3 \end{bmatrix} = \begin{bmatrix} -3 \\ 12 \end{bmatrix} \rightarrow v_2 = -0,5 \text{ e } v_3 = -2 \rightarrow v = \begin{bmatrix} 1 \\ -0,5 \\ -2 \end{bmatrix}.$$

Para o cálculo do autovetor w correspondente à $\lambda_2 = 1$ resolve-se o sistema

$$\begin{bmatrix} 6 & 14 & -2 \\ -3 & -11 & 2 \\ -12 & -28 & 4 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Sendo as equações 1 e 3 redundantes, então elimina-se a terceira e faz-se $w_1 = 1$, assim

$$\begin{bmatrix} 14 & -2 \\ -11 & 2 \end{bmatrix} \begin{bmatrix} w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} -6 \\ 3 \end{bmatrix} \rightarrow w_2 = -1 \text{ e } w_3 = -4 \rightarrow w = \begin{bmatrix} 1 \\ -1 \\ -4 \end{bmatrix}.$$

Para o cálculo do autovetor z correspondente à $\lambda_3 = -3$, determina-se a solução de

$$\begin{bmatrix} 10 & 14 & -2 \\ -3 & -7 & 2 \\ -12 & -28 & 8 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

Sendo as equações 2 e 3 redundantes, então elimina-se a terceira e faz-se $z_1 = 1$

$$\begin{bmatrix} 14 & -2 \\ -7 & 2 \end{bmatrix} \begin{bmatrix} z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} -10 \\ 3 \end{bmatrix} \rightarrow z_2 = -1 \text{ e } z_3 = -2 \rightarrow z = \begin{bmatrix} 1 \\ -1 \\ -2 \end{bmatrix}.$$

Conseqüentemente, a matriz V contendo os autovetores de A é

$$V = [v \ w \ z] = \begin{bmatrix} 1 & 1 & 1 \\ -0,5 & -1 & -1 \\ -2 & -4 & -2 \end{bmatrix}.$$

O processo de cálculo de matriz inversa será abordado somente na Seção 2.7.2; no entanto, pode-se mostrar que

$$V^{-1} = \begin{bmatrix} 2 & 2 & 0 \\ -1 & 0 & -0,5 \\ 0 & -2 & 0,5 \end{bmatrix}.$$

Deste modo, a decomposição espectral $A = V\Lambda V^{-1}$ pode ser confirmada por

$$\begin{bmatrix} 7 & 14 & -2 \\ -3 & -10 & 2 \\ -12 & -28 & 5 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ -0,5 & -1 & -1 \\ -2 & -4 & -2 \end{bmatrix} \begin{bmatrix} 4 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -3 \end{bmatrix} \begin{bmatrix} 2 & 2 & 0 \\ -1 & 0 & -0,5 \\ 0 & -2 & 0,5 \end{bmatrix}.$$

2.6.2 Solução de sistema linear

A solução do sistema $Ax = b$ pode ser obtida por $x = A^{-1}b$. Então, por (2.23)

$$x = (V\Lambda V^{-1})^{-1}b \sim \boxed{x = (V\Lambda^{-1}V^{-1})b}. \quad (2.25)$$

Esta expressão mostra que o vetor solução x depende dos recíprocos dos autovalores λ_i . Assim, no caso de uma quase singularidade da matriz A , quando pelo menos um autovalor possui um valor próximo de zero faz com que a solução x tenha elementos muito grandes.

Exemplo 2.43 Calcular a solução do sistema abaixo, o qual envolve a matriz dos coeficientes do Exemplo 2.42

$$A = \begin{bmatrix} 7 & 14 & -2 \\ -3 & -10 & 2 \\ -12 & -28 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -10 \\ 14 \\ 29 \end{bmatrix}.$$

Por (2.25) e utilizando os resultados do Exemplo 2.42

$$x = (V\Lambda^{-1}V^{-1})b$$

$$x = \begin{bmatrix} 1 & 1 & 1 \\ -0,5 & -1 & -1 \\ -2 & -4 & -2 \end{bmatrix} \begin{bmatrix} \frac{1}{4} & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -\frac{1}{3} \end{bmatrix} \begin{bmatrix} 2 & 2 & 0 \\ -1 & 0 & -0,5 \\ 0 & -2 & 0,5 \end{bmatrix} \begin{bmatrix} -10 \\ 14 \\ 29 \end{bmatrix} \sim$$

$$x = \begin{bmatrix} 2 \\ -1 \\ 5 \end{bmatrix}.$$

A solução é exata porque o vetor resíduo $r = b - Ax$ é nulo

$$r = \begin{bmatrix} -10 \\ 14 \\ 29 \end{bmatrix} - \begin{bmatrix} 7 & 14 & -2 \\ -3 & -10 & 2 \\ -12 & -28 & 5 \end{bmatrix} \begin{bmatrix} 2 \\ -1 \\ 5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

É claro pelos exemplos acima que a decomposição espectral é de grande custo computacional quando comparada à decomposição LU ou Cholesky e, normalmente, não é utilizada para a solução de sistemas de equações lineares. No entanto, existem situações em que seu uso se faz necessário [7].

2.7 Uso da decomposição

As decomposições LU , LL^T e LDL^T podem ser usadas para resolver sistemas de equações lineares, bem como calcular o determinante de uma matriz, conforme visto nas seções anteriores. Outras importantes aplicações dessas decomposições estão no refinamento da solução de sistemas e no cálculo da matriz inversa.

2.7.1 Refinamento da solução

Seja x^0 uma solução aproximada do sistema $Ax = b$ calculada por (2.11), usando decomposição LU com pivotação parcial

$$LUx^0 = Pb \rightarrow Lt = Pb \text{ e } Ux^0 = t.$$

O vetor temporário t é obtido pelas substituições sucessivas (2.7), e a solução aproximada x^0 é calculada pelas substituições retroativas (2.8). Apesar de a decomposição LU ser um método direto e, portanto, teoricamente exato, a solução será aproximada quando os fatores

L e U perderem exatidão devido aos erros de arredondamento. Uma solução melhorada x^1 é obtida por

$$x^1 = x^0 + c^0,$$

onde c^0 é um vetor de correção. Assim,

$$Ax^1 = b \rightarrow A(x^0 + c^0) = b \rightarrow Ac^0 = b - Ax^0 \sim Ac^0 = r^0.$$

Deste modo, a parcela de correção c^0 é a solução do sistema $Ac^0 = r^0$ dada por

$$LUc^0 = Pr^0 \rightarrow Lt = Pr^0 \text{ e } Uc^0 = t.$$

Uma melhor aproximação x^2 é conseguida por

$$x^2 = x^1 + c^1,$$

onde c^1 é a solução de $Ac^1 = r^1$ obtida por $LUc^1 = Pr^1$, e assim sucessivamente. Esquematicamente,

$$\boxed{\begin{aligned} LUx^0 &= Pb \rightarrow Lt = Pb \text{ e } Ux^0 = t, \\ r^k &= b - Ax^k \\ LUc^k &= Pr^k \rightarrow Lt = Pr^k \text{ e } Uc^k = t \\ x^{k+1} &= x^k + c^k \end{aligned}} \quad k = 0, 1, 2, \dots$$

O processo se repete até que um critério de parada seja satisfeito. As decomposições LL^T e LDL^T também podem ser usadas para o refinamento da solução de um sistema linear, desde que a matriz dos coeficientes seja simétrica definida positiva. No caso da decomposição de Cholesky, basta usar L^T em vez de U e $P = I$ nas equações acima.

Exemplo 2.44 Resolver o sistema abaixo e refinar a solução até que $\|c\|_\infty < 10^{-3}$

$$\begin{bmatrix} 2 & 3 & -1 \\ -3 & 5 & 6 \\ 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -4 \\ 19 \\ 11 \end{bmatrix}.$$

A decomposição LU com pivotação parcial fornece as três matrizes

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -0,67 & 1 & 0 \\ -0,33 & 0,42 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} -3 & 5 & 6 \\ 0 & 6,33 & 3 \\ 0 & 0 & 2,74 \end{bmatrix} \text{ e } P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Inicialmente, calcula-se x^0

$$Ax^0 = b \rightarrow LUx^0 = Pb,$$

$$Lt = Pb \rightsquigarrow t = \begin{bmatrix} 19 \\ 8,73 \\ 13,6034 \end{bmatrix} \text{ e } Ux^0 = t \rightsquigarrow x^0 = \begin{bmatrix} 1,9731 \\ -0,9738 \\ 4,9647 \end{bmatrix}.$$

Em seguida, a solução é refinada até que a condição de parada seja satisfeita

$$r^0 = b - Ax^0 = \begin{bmatrix} -0,0601 \\ 0 \\ 0,0712 \end{bmatrix}, \quad LUc^0 = Pr^0 \rightsquigarrow c^0 = \begin{bmatrix} 0,0268 \\ -0,0262 \\ 0,0352 \end{bmatrix},$$

$$x^1 = x^0 + c^0 = \begin{bmatrix} 1,9999 \\ -1,0000 \\ 4,9999 \end{bmatrix},$$

$$r^1 = b - Ax^1 = \begin{bmatrix} 0,0001 \\ 0 \\ 0,0002 \end{bmatrix}, \quad LUc^1 = Pr^1 \rightsquigarrow c^1 = \begin{bmatrix} 0,0001 \\ 0,0000 \\ 0,0001 \end{bmatrix},$$

$$x^2 = x^1 + c^1 = \begin{bmatrix} 2,0000 \\ -1,0000 \\ 5,0000 \end{bmatrix}.$$

O refinamento é interrompido porque $\|c^1\|_\infty = 0,0001 < 10^{-3}$.

2.7.2 Cálculo da matriz inversa

A matriz inversa satisfaz à propriedade

$$AA^{-1} = I$$

ou

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nn} \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix},$$

onde $V = A^{-1}$ é usado para simplificar a notação. Para calcular V , basta resolver os n sistemas lineares da forma

$$Av_i = e_i, \quad i = 1, 2, \dots, n,$$

onde v_i e e_i são as i -ésimas colunas das matrizes inversa e identidade, respectivamente. Como a matriz dos coeficientes é a mesma para os n sistemas, deve ser feita a decomposição de A usando LL^T ou LDL^T se A for simétrica definida positiva ou LU se A for não simétrica. Depois, os n vetores v_i que compõem a inversa são calculados usando as substituições sucessivas e retroativas.

Exemplo 2.45 Calcular a inversa da matriz

$$A = \begin{bmatrix} 4 & -6 & 2 \\ -6 & 10 & -5 \\ 2 & -5 & 30 \end{bmatrix}.$$

Como A é simétrica pode ser usada a decomposição de Cholesky, cujo fator é

$$L = \begin{bmatrix} 2 & 0 & 0 \\ -3 & 1 & 0 \\ 1 & -2 & 5 \end{bmatrix}.$$

As três colunas são calculadas como se segue:

Coluna 1: $Av_1 = e_1$

$$LL^T v_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \rightarrow Lt = e_1 \rightsquigarrow t = \begin{bmatrix} 0,5 \\ 1,5 \\ 0,5 \end{bmatrix} \text{ e } L^T v_1 = t \rightsquigarrow v_1 = \begin{bmatrix} 2,75 \\ 1,70 \\ 0,10 \end{bmatrix}.$$

Coluna 2: $Av_2 = e_2$

$$LL^T v_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \rightarrow Lt = e_2 \rightsquigarrow t = \begin{bmatrix} 0 \\ -1 \\ 0,4 \end{bmatrix} \text{ e } L^T v_2 = t \rightsquigarrow v_2 = \begin{bmatrix} 1,70 \\ 1,16 \\ 0,08 \end{bmatrix}.$$

Coluna 3: $Av_3 = e_3$

$$LL^T v_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \rightarrow Lt = e_3 \rightsquigarrow t = \begin{bmatrix} 0 \\ 0 \\ 0,2 \end{bmatrix} \text{ e } L^T v_3 = t \rightsquigarrow v_3 = \begin{bmatrix} 0,10 \\ 0,08 \\ 0,04 \end{bmatrix}.$$

Conseqüentemente, $A^{-1} = V = [v_1 \ v_2 \ v_3]$, ou seja,

$$A^{-1} = \begin{bmatrix} 2,75 & 1,70 & 0,10 \\ 1,70 & 1,16 & 0,08 \\ 0,10 & 0,08 & 0,04 \end{bmatrix}.$$

A relação $AA^{-1} = I$ pode ser verificada

$$\begin{bmatrix} 4 & -6 & 2 \\ -6 & 10 & -5 \\ 2 & -5 & 30 \end{bmatrix} \begin{bmatrix} 2,75 & 1,70 & 0,10 \\ 1,70 & 1,16 & 0,08 \\ 0,10 & 0,08 & 0,04 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

2.8 Métodos iterativos estacionários

Um sistema de equações lineares $Ax = b$ pode ser resolvido por um processo que consiste em gerar, a partir de um vetor inicial x^0 , uma seqüência de vetores $\{x^1, x^2, x^3, \dots, x^k, \dots\}$ que deve convergir para a solução x do sistema. Tal processo é chamado iterativo, pois uma mesma série de operações é repetida várias vezes. Existem várias classes de métodos iterativos, todavia somente os estacionários serão estudados neste texto.

Seja uma matriz M chamada matriz de iteração e c um vetor constante. Um método iterativo escrito na forma

$$x^{k+1} = Mx^k + c \quad (2.26)$$

é dito estacionário quando a matriz M for fixa, ou seja, quando ela não for alterada durante o processo. Serão abordados, nesta seção, três métodos iterativos estacionários: Jacobi, Gauss-Seidel e sobre-relaxação sucessiva.

2.8.1 Condição de convergência

O fato de a seqüência de vetores $\{x^0, x^1, x^2, \dots, x^k, \dots\}$ convergir para a solução do sistema $Ax = b$ é garantido pela condição do Teorema 2.3 [3, Teorema 5.3].

Teorema 2.3 O método iterativo (2.26) converge com qualquer valor inicial x^0 se, e somente se, $\rho(M) < 1$, sendo $\rho(M)$ o raio espectral (maior autovalor em módulo) da matriz de iteração M .

Porém, a determinação do raio espectral da matriz de iteração $\rho(M)$ pode requerer maior esforço computacional que a própria solução do sistema $Ax = b$. Por isso, para alguns métodos iterativos estacionários usualmente se utiliza o chamado critério das linhas para prever a convergência [3, Teorema 5.8].

Teorema 2.4 É condição suficiente para a convergência dos métodos iterativos de Jacobi e Gauss-Seidel que a matriz dos coeficientes A seja diagonal estritamente dominante, ou seja,

$$|a_{ii}| > \sum_{j=1, j \neq i}^n |a_{ij}|, \quad i = 1, 2, \dots, n. \quad (2.27)$$

Em palavras, é condição suficiente para convergência se a matriz dos coeficientes for diagonalmente dominante, isto é, que o elemento da diagonal, em módulo, seja maior que a soma, em módulo, dos demais elementos da mesma linha, para todas as linhas. Pelos Teoremas 2.3 e 2.4, pode-se notar que a convergência não depende da escolha do vetor inicial x^0 .

2.8.2 Critério de parada

A cada passo do método iterativo (2.26) a solução é obtida com uma exatidão crescente, isto é,

$$\lim_{k \rightarrow \infty} x^k = x.$$

O processo deve ser interrompido quando algum critério de parada for satisfeito, como, por exemplo,

$$\frac{\|x^k - x^{k-1}\|}{\|x^k\|} \leq \varepsilon \quad \text{ou} \quad (2.28)$$

$$k \geq k_{\max}, \quad (2.29)$$

onde ε é a tolerância e k_{\max} é o número máximo de iterações. Nos algoritmos, será adotada a norma- ∞ em (2.28), ou seja,

$$\frac{\max_{1 \leq i \leq n} |x_i^k - x_i^{k-1}|}{\max_{1 \leq i \leq n} |x_i^k|} \leq \varepsilon,$$

sendo x_i^k o i -ésimo componente do vetor x^k obtido na k -ésima iteração. Na prática, a tolerância ε define com qual exatidão a solução é calculada. Entretanto, quando se utiliza aritmética de ponto flutuante a exatidão não pode ser tão grande quanto se queira, pois ela é limitada de acordo com o número de *bytes* das variáveis do programa.

2.8.3 Método de Jacobi

Considere o sistema linear $Ax = b$, com a matriz A decomposta de modo que

$$A = D + E + F,$$

onde D é uma matriz diagonal e E e F são matrizes triangulares inferior e superior, respectivamente, com diagonais nulas. O sistema pode, então, ser escrito na forma

$$(D + E + F)x = b \rightarrow Dx = -(E + F)x + b.$$

Esta igualdade pode ser convertida em um processo iterativo formando a recorrência

$$x^{k+1} = (-D^{-1}(E + F))x^k + D^{-1}b \rightarrow x^{k+1} = Jx^k + c, \quad (2.30)$$

tal que, por (2.26), a matriz $J = -D^{-1}(E + F)$ é a matriz de iteração do método de Jacobi [26]. Uma forma análoga de deduzir o método de Jacobi consiste em escrever o sistema de equações lineares na forma

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 + \cdots + a_{2n}x_n &= b_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 + \cdots + a_{3n}x_n &= b_3 \\ &\vdots && \vdots \\ a_{n1}x_1 + a_{n2}x_2 + a_{n3}x_3 + \cdots + a_{nn}x_n &= b_n \end{aligned}$$

e explicitar x_i na i -ésima equação. Escrevendo na forma de iteração, têm-se as chamadas equações de iterações do método de Jacobi

$$\left. \begin{aligned} x_1^{k+1} &= \frac{1}{a_{11}}(-a_{12}x_2^k - a_{13}x_3^k - \cdots - a_{1n}x_n^k + b_1), \\ x_2^{k+1} &= \frac{1}{a_{22}}(-a_{21}x_1^k - a_{23}x_3^k - \cdots - a_{2n}x_n^k + b_2), \\ x_3^{k+1} &= \frac{1}{a_{33}}(-a_{31}x_1^k - a_{32}x_2^k - \cdots - a_{3n}x_n^k + b_3), \\ &\vdots \\ x_n^{k+1} &= \frac{1}{a_{nn}}(-a_{n1}x_1^k - a_{n2}x_2^k - \cdots - a_{n,n-1}x_{n-1}^k + b_n), \end{aligned} \right\} \quad (2.31)$$

ou na forma matricial

$$\begin{bmatrix} x_1^{k+1} \\ x_2^{k+1} \\ x_3^{k+1} \\ \vdots \\ x_n^{k+1} \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & -\frac{a_{12}}{a_{11}} & -\frac{a_{13}}{a_{11}} & \cdots & -\frac{a_{1n}}{a_{11}} \\ -\frac{a_{21}}{a_{22}} & 0 & -\frac{a_{23}}{a_{22}} & \cdots & -\frac{a_{2n}}{a_{22}} \\ -\frac{a_{31}}{a_{33}} & -\frac{a_{32}}{a_{33}} & 0 & \cdots & -\frac{a_{3n}}{a_{33}} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\frac{a_{n1}}{a_{nn}} & -\frac{a_{n2}}{a_{nn}} & \cdots & -\frac{a_{n,n-1}}{a_{nn}} & 0 \end{bmatrix}}_J \begin{bmatrix} x_1^k \\ x_2^k \\ x_3^k \\ \vdots \\ x_n^k \end{bmatrix} + \underbrace{\begin{bmatrix} \frac{b_1}{a_{11}} \\ \frac{b_2}{a_{22}} \\ \frac{b_3}{a_{33}} \\ \vdots \\ \frac{b_n}{a_{nn}} \end{bmatrix}}_c$$

Uma das vantagens dos métodos iterativos é que a convergência independe do valor inicial x^0 . Assim, pode ser usado como x^0 ou uma estimativa conhecida, ou um valor qualquer, caso esta não esteja disponível. Usualmente, faz-se $x^0 = 0$; no entanto, usando este valor em (2.30), tem-se que

$$x^1 = c \rightarrow x_i^1 = \frac{b_i}{a_{ii}}.$$

Deste modo, este valor pode ser tomado como o valor inicial

$$x_i^0 = \frac{b_i}{a_{ii}}. \quad (2.32)$$

Algoritmo e complexidade

O algoritmo mostrado na Figura 2.9 calcula a solução de um sistema linear $Ax = b$ pelo método iterativo de Jacobi, com critério de parada dado por (2.28) e (2.29). Os parâmetros de entrada são a ordem n , a matriz A , o vetor b , a tolerância (critério de parada) *Toler* e o número máximo de iterações *IterMax*. Os parâmetros de saída são o vetor solução x , o número de iterações gastos *Iter* e a condição de erro *CondErro* para verificar se houve convergência (*CondErro* = 0 significa que houve convergência e *CondErro* = 1, que não houve). Os valores intermediários do vetor solução também são listados durante a execução do algoritmo. A complexidade computacional do algoritmo da Figura 2.9 é mostrada na Tabela 2.7.

Tabela 2.7 Complexidade de Jacobi usando k iterações em sistema de ordem $n > 1$.

| Operações | Complexidade |
|----------------|-----------------|
| adições | $kn^2 + kn + k$ |
| multiplicações | $(k+1)n^2 - kn$ |
| divisões | $n + k$ |

```

Algoritmo Jacobi
{ Objetivo: Resolver o sistema  $Ax = b$  pelo método iterativo de Jacobi }
parâmetros de entrada  $n$ ,  $A$ ,  $b$ ,  $Toler$ ,  $IterMax$ 
    { ordem, matriz, vetor independente, }
    { tolerância e número máximo de iterações }
parâmetros de saída  $x$ ,  $Iter$ ,  $CondErro$ 
    { vetor solução, número de iterações e condição de erro }
    { construção das matrizes para as iterações }
para  $i \leftarrow 1$  até  $n$  faça
     $r \leftarrow 1/A(i, i)$ 
    para  $j \leftarrow 1$  até  $n$  faça
        se  $i \neq j$  então  $A(i, j) \leftarrow A(i, j) * r$ , fimse
    fimpara
     $b(j) \leftarrow b(j) * r$ ;  $x(j) \leftarrow b(j)$ 
fimpara;  $Iter \leftarrow 0$ 
{ iterações de Jacobi }
repita
     $Iter \leftarrow Iter + 1$ 
    para  $i \leftarrow 1$  até  $n$  faça
         $Soma \leftarrow 0$ 
        para  $j \leftarrow 1$  até  $n$  faça
            se  $i \neq j$  então  $Soma \leftarrow Soma + A(i, j) * x(j)$ , fimse
        fimpara
         $v(j) \leftarrow b(j) - Soma$ 
    fimpara
     $NormaNum \leftarrow 0$ ;  $NormaDen \leftarrow 0$ 
    para  $i \leftarrow 1$  até  $n$  faça
         $t \leftarrow \text{abs}(v(i) - x(i))$ 
        se  $t > NormaNum$  então  $NormaNum \leftarrow t$ , fimse
        se  $\text{abs}(v(i)) > NormaDen$  então  $NormaDen \leftarrow \text{abs}(v(i))$ , fimse
         $x(i) \leftarrow v(i)$ 
    fimpara
     $NormaRel \leftarrow NormaNum / NormaDen$ 
    escreva  $Iter$ ,  $x$ ,  $NormaRel$ 
    { teste de convergência }
    se  $NormaRel \leq Toler$  ou  $Iter \geq IterMax$  então interrompa, fimse
    fimrepita
    se  $NormaRel \leq Toler$  então  $CondErro \leftarrow 0$ , senão  $CondErro \leftarrow 1$ 
    fimse
finalgoritmo

```

Figura 2.9 Método de Jacobi para solução do sistema linear $Ax = b$.

(Ver significado da função abs na Tabela 1.1, na página 6.)

Exemplo 2.46 Resolver o sistema de equações pelo método de Jacobi com $\epsilon < 10^{-5}$ e $k_{\max} = 50$ usando os critérios (2.28) e (2.29)

$$\begin{bmatrix} 10 & 3 & -2 \\ 2 & 8 & -1 \\ 1 & 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 57 \\ 20 \\ -4 \end{bmatrix}.$$

Pelas condições (2.27) o processo convergirá, pois a matriz dos coeficientes é diagonal estritamente dominante, isto é,

$$|10| > |3| + |-2|, \quad |8| > |2| + |-1| \quad \text{e} \quad |5| > |1| + |1|.$$

Por (2.31), as equações de iterações são

$$x_1^{k+1} = \frac{1}{10} (-3x_2^k + 2x_3^k + 57),$$

$$x_2^{k+1} = \frac{1}{8} (-2x_1^k + x_3^k + 20) \quad \text{e}$$

$$x_3^{k+1} = \frac{1}{5} (-x_1^k - x_2^k - 4).$$

Partindo-se de x^0 dado por (2.32) tem-se que $x^0 = [5,7 \ 2,5 \ -0,8]^T$ e, pelas equações de iterações, as coordenadas do vetor da primeira iteração são

$$x_1^1 = \frac{1}{10} (-3x_2^0 + 2x_3^0 + 57) = \frac{1}{10} (-3(2,5) + 2(-0,8) + 57) \sim x_1^1 = 4,79,$$

$$x_2^1 = \frac{1}{8} (-2x_1^0 + x_3^0 + 20) = \frac{1}{8} (-2(5,7) + (-0,8) + 20) \sim x_2^1 = 0,975 \quad \text{e}$$

$$x_3^1 = \frac{1}{5} (-x_1^0 - x_2^0 - 4) = \frac{1}{5} ((-5,7) - (2,5) - 4) \sim x_3^1 = -2,44.$$

Portanto, $x^1 = [4,79 \ 0,975 \ -2,44]^T$ e

$$\frac{\|x^1 - x^0\|_\infty}{\|x^1\|_\infty} = \frac{\max(|4,79 - 5,7|, |0,975 - 2,5|, |-2,44 - (-0,8)|)}{\max(|4,79|, |0,975|, |-2,44|)},$$

$$\frac{\|x^1 - x^0\|_\infty}{\|x^1\|_\infty} = \frac{\max(0,91; 1,525; 1,64)}{\max(4,79; 0,975; 2,44)} = 0,3424.$$

Os resultados podem ser gerados pelo algoritmo da Figura 2.9

% Os valores de entrada

$n = 3$

$A =$

$$\begin{matrix} 10 & 3 & -2 \\ 2 & 8 & -1 \\ 1 & 1 & 5 \end{matrix}$$

$b =$

$$57$$

$$20$$

$$-4$$

```

Toler = 1.0000e-05
IterMax = 50
% produzem os resultados
Solucao de sistema linear pelo metodo de Jacobi
Iter    x1      x2      x3      NormaRelativa
 0   5.70000  2.50000 -0.80000
 1   4.79000  0.97500 -2.44000  3.42380e-01
 2   4.91950  0.99750 -1.95300  9.89938e-02
 3   5.01015  1.02600 -1.98340  1.80933e-02
 4   4.99952  0.99954 -2.00723  5.29725e-03
 5   4.99869  1.00022 -1.99901  1.64413e-03
 6   5.00013  1.00045 -1.99978  2.88007e-04
 7   4.99991  0.99999 -2.00012  9.12629e-05
 8   4.99998  1.00001 -1.99998  2.72243e-05
 9   5.00000  1.00001 -2.00000  4.59167e-06

```

Solucao = 5.00000 1.00001 -2.00000

Iter = 9
CondErro = 0

Na implementação do algoritmo foram acrescentadas algumas informações para facilitar o entendimento dos resultados. Pelos valores acima,

$$x \approx x^9 = [5,00000 \ 1,00001 \ -2,00000]^T.$$

O valor CondErro = 0 indica que a solução convergiu dentro das condições especificadas pelos parâmetros Toler e IterMax.

Exemplo 2.47 Calcular três aproximações do vetor solução do sistema do Exemplo 2.46 utilizando a formulação (2.30): $x^{k+1} = -D^{-1}(E+F)x^k + D^{-1}b = Jx^k + c$.

Decompondo $A = D + E + F$, tem-se

$$D = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 5 \end{bmatrix}, E = \begin{bmatrix} 0 & 0 & 0 \\ 2 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \text{ e } F = \begin{bmatrix} 0 & 3 & -2 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}.$$

A matriz J e os vetores c e x^0 são

$$J = -D^{-1}(E+F) = \begin{bmatrix} 0 & -0,3 & 0,2 \\ -0,25 & 0 & 0,125 \\ -0,2 & -0,2 & 0 \end{bmatrix} \text{ e } c = D^{-1}b = x^0 = [5,7 \ 2,5 \ -0,8]^T.$$

As primeiras aproximações da solução por (2.30) são mostradas na tabela a seguir, cujos resultados são idênticos àqueles gerados pelo algoritmo da Figura 2.9.

| k | x_1^k | x_2^k | x_3^k |
|-----|---------|---------|----------|
| 0 | 5,70000 | 2,50000 | -0,80000 |
| 1 | 4,79000 | 0,97500 | -2,44000 |
| 2 | 4,91950 | 0,99750 | -1,95300 |
| 3 | 5,01015 | 1,02600 | -1,98340 |

Exemplo 2.48 Resolver o sistema pelo método de Jacobi com $\varepsilon < 10^{-3}$ e $k_{\max} = 50$ usando os critérios (2.28) e (2.29)

$$\begin{bmatrix} 5 & 2 & 0 & -1 \\ 1 & 8 & -3 & 2 \\ 0 & 1 & 6 & 1 \\ 1 & -1 & 2 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 6 \\ 10 \\ -5 \\ 0 \end{bmatrix}.$$

O processo irá convergir, pois a matriz dos coeficientes é diagonalmente dominante, ou seja, de acordo com as condições (2.27), os elementos das linhas satisfazem

$$|5| > |2| + |0| + |-1|, |8| > |1| + |-3| + |2|,$$

$$|6| > |0| + |1| + |1| \text{ e } |9| > |1| + |-1| + |2|.$$

Por (2.31), as equações de iterações são

$$x_1^{k+1} = \frac{1}{5}(-2x_2^k + x_4^k + 6),$$

$$x_2^{k+1} = \frac{1}{8}(-x_1^k + 3x_3^k - 2x_4^k + 10),$$

$$x_3^{k+1} = \frac{1}{6}(-x_2^k - x_4^k - 5) \text{ e}$$

$$x_4^{k+1} = \frac{1}{9}(-x_1^k + x_2^k - 2x_3^k).$$

Usando x^0 dado por (2.32), $x^0 = [1,2 \ 1,25 \ -0,8333 \ 0]^T$. Pelas equações de iterações

$$x_1^1 = \frac{1}{5}(-2x_2^0 + x_4^0 + 6) = \frac{1}{5}(-2(1,25) + (0) + 6) = 0,7;$$

$$x_2^1 = \frac{1}{8}(-x_1^0 + 3x_3^0 - 2x_4^0 + 10) = \frac{1}{8}(-(1,2) + 3(-0,8333) - 2(0) + 10) = 0,7875;$$

$$x_3^1 = \frac{1}{6}(-x_2^0 - x_4^0 - 5) = \frac{1}{6}(-(1,25) - (0) - 5) = -1,0417;$$

$$x_4^1 = \frac{1}{9}(-x_1^0 + x_2^0 - 2x_3^0) = \frac{1}{9}(-(1,2) + (1,25) - 2(-0,8333)) = 0,1907.$$

Conseqüentemente, o vetor da primeira iteração é

$$x^1 = [0,7 \ 0,7875 \ -1,0417 \ 0,1907]^T \text{ e}$$

$$\frac{\|x^1 - x^0\|_\infty}{\|x^1\|_\infty} = \frac{\max(|0,7 - 1,2|, |0,7875 - 1,25|, |-1,0417 - (-0,8333)|, |0,1907 - 0|)}{\max(|0,7|, |0,7875|, |-1,0417|, |0,1907|)},$$

$$\frac{\|x^1 - x^0\|_\infty}{\|x^1\|_\infty} = \frac{\max(0,5; 0,4625; 0,2084; 0,1907)}{\max(0,7; 0,7875; 1,0417; 0,1907)} = 0,4800.$$

Pelo algoritmo da Figura 2.9, tem-se

```
% Os valores de entrada
n = 4
A =
    5   2   0   -1
    1   8  -3   2
    0   1   6   1
    1  -1   2   9
b =
    6
   10
   -5
    0
Toler = 1.0000e-03
IterMax = 50
% produzem os resultados
Solucao de sistema linear pelo metodo de Jacobi
Iter      x1          x2          x3          x4      NormaRelativa
  0  1.20000  1.25000 -0.83333  0.00000
  1  0.70000  0.78750 -1.04167  0.19074  4.80000e-01
  2  0.92315  0.72419 -0.99637  0.24120  2.23960e-01
  3  0.95856  0.70067 -0.99423  0.19931  4.21369e-02
  4  0.95960  0.70751 -0.98333  0.19229  1.10879e-02
  5  0.95545  0.71323 -0.98330  0.19051  5.81305e-03
  6  0.95281  0.71420 -0.98396  0.19160  2.68474e-03
  7  0.95264  0.71402 -0.98430  0.19215  5.56291e-04
```

$$\text{Solucao} = 0.95264 \quad 0.71402 \quad -0.98430 \quad 0.19215$$

Iter = 7
CondErro = 0

Deste modo, $x \approx x^7 = [0.95264 \quad 0.71402 \quad -0.98430 \quad 0.19215]^T$.

2.8.4 Método de Gauss-Seidel

Seja o sistema $Ax = b$, com a matriz A sendo decomposta tal que $A = D + E + F$, sendo D uma matriz diagonal e E e F matrizes triangulares inferiores e superiores, respectivamente, com diagonais nulas. O sistema linear pode, então, ser escrito como

$$(D + E + F)x = b \rightarrow (D + E)x = -Fx + b$$

ou na forma de iteração obtida pela recorrência

$$x^{k+1} = -(D + E)^{-1}F x^k + (D + E)^{-1}b \rightarrow x^{k+1} = Sx^k + d. \quad (2.33)$$

Por (2.26), a matriz $S = -(D + E)^{-1}F$ é a matriz de iteração do método de Gauss-Seidel². Um modo mais prático de implementar este método, sem usar a matriz de iteração que utiliza inversa, é mostrado a seguir. Seja

$$(D + E + F)x = b \rightarrow (D + E)x = -Fx + b.$$

²Há uma curiosidade a respeito do nome deste método. Segundo G. E. Forsythe [24, pág. 115], o método de Gauss-Seidel não era conhecido por Gauss e nem recomendado por Seidel!

Na forma de recorrência,

$$(D + E)x^{k+1} = -Fx^k + b \rightarrow Dx^{k+1} = -Ex^{k+1} - Fx^k + b.$$

Escrevendo a segunda equação na forma matricial

$$Dx^{k+1} = -\underbrace{\begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ a_{21} & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n-1,1} & a_{n-1,2} & \cdots & 0 & 0 \\ a_{n1} & a_{n2} & \cdots & a_{n,n-1} & 0 \end{bmatrix}}_E \underbrace{\begin{bmatrix} x_1^{k+1} \\ x_2^{k+1} \\ x_3^{k+1} \\ \vdots \\ x_n^{k+1} \end{bmatrix}}_{x^{k+1}} -$$

$$\underbrace{\begin{bmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & 0 & a_{23} & \cdots & a_{2n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & a_{n-1,n} \\ 0 & 0 & \cdots & 0 & 0 \end{bmatrix}}_F \underbrace{\begin{bmatrix} x_1^k \\ x_2^k \\ x_3^k \\ \vdots \\ x_n^k \end{bmatrix}}_{x^k} + \underbrace{\begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ \vdots \\ b_n \end{bmatrix}}_b,$$

obtém-se, então, as equações de iterações do método de Gauss-Seidel

$$\left. \begin{aligned} x_1^{k+1} &= \frac{1}{a_{11}} (-a_{12}x_2^k - a_{13}x_3^k - \cdots - a_{1n}x_n^k + b_1), \\ x_2^{k+1} &= \frac{1}{a_{22}} (-a_{21}x_1^{k+1} - a_{23}x_3^k - \cdots - a_{2n}x_n^k + b_2), \\ x_3^{k+1} &= \frac{1}{a_{33}} (-a_{31}x_1^{k+1} - a_{32}x_2^{k+1} - \cdots - a_{3n}x_n^k + b_3), \\ &\vdots \\ x_n^{k+1} &= \frac{1}{a_{nn}} (-a_{n1}x_1^{k+1} - a_{n2}x_2^{k+1} - \cdots - a_{n,n-1}x_{n-1}^{k+1} + b_n). \end{aligned} \right\} \quad (2.34)$$

As equações de iterações do método de Jacobi (2.31) mostram que x^{k+1} é calculado usando somente valores x_i^k da iteração anterior. Por outro lado, (2.34) deixa claro que no método de Gauss-Seidel o vetor x^{k+1} é obtido a partir dos elementos mais recentes, incluindo o próprio x^{k+1} e x^k . O vetor inicial x^0 para o método de Gauss-Seidel pode ser o mesmo usado pelo método de Jacobi, dado por (2.32)

$$x_i^0 = \frac{b_i}{a_{ii}}.$$

Algoritmo e complexidade

A Figura 2.10 mostra um algoritmo para achar a solução de um sistema linear $Ax = b$ pelo método iterativo de Gauss-Seidel, com critério de parada dado por (2.28) e (2.29). Como

no algoritmo de Jacobi, os parâmetros de entrada são a ordem n , a matriz A , o vetor b , a tolerância $Toler$ e o número máximo de iterações $IterMax$; os parâmetros de saída são a solução x , o número de iterações gastos $Iter$ e a condição de erro $CondErro$ para verificar se a solução convergiu ($CondErro = 0$ significa que houve convergência e $CondErro = 1$, que não houve). A complexidade computacional do algoritmo da Figura 2.10 é mostrada na Tabela 2.8.

Tabela 2.8 Complexidade de Gauss-Seidel usando k iterações em sistema de ordem $n > 1$.

| Operações | Complexidade |
|----------------|-----------------|
| adições | $kn^2 + kn + k$ |
| multiplicações | $(k+1)n^2 - kn$ |
| divisões | $n + k$ |

Exemplo 2.49 Resolver o sistema do Exemplo 2.46 pelo método de Gauss-Seidel com $\varepsilon < 10^{-5}$ e $k_{\max} = 50$ usando os critérios (2.28) e (2.29)

$$\begin{bmatrix} 10 & 3 & -2 \\ 2 & 8 & -1 \\ 1 & 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 57 \\ 20 \\ -4 \end{bmatrix}.$$

Já se sabe, pelas condições (2.27), que o processo convergirá porque a matriz dos coeficientes é diagonalmente dominante. Por (2.34), as equações de iterações são

$$x_1^{k+1} = \frac{1}{10}(-3x_2^k + 2x_3^k + 57),$$

$$x_2^{k+1} = \frac{1}{8}(-2x_1^{k+1} + x_3^k + 20) \text{ e}$$

$$x_3^{k+1} = \frac{1}{5}(-x_1^{k+1} - x_2^{k+1} - 4).$$

Utilizando-se x^0 dado por (2.32), tem-se que $x^0 = [5,7 \ 2,5 \ -0,8]^T$, e pelas equações de iterações, as coordenadas do vetor da primeira iteração são

$$x_1^1 = \frac{1}{10}(-3x_2^0 + 2x_3^0 + 57) = \frac{1}{10}(-3(2,5) + 2(-0,8) + 57) \rightsquigarrow x_1^1 = 4,79,$$

$$x_2^1 = \frac{1}{8}(-2x_1^1 + x_3^0 + 20) = \frac{1}{8}(-2(4,79) + (-0,8) + 20) \rightsquigarrow x_2^1 = 1,2025 \text{ e}$$

$$x_3^1 = \frac{1}{5}(-x_1^1 - x_2^1 - 4) = \frac{1}{5}(-(4,79) - (1,2025) - 4) \rightsquigarrow x_3^1 = -1,9985.$$

Deste modo, o vetor da primeira iteração é $x^1 = [4,79 \ 1,2025 \ -1,9985]^T$ e

$$\frac{\|x^1 - x^0\|_\infty}{\|x^1\|_\infty} = \frac{\max(|4,79 - 5,7|, |1,2025 - 2,5|, |-1,9985 - (-0,8)|)}{\max(|4,79|, |1,2025|, |-1,9985|)},$$

$$\frac{\|x^1 - x^0\|_\infty}{\|x^1\|_\infty} = \frac{\max(0,91; 1,2975; 1,1985)}{\max(4,79; 1,2025; 1,9985)} = 0,2709.$$

Esses e os demais valores das iterações podem ser gerados utilizando-se o algoritmo da Figura 2.10.

```

Algoritmo Gauss-Seidel
{ Objetivo: Resolver o sistema  $Ax = b$  pelo método iterativo de }
{ Gauss-Seidel }
{ parâmetros de entrada n, A, b, Toler, IterMax
  { ordem, matriz, vetor independente, }
  { tolerância e número máximo de iterações } }
{ parâmetros de saída x, Iter, CondErro
  { vetor solução, número de iterações e condição de erro } }
{ construção das matrizes para as iterações }
para i ← 1 até n faça
  r ← 1/A(i,i)
  para j ← 1 até n faça
    se i ≠ j então A(i,j) ← A(i,j) * r, fimse
  fimpara
  b(i) ← b(i) * r; x(i) ← b(i)
fimpara; Iter ← 0
{ iterações de Gauss-Seidel }
repita
  Iter ← Iter + 1
  para i ← 1 até n faça
    Soma ← 0
    para j ← 1 até n faça
      se i ≠ j então Soma ← Soma + A(i,j) * x(j), fimse
    fimpara
    v(i) ← x(i); x(i) ← b(i) - Soma
  fimpara
  NormaNum ← 0; NormaDen ← 0
  para i ← 1 até n faça
    t ← abs(x(i) - v(i))
    se t > NormaNum então NormaNum ← t, fimse
    se abs(x(i)) > NormaDen então NormaDen ← abs(x(i)), fimse
  fimpara
  NormaRel ← NormaNum/NormaDen
  escreva Iter, x, NormaRel
  { teste de convergência }
  se NormaRel ≤ Toler ou Iter ≥ IterMax então
    interrompa, fimse
  fimrepita
  se NormaRel ≤ Toler então CondErro ← 0, senão CondErro ← 1
fimse
finalgoritmo

```

Figura 2.10 Método de Gauss-Seidel para solução do sistema linear $Ax = b$.

(Ver significado da função abs na Tabela 1.1, na página 6.)

```
% Os valores de entrada
n = 3
A =
  10   3   -2
  2   8   -1
  1   1   5
b =
  57
  20
  -4
Toler = 1.0000e-05
IterMax = 50
% produzem os resultados
% Solucao de sistema linear pelo metodo de Gauss-Seidel
Solucao do sistema linear pelo metodo de Gauss-Seidel
Iter   x1      x2      x3      NormaRelativa
  0   5.70000  2.50000 -0.80000
  1   4.79000  1.20250 -1.99850  2.70877e-01
  2   4.93955  1.01530 -1.99097  3.78982e-02
  3   4.99722  1.00182 -1.99981  1.15396e-02
  4   4.99949  1.00015 -1.99993  4.55035e-04
  5   4.99997  1.00002 -2.00000  9.55994e-05
  6   5.00000  1.00000 -2.00000  5.32440e-06
```

Solucao = 5.00000 1.00000 -2.00000

Iter = 6
CondErro = 0

Conseqüentemente, $x \approx x^6 = [5.00000 \ 1.00000 \ -2.00000]^T$. Como na implementação do algoritmo de Jacobi, neste também foram acrescentadas algumas informações para facilitar o entendimento dos resultados.

Exemplo 2.50 Calcular três aproximações do vetor solução do sistema do Exemplo 2.49 usando a formulação (2.33): $x^{k+1} = -(D+E)^{-1}F x^k + (D+E)^{-1}b = Sx^k + d$.

- Decompondo $A = D + E + F$, obtém-se

$$D = \begin{bmatrix} 10 & 0 & 0 \\ 0 & 8 & 0 \\ 0 & 0 & 5 \end{bmatrix}, E = \begin{bmatrix} 0 & 0 & 0 \\ 2 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix} \text{ e } F = \begin{bmatrix} 0 & 3 & -2 \\ 0 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}.$$

A matriz $(D+E)^{-1}$ pode ser calculada utilizando o esquema apresentado na Seção 2.7.2: $(D+E)(D+E)^{-1} = I$. Como a matriz $(D+E)$ é triangular inferior, cada coluna da inversa é obtida pelas substituições sucessivas. Deste modo, as matrizes $(D+E)^{-1}$ e S são

$$(D+E)^{-1} = \begin{bmatrix} 0,1 & 0 & 0 \\ -0,025 & 0,125 & 0 \\ -0,015 & -0,025 & 0,2 \end{bmatrix} \text{ e } S = -(D+E)^{-1}F = \begin{bmatrix} 0 & -0,3 & 0,2 \\ 0 & 0,075 & 0,075 \\ 0 & 0,045 & -0,055 \end{bmatrix}.$$

Os vetores d e x^0 são

$$d = (D+E)^{-1}b = [5,7 \ 1,075 \ -2,155]^T \text{ e } x^0 = D^{-1}b = [5,7 \ 2,5 \ -0,8]^T.$$

As primeiras aproximações da solução por (2.33) são mostradas na tabela a seguir, que, obviamente, são iguais àqueles valores produzidos pelo algoritmo da Figura 2.10.

| k | x_1^k | x_2^k | x_3^k |
|-----|---------|---------|----------|
| 0 | 5,70000 | 2,50000 | -0,80000 |
| 1 | 4,79000 | 1,20250 | -1,99850 |
| 2 | 4,93955 | 1,01530 | -1,99097 |
| 3 | 4,99722 | 1,00182 | -1,99981 |

Exemplo 2.51 Resolver o sistema do Exemplo 2.48 pelo método de Gauss-Seidel com $\varepsilon < 10^{-3}$ e $k_{\max} = 50$ usando os critérios (2.28) e (2.29)

$$\begin{bmatrix} 5 & 2 & 0 & -1 \\ 1 & 8 & -3 & 2 \\ 0 & 1 & 6 & 1 \\ 1 & -1 & 2 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 6 \\ 10 \\ -5 \\ 0 \end{bmatrix}.$$

Pelas condições (2.27), o processo convergirá, pois a matriz dos coeficientes é diagonal estritamente dominante

$$|5| > |2| + |0| + |-1|, |8| > |1| + |-3| + |2|,$$

$$|6| > |0| + |1| + |1| \text{ e } |9| > |1| + |-1| + |2|.$$

Por (2.34), as equações de iterações do método de Gauss-Seidel são

$$x_1^{k+1} = \frac{1}{5}(-2x_2^k + x_4^k + 6),$$

$$x_2^{k+1} = \frac{1}{8}(-x_1^k + 3x_3^k - 2x_4^k + 10),$$

$$x_3^{k+1} = \frac{1}{6}(-x_2^k - x_4^k - 5) \text{ e}$$

$$x_4^{k+1} = \frac{1}{9}(-x_1^k + x_2^k - 2x_3^k).$$

Com x^0 dado por (2.32), $x^0 = [1,2 \ 1,25 \ -0,8333 \ 0]^T$. Pelas equações de iterações, as coordenadas do vetor da primeira iteração são

$$x_1^1 = \frac{1}{5}(-2x_2^0 + x_4^0 + 6) = \frac{1}{5}(-2(1,25) + (0) + 6) = 0,7;$$

$$x_2^1 = \frac{1}{8}(-x_1^0 + 3x_3^0 - 2x_4^0 + 10) = \frac{1}{8}(-(0,7) + 3(-0,8333) - 2(0) + 10) = 0,85;$$

$$x_3^1 = \frac{1}{6}(-x_2^0 - x_4^0 - 5) = \frac{1}{6}(-(0,85) - (0) - 5) = -0,975;$$

$$x_4^1 = \frac{1}{9}(-x_1^0 + x_2^0 - 2x_3^0) = \frac{1}{9}(-(0,7) + (0,85) - 2(-0,975)) = 0,2333.$$

Assim, $x^1 = [0,7 \ 0,85 \ -0,975 \ 0,2333]^T$ e

$$\frac{\|x^1 - x^0\|_\infty}{\|x^1\|_\infty} = \frac{\max(|0,7 - 1,2|, |0,85 - 1,25|, |-0,975 - (-0,8333)|, |0,2333 - 0|)}{\max(|0,7|, |0,85|, |-0,975|, |0,2333|)},$$

$$\frac{\|x^1 - x^0\|_\infty}{\|x^1\|_\infty} = \frac{\max(0,5; 0,4; 0,1417; 0,2333)}{\max(0,7; 0,85; 0,975; 0,2333)} = 0,5128.$$

Usando o algoritmo da Figura 2.10, tem-se

```
% Os valores de entrada
n = 4
A =
  5   2   0   -1
  1   8  -3    2
  0   1   6    1
  1  -1   2    9

b =
  6
 10
 -5
  0

Toler = 1.0000e-03
IterMax = 50
% produzem os resultados

Solucao de sistema linear pelo metodo de Gauss-Seidel
Iter   x1      x2      x3      x4      NormaRelativa
  0   1.20000  1.25000 -0.83333  0.00000
  1   0.70000  0.85000 -0.97500  0.23333  5.12821e-01
  2   0.90667  0.71271 -0.99101  0.19867  2.08542e-01
  3   0.95465  0.70937 -0.98467  0.19156  4.87314e-02
  4   0.95456  0.71354 -0.98418  0.19193  4.22999e-03
  5   0.95297  0.71383 -0.98429  0.19216  1.61801e-03
  6   0.95290  0.71374 -0.98432  0.19216  9.20739e-05

Solucao =  0.95290  0.71374 -0.98432  0.19216
Iter = 6
CondErro = 0
```

Portanto, $x \approx x^6 = [0.95290 \ 0.71374 \ -0.98432 \ 0.19216]^T$.

2.8.5 Método da sobre-relaxação sucessiva

Considere a matriz A decomposta na forma $A = D + E + F$, onde D é uma matriz diagonal e E e F são matrizes triangulares inferior e superior, respectivamente, com diagonais nulas. Multiplicando o sistema linear por um parâmetro ω , tem-se

$$\omega(D + E + F)x = \omega b.$$

Somando o vetor nulo $(D - D)x$ ao primeiro termo,

$$(D - D)x + \omega(D + E + F)x = \omega b$$

e rearranjando,

$$(D + \omega E)x = [(1 - \omega)D - \omega F]x + \omega b,$$

chega-se à forma de iteração

$$(D + \omega E)x^{k+1} = [(1 - \omega)D - \omega F]x^k + \omega b, \quad (2.35)$$

$$x^{k+1} = (D + \omega E)^{-1}[(1 - \omega)D - \omega F]x^k + \omega(D + \omega E)^{-1}b \rightarrow x^{k+1} = Rx^k + e. \quad (2.36)$$

Por (2.26), a matriz $R = (D + \omega E)^{-1}[(1 - \omega)D - \omega F]$ é a matriz de iteração do método denominado sobre-relaxação sucessiva (SOR: successive over-relaxation).

A convergência do método SOR depende da escolha do parâmetro ω e é necessário que $0 < \omega < 2$ para garantir a convergência [3, Teorema 5.4]. Usualmente, utiliza-se $1 < \omega < 2$. Particularmente para $\omega = 1$, a recorrência (2.36) torna-se

$$x^{k+1} = -(D + E)^{-1}Fx^k + (D + E)^{-1}b,$$

que é idêntica à (2.33), ou seja, o método de Gauss-Seidel é um caso particular da sobre-relaxação sucessiva.

Podem-se obter as equações de iterações do SOR a partir de (2.35)

$$\begin{aligned} (D + \omega E)x^{k+1} &= [(1 - \omega)D - \omega F]x^k + \omega b, \\ Dx^{k+1} &= \omega(-Ex^{k+1} - Fx^k + b) + (1 - \omega)Dx^k, \\ x^{k+1} &= \omega D^{-1}(-Ex^{k+1} - Fx^k + b) + (1 - \omega)x^k. \end{aligned}$$

Assim, as equações de iterações do método da sobre-relaxação sucessiva são

$$\left. \begin{aligned} x_1^{k+1} &= \frac{\omega}{a_{11}}(-a_{12}x_2^k - a_{13}x_3^k - \dots - a_{1n}x_n^k + b_1) + (1 - \omega)x_1^k, \\ x_2^{k+1} &= \frac{\omega}{a_{22}}(-a_{21}x_1^k - a_{23}x_3^k - \dots - a_{2n}x_n^k + b_2) + (1 - \omega)x_2^k, \\ x_3^{k+1} &= \frac{\omega}{a_{33}}(-a_{31}x_1^{k+1} - a_{32}x_2^{k+1} - \dots - a_{3n}x_n^k + b_3) + (1 - \omega)x_3^k, \\ &\vdots \\ x_n^{k+1} &= \frac{\omega}{a_{nn}}(-a_{n1}x_1^{k+1} - a_{n2}x_2^{k+1} - \dots - a_{n,n-1}x_{n-1}^{k+1} + b_n) + (1 - \omega)x_n^k. \end{aligned} \right\} \quad (2.37)$$

Um algoritmo para achar a solução de um sistema linear $Ax = b$ pelo método da sobre-relaxação sucessiva, com critério de parada dado por (2.28) e (2.29), é apresentado na Figura 2.11. Os parâmetros de entrada são a ordem n , a matriz A , o vetor b , o parâmetro \Omegamega , a tolerância $Toler$ e o número máximo de iterações $IterMax$. Os parâmetros de saída são a solução x , o número de iterações gastos $Iter$ e a condição de erro $CondErro$ para verificar se a solução convergiu ($CondErro = 0$ significa que houve convergência e $CondErro = 1$, que não houve).

Exemplo 2.52 Resolver o sistema pelo método SOR, com $\varepsilon < 10^{-5}$ e $k_{max} = 500$ usando os critérios (2.28) e (2.29)

$$\begin{bmatrix} 4 & -2 & 1 & 3 & 0 \\ -1 & 10 & 0 & 8 & 1 \\ -1 & 1 & 15 & 2 & 4 \\ 0 & 1 & 10 & 5 & 1 \\ 2 & -3 & 1 & 2 & 20 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 15 \\ 56 \\ 74 \\ 57 \\ 107 \end{bmatrix}.$$

```

Algoritmo SOR
{ Objetivo: Resolver o sistema  $Ax = b$  pelo método iterativo da }
{ sobre-relaxação sucessiva }
parâmetros de entrada  $n, A, b, \Omega\omega, Toler, IterMax$ 
{ ordem, matriz, vetor independente, parâmetro  $\omega$ , }
{ tolerância e número máximo de iterações }
parâmetros de saída  $x, Iter, CondError$ 
{ vetor solução, número de iterações e condição de erro }
{ construção das matrizes para as iterações }
para  $i \leftarrow 1$  até  $n$  faça
   $r \leftarrow 1/A(i,i)$ 
  para  $j \leftarrow 1$  até  $n$  faça
    se  $i \neq j$  então  $A(i,j) \leftarrow A(i,j) * r$ , fimse
  fimpara
   $b(i) \leftarrow b(i) + r; x(i) \leftarrow b(i)$ 
fimpara;  $Iter \leftarrow 0$ 
{ iterações da sobre-relaxação sucessiva }
repita
   $Iter \leftarrow Iter + 1$ 
  para  $i \leftarrow 1$  até  $n$  faça
     $Soma \leftarrow 0$ 
    para  $j \leftarrow 1$  até  $n$  faça
      se  $i \neq j$  então  $Soma \leftarrow Soma + A(i,j) * x(j)$ , fimse
    fimpara
     $v(i) \leftarrow x(i); x(i) \leftarrow \Omega\omega * ((b(i) - Soma) + (1 - \Omega\omega)) * x(i)$ 
  fimpara
   $NormaNum \leftarrow 0; NormaDen \leftarrow 0$ 
  para  $i \leftarrow 1$  até  $n$  faça
     $t \leftarrow \text{abs}(x(i) - v(i))$ 
    se  $t > NormaNum$  então  $NormaNum \leftarrow t$ , fimse
    se  $\text{abs}(x(i)) > NormaDen$  então  $NormaDen \leftarrow \text{abs}(x(i))$ , fimse
  fimpara
   $NormaRel \leftarrow NormaNum / NormaDen$ 
  escreva  $Iter, x, NormaRel$ 
  { teste de convergência }
  se  $NormaRel \leq Toler$  ou  $Iter \geq IterMax$  então interrompa, fimse
fimrepita
se  $NormaRel \leq Toler$  então  $CondError \leftarrow 0$ , senão  $CondError \leftarrow 1$ 
fimse
finalgoritmo

```

Figura 2.11 Método da sobre-relaxação sucessiva para solução do sistema linear $Ax = b$.
(Ver significado da função abs na Tabela 1.1, na página 6.)

A Tabela 2.9 mostra a influência do parâmetro ω no raio espectral ρ da matriz de iteração $R = (D + \omega E)^{-1} [(1 - \omega)D - \omega F]$ do método da sobre-relaxação sucessiva. Quanto menor $\rho(R)$ menor será o número de iterações. O processo não convergiu para $\omega = 1,6$ e $\omega = 1,8$ devido a $\rho(R) > 1$ nos dois casos. Apesar de a matriz A não ser diagonal estritamente dominante o processo convergiu para diversos valores de ω .

Tabela 2.9 Influência do parâmetro ω no raio espectral da matriz de iteração R .

| ω | 0,2 | 0,4 | 0,6 | 0,8 | 1,0 | 1,2 | 1,4 | 1,6 | 1,8 |
|-----------|--------|--------|--------|--------|--------|--------|--------|--------|--------|
| $\rho(R)$ | 0,9357 | 0,8618 | 0,7747 | 0,6674 | 0,5231 | 0,4459 | 0,7506 | 1,0660 | 1,3943 |
| iterações | 118 | 63 | 41 | 29 | 20 | 17 | 44 | >500 | >500 |

O Teorema 2.4 não se aplica ao método da sobre-relaxação sucessiva devido ao parâmetro ω . No Exemplo 2.51, onde a matriz A é diagonal estritamente dominante, o método de Gauss-Seidel converge com 6 iterações. No entanto, a sobre-relaxação sucessiva não converge com $\omega = 1,8$ porque neste caso $\rho(R_{\omega=1,8}) = 1,0131 > 1$.

2.8.6 Análise de convergência

Seja o erro ϵ^k na k -ésima iteração de um processo iterativo

$$\epsilon^k = x^k - x^*,$$

onde x^* é a solução exata do sistema $Ax = b$ de ordem n e x^k é uma aproximação da solução. Substituindo a equação acima para ϵ^{k+1} em (2.26)

$$\epsilon^{k+1} = x^{k+1} - x^* = Mx^k + c - x^*.$$

Sendo $x^k = \epsilon^k + x^*$, então

$$\epsilon^{k+1} = M(\epsilon^k + x^*) + c - x^*,$$

$$\epsilon^{k+1} = M\epsilon^k + (Mx^* + c - x^*). \quad (2.38)$$

Tomando o limite de (2.26),

$$\lim_{k \rightarrow \infty} \epsilon^{k+1} = \lim_{k \rightarrow \infty} Mx^k + c \longrightarrow x^* = Mx^* + c.$$

Substituindo a expressão acima em (2.38), tem-se que a propagação de erro é da forma

$$\epsilon^{k+1} = M\epsilon^k. \quad (2.39)$$

Sendo λ_i um autovalor da matriz de iteração M e v_i o seu correspondente autovetor, então, por (2.2)

$$Mv_i = \lambda_i v_i, i = 1, 2, \dots, n \text{ ou}$$

$$MV = V\Lambda$$

onde $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$ é uma matriz diagonal contendo os autovalores λ_i e V é a matriz composta pelos autovetores v_i , ou seja, $V = [v_1 \ v_2 \ \dots \ v_n]$. Expressando o vetor erro inicial ϵ^0 como uma combinação linear dos autovetores V de M , tem-se

$$\epsilon^0 = Vc$$

sendo c um vetor de coeficientes obtido pela solução do sistema linear acima. Substituindo esta expressão em (2.39), obtém-se

$$\epsilon^1 = M\epsilon^0 = MVc \rightarrow \epsilon^1 = V\Lambda c.$$

De maneira similar,

$$\epsilon^2 = M\epsilon^1 = MV\Lambda c \rightarrow \epsilon^2 = V\Lambda^2 c.$$

Assim, na k -ésima iteração o vetor erro é

$$\epsilon^k = V\Lambda^k c,$$

(2.40)

que é da forma

$$\begin{bmatrix} \epsilon_1^k \\ \epsilon_2^k \\ \vdots \\ \epsilon_n^k \end{bmatrix} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nn} \end{bmatrix} \begin{bmatrix} c_1 \lambda_1^k \\ c_2 \lambda_2^k \\ \vdots \\ c_n \lambda_n^k \end{bmatrix}.$$

Deste modo, quando k aumentar, o vetor erro ϵ^k irá reduzir se, e somente se, o módulo de todos os autovalores λ_i da matriz de iteração M for menor que a unidade. Além do mais, a taxa de convergência será controlada pela magnitude do maior autovalor em módulo, o chamado raio espectral $\rho(M)$ (ver Teorema 2.3).

Exemplo 2.53 Seja o sistema $Ax = b$ do Exemplo 2.49 e sua solução exata x^*

$$\begin{bmatrix} 10 & 3 & -2 \\ 2 & 8 & -1 \\ 1 & 1 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 57 \\ 20 \\ -4 \end{bmatrix} \text{ e } x^* = \begin{bmatrix} 5 \\ 1 \\ -2 \end{bmatrix}.$$

A partir dos resultados do Exemplo 2.50, obtém-se os valores para a fórmula de recorrência de Gauss-Seidel $x^{k+1} = Sx^k + d$

$$S = \begin{bmatrix} 0 & -0,300 & 0,200 \\ 0 & 0,075 & 0,075 \\ 0 & 0,045 & -0,055 \end{bmatrix}, d = \begin{bmatrix} 5,700 \\ 1,075 \\ -2,155 \end{bmatrix} \text{ e } x^0 = \begin{bmatrix} 5,7 \\ 2,5 \\ -0,8 \end{bmatrix}.$$

O vetor erro inicial é $\epsilon^0 = x^0 - x^* = [0,7 \ 1,5 \ 1,2]^T$. Os autovalores Λ da matriz de iteração S e seus respectivos autovetores V são

$$\Lambda = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0,09718 & 0 \\ 0 & 0 & -0,07718 \end{bmatrix} \text{ e } V = \begin{bmatrix} 1 & -0,92174 & -0,97074 \\ 0 & 0,37189 & -0,10615 \\ 0 & 0,10997 & 0,21538 \end{bmatrix}.$$

Por (2.40) o vetor erro na k -ésima iteração é

$$\begin{bmatrix} \epsilon_1^k \\ \epsilon_2^k \\ \epsilon_3^k \end{bmatrix} = \begin{bmatrix} 1 & -0,92174 & -0,97074 \\ 0 & 0,37189 & -0,10615 \\ 0 & 0,10997 & 0,21538 \end{bmatrix} \begin{bmatrix} 8,19997(0)^k \\ 4,90841(0,09718)^k \\ 3,06538(-0,07718)^k \end{bmatrix},$$

sendo o vetor c a solução do sistema linear $Vc = \epsilon^0$. A partir da equação acima, calcula-se o vetor erro ϵ^k a cada iteração. Como $x^k = \epsilon^k + x^*$, é possível obter a solução aproximada x^k para comparar com os valores mostrados no Exemplo 2.50

| k | ϵ_1^k | ϵ_2^k | ϵ_3^k | $\epsilon_1^k - x_1^*$ | $\epsilon_2^k - x_2^*$ | $\epsilon_3^k - x_3^*$ |
|-----|----------------|----------------|----------------|------------------------|------------------------|------------------------|
| 0 | 0,70000 | 1,50000 | 1,20000 | 5,70000 | 2,50000 | -0,80000 |
| 1 | -0,21001 | 0,20250 | 0,00150 | 4,78999 | 1,20250 | -1,99850 |
| 2 | -0,06045 | 0,01530 | 0,00093 | 4,93955 | 1,01530 | -1,99097 |
| 3 | -0,00278 | 0,00182 | 0,00019 | 4,99722 | 1,00182 | -1,99981 |

Como $|\lambda_i| < 1 \forall i$, então

$$\lim_{k \rightarrow \infty} \lambda_i^k = 0 \rightarrow \lim_{k \rightarrow \infty} \epsilon^k = 0,$$

fazendo com que o processo converja. Se pelo menos um $|\lambda_i| > 1$, então

$$\lim_{k \rightarrow \infty} \lambda_i^k = \infty \rightarrow \lim_{k \rightarrow \infty} \epsilon^k = \infty,$$

implicando uma divergência da solução.

2.8.7 Comparação dos métodos iterativos estacionários

Pelo Teorema 2.4, se a matriz dos coeficientes A for diagonal estritamente dominante, então a solução pelos métodos de Jacobi e Gauss-Seidel converge. No entanto, se A não for diagonalmente dominante, a previsão de convergência deve ser feita pelo Teorema 2.3 usando o raio espectral $\rho(M)$ da matriz de iteração M . Neste caso, um método pode convergir e o outro não.

Exemplo 2.54 Verificar se o sistema abaixo pode ser resolvido pelo método de Jacobi ou de Gauss-Seidel

$$\begin{bmatrix} 0,5 & 0,6 & 0,3 \\ 1 & 1 & 1 \\ 0,4 & -0,4 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0,2 \\ 0 \\ -0,6 \end{bmatrix}.$$

Como a matriz não é diagonal estritamente dominante, faz-se necessário o uso do Teorema 2.3. As matrizes de iteração dos dois métodos são, por (2.30) e (2.33)

$$J = -D^{-1}(E + F) = \begin{bmatrix} 0 & -1,2 & -0,6 \\ -1 & 0 & -1 \\ -0,4 & 0,4 & 0 \end{bmatrix} \rightsquigarrow \rho(J) = 1,1200 \text{ e}$$

$$S = -(D + E)^{-1}F = \begin{bmatrix} 0 & -1,2 & -0,6 \\ 0 & 1,2 & -0,4 \\ 0 & 0,96 & 0,08 \end{bmatrix} \rightsquigarrow \rho(S) = 0,6928.$$

Como $\rho(J) > 1$ e $\rho(S) < 1$, então pelo método de Jacobi a solução não irá convergir, enquanto pelo de Gauss-Seidel irá. De fato, as dez primeiras iterações confirmam

Solução de sistema linear pelo método de Jacobi

| Iter | x1 | x2 | x3 | NormaRelativa |
|------|---------|----------|----------|---------------|
| 0 | 0.40000 | 0.00000 | -0.60000 | |
| 1 | 0.76000 | 0.20000 | -0.76000 | 4.73684e-01 |
| 2 | 0.61600 | 0.00000 | -0.82400 | 2.42718e-01 |
| 3 | 0.89440 | 0.20800 | -0.84640 | 3.11270e-01 |
| 4 | 0.65824 | -0.04800 | -0.87456 | 2.92719e-01 |
| 5 | 0.98234 | 0.21632 | -0.88250 | 3.29924e-01 |
| 6 | 0.66991 | -0.09984 | -0.90641 | 3.48806e-01 |
| 7 | 1.06365 | 0.23649 | -0.90790 | 3.70176e-01 |
| 8 | 0.66095 | -0.15575 | -0.93086 | 4.32612e-01 |
| 9 | 1.14542 | 0.26991 | -0.92668 | 4.22963e-01 |
| 10 | 0.63211 | -0.21874 | -0.95020 | 5.40209e-01 |

Solução de sistema linear pelo método de Gauss-Seidel

| Iter | x1 | x2 | x3 | NormaRelativa |
|------|---------|----------|----------|---------------|
| 0 | 0.40000 | 0.00000 | -0.60000 | |
| 1 | 0.76000 | -0.16000 | -0.96800 | 3.80165e-01 |
| 2 | 1.17280 | -0.20480 | -1.15104 | 3.51978e-01 |
| 3 | 1.33638 | -0.18534 | -1.20869 | 1.22408e-01 |
| 4 | 1.34763 | -0.13894 | -1.19463 | 3.44366e-02 |
| 5 | 1.28350 | -0.08887 | -1.14895 | 4.99639e-02 |
| 6 | 1.19602 | -0.04707 | -1.09723 | 7.31440e-02 |
| 7 | 1.11482 | -0.01759 | -1.05296 | 7.28318e-02 |
| 8 | 1.05288 | 0.00008 | -1.02112 | 5.88269e-02 |
| 9 | 1.01258 | 0.00854 | -1.00161 | 3.98065e-02 |
| 10 | 0.99071 | 0.01090 | -0.99193 | 2.20409e-02 |

Exemplo 2.55 Verificar se o sistema a seguir pode ser resolvido pelo método de Jacobi ou de Gauss-Seidel

$$\begin{bmatrix} 0,5 & 0,6 & 0,3 \\ 1 & -1 & 1 \\ 0,4 & -0,4 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 0,2 \\ 0 \\ -0,6 \end{bmatrix}.$$

As matrizes de iteração são

$$J = -D^{-1}(E + F) = \begin{bmatrix} 0 & -1,2 & -0,6 \\ 1 & 0 & 1 \\ -0,4 & 0,4 & 0 \end{bmatrix} \rightsquigarrow \rho(J) = 0,8266 \text{ e}$$

$$S = -(D + E)^{-1}F = \begin{bmatrix} 0 & -1,2 & -0,6 \\ 0 & -1,2 & 0,4 \\ 0 & 0 & 0,4 \end{bmatrix} \rightsquigarrow \rho(S) = 1,2000.$$

Como $\rho(J) < 1$ e $\rho(S) > 1$, pelo método de Jacobi a solução irá convergir, mas pelo de Gauss-Seidel não irá. Para certificar, as dez primeiras iterações mostram

Solução de sistema linear pelo método de Jacobi

| Iter | x1 | x2 | x3 | NormaRelativa |
|------|---------|----------|----------|---------------|
| 0 | 0.40000 | 0.00000 | -0.60000 | |
| 1 | 0.76000 | -0.20000 | -0.76000 | 4.73684e-01 |
| 2 | 1.09600 | 0.00000 | -0.98400 | 3.06569e-01 |
| 3 | 0.99040 | 0.11200 | -1.03840 | 1.07858e-01 |
| 4 | 0.88864 | -0.04800 | -0.95136 | 1.68180e-01 |
| 5 | 1.02842 | -0.06272 | -0.97466 | 1.35914e-01 |
| 6 | 1.06006 | 0.05376 | -1.03645 | 1.09881e-01 |
| 7 | 0.95736 | 0.02360 | -1.00252 | 1.02439e-01 |
| 8 | 0.97319 | -0.04516 | -0.97350 | 7.06332e-02 |
| 9 | 1.03829 | -0.00032 | -1.00734 | 6.27033e-02 |
| 10 | 1.00478 | 0.03095 | -1.01544 | 3.30007e-02 |

Solução de sistema linear pelo método de Gauss-Seidel

| Iter | x1 | x2 | x3 | NormaRelativa |
|------|---------|----------|----------|---------------|
| 0 | 0.40000 | 0.00000 | -0.60000 | |
| 1 | 0.76000 | 0.16000 | -0.84000 | 4.28571e-01 |
| 2 | 0.71200 | -0.12800 | -0.93600 | 3.07692e-01 |
| 3 | 1.11520 | 0.17920 | -0.97440 | 3.61549e-01 |
| 4 | 0.76960 | -0.20480 | -0.98976 | 3.87973e-01 |
| 5 | 1.23962 | 0.24986 | -0.99590 | 3.79163e-01 |
| 6 | 0.69772 | -0.29819 | -0.99836 | 5.48944e-01 |
| 7 | 1.35684 | 0.35848 | -0.99934 | 4.85781e-01 |
| 8 | 0.56943 | -0.42992 | -0.99974 | 7.88605e-01 |
| 9 | 1.51574 | 0.51600 | -0.99990 | 6.24324e-01 |
| 10 | 0.38073 | -0.61916 | -0.99996 | 1.13522e+00 |

É curioso notar que os sistemas dos Exemplos 2.54 e 2.55 diferem apenas no elemento a_{22} , que possui sinais contrários. A mudança de sinal daquele coeficiente faz com que a solução convirja por um método e não convirja por outro. Quanto menor o valor de $\rho(M)$, mais rápida será a convergência do método iterativo (ver (2.40)). Para o sistema do Exemplo 2.46, as matrizes de iteração dos dois métodos são

$$J = \begin{bmatrix} 0 & -0,3 & 0,2 \\ -0,25 & 0 & 0,125 \\ -0,2 & -0,2 & 0 \end{bmatrix} \rightsquigarrow \rho(J) = 0,2725 \text{ e}$$

$$S = \begin{bmatrix} 0 & -0,3 & 0,2 \\ 0 & 0,075 & 0,075 \\ 0 & 0,045 & -0,055 \end{bmatrix} \rightsquigarrow \rho(S) = 0,0972.$$

Considerando que $\rho(S) < \rho(J)$, então o método de Gauss-Seidel converge mais rápido, pois ele gasta 6 iterações (Exemplo 2.49) contra 9 do método de Jacobi (Exemplo 2.46).

2.8.8 Refinamento como método estacionário

Na Seção 2.7.1, foi mostrado como refinar a solução de um sistema linear a partir dos fatores da decomposição. Aquelas equações podem ser rearranjadas

$$x^{k+1} = x^k + c^k$$

$$\begin{aligned}
 &= x^k + U^{-1}L^{-1}Pr^k \\
 &= x^k + U^{-1}L^{-1}P(b - Ax^k) \\
 x^{k+1} &= x^k - U^{-1}L^{-1}PAx^k + U^{-1}L^{-1}Pb,
 \end{aligned}$$

resultando em

$$x^{k+1} = (I - U^{-1}L^{-1}PA)x^k + U^{-1}L^{-1}Pb, \quad (2.41)$$

que é da forma (2.26) de um método iterativo estacionário.

Exemplo 2.56 Resolver o sistema do Exemplo 2.44, na página 86, utilizando a formulação mostrada em (2.41)

$$\begin{bmatrix} 2 & 3 & -1 \\ -3 & 5 & 6 \\ 1 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -4 \\ 19 \\ 11 \end{bmatrix}.$$

Os três fatores obtidos pela decomposição LU com pivotação parcial são

$$L = \begin{bmatrix} 1 & 0 & 0 \\ -0,67 & 1 & 0 \\ -0,33 & 0,42 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} -3 & 5 & 6 \\ 0 & 6,33 & 3 \\ 0 & 0 & 2,74 \end{bmatrix} \text{ e } P = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

A matriz de iteração é

$$(I - U^{-1}L^{-1}PA) = \begin{bmatrix} -3,6384 & 2,2421 & 7,2768 \\ 4,0359 & -6,1000 & -8,0719 \\ -5,1825 & 6,2044 & 10,365 \end{bmatrix} \times 10^{-3},$$

e os vetores

$$U^{-1}L^{-1}Pb = x^0 = [1,9731 \ -0,9738 \ 4,9647]^T.$$

As iterações calculadas por (2.41) são exibidas na tabela a seguir para comparação com os resultados do Exemplo 2.44

| k | x_1^k | x_2^k | x_3^k |
|-----|---------|---------|---------|
| 0 | 1,9731 | -0,9738 | 4,9647 |
| 1 | 1,9999 | -1,0000 | 4,9999 |
| 2 | 2,0000 | -1,0000 | 5,0000 |

O raio espectral da matriz de iteração $\rho(I - U^{-1}L^{-1}PA) = 6,2355 \times 10^{-4} < 1$. Por esta razão, o processo convergiu. Caso a perturbação nos fatores L e U seja grande o suficiente para $\rho(I - U^{-1}L^{-1}PA) \geq 1$, então o processo não mais convergirá.

2.9 Análise de erro na solução de sistemas

É importante analisar como pequenas variações nos elementos da matriz dos coeficientes A e/ou no vetor de termos independentes b influenciam a solução x do sistema linear $Ax = b$.

2.9.1 Malcondicionamento

Considere o sistema linear $Ax = b$, com

$$A = \begin{bmatrix} 1 & 0,99 \\ 0,99 & 0,98 \end{bmatrix} \text{ e } b = \begin{bmatrix} 1,99 \\ 1,97 \end{bmatrix},$$

cujas soluções exatas é $x = [1 \ 1]^T$. Seja o vetor $\tilde{b} = [1,99 \ 1,98]^T \approx b$. A solução exata do sistema $Ay = \tilde{b}$ é $y = [100 \ -99]^T$. Portanto, uma pequena perturbação no vetor de termos independentes ocasionou uma grande modificação no vetor solução. Considere agora a matriz

$$\tilde{A} = \begin{bmatrix} 1 & 0,99 \\ 0,99 & 0,99 \end{bmatrix} \approx A.$$

A solução exata do sistema $\tilde{A}z = b$ é $z = [2 \ -1/99]^T$. Neste caso, foi uma pequena perturbação na matriz dos coeficientes que acarretou uma grande mudança no vetor solução. Estes problemas são causados porque a matriz A é quase singular ($\det(A) = -10^{-4}$). Um sistema linear com uma matriz com esta característica é chamado malcondicionado.

A Figura 2.12 mostra os três planos definidos por um sistema linear, e dois planos são quase coincidentes. Deste modo, pequenas variações em qualquer um dos três planos causam um grande deslocamento no ponto de interseção, que é a solução do sistema.

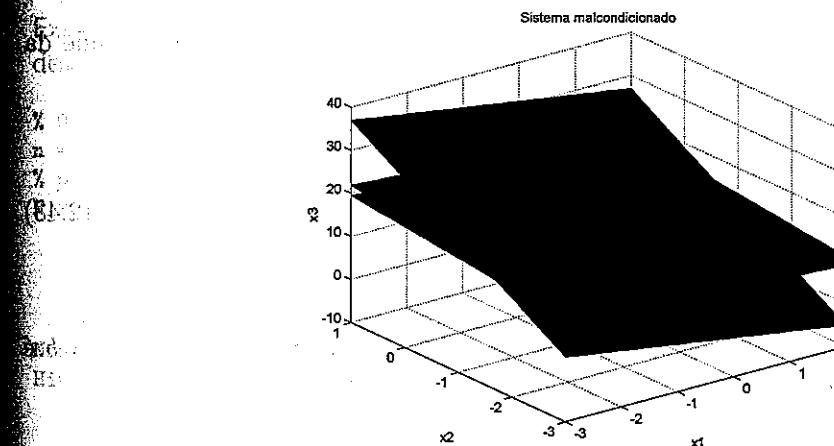


Figura 2.12 Interpretação geométrica do malcondicionamento.

A solução exata de $Ax = b$ é $x = [1 \ 1]^T$; no entanto, o resíduo para $\tilde{x} = [0,9 \ 1,1]^T$ é

$$\tilde{r} = b - A\tilde{x} = \begin{bmatrix} 1,99 \\ 1,97 \end{bmatrix} - \begin{bmatrix} 1 & 0,99 \\ 0,99 & 0,98 \end{bmatrix} \begin{bmatrix} 0,9 \\ 1,1 \end{bmatrix} \rightsquigarrow \tilde{r} = \begin{bmatrix} 10^{-3} \\ 10^{-3} \end{bmatrix}.$$

Apesar de \tilde{x} ser diferente da solução exata x , o resíduo \tilde{r} é pequeno. Quando a solução for quase exata, o resíduo será pequeno, porém a recíproca não é verdadeira para sistemas

malcondicionados. Conseqüentemente, o resíduo não será um bom indicador de exatidão da solução quando o sistema for malcondicionado. O grande problema em resolver sistemas malcondicionados é a instabilidade da solução. Se A e/ou b forem medidas experimentais e, portanto, sujeitas a erros, qualquer pequena variação em seus elementos acarretará bruscas alterações no vetor solução.

2.9.2 Número de condição

O malcondicionamento é devido à quase singularidade da matriz A dos coeficientes. No entanto, medir a singularidade de A pelo seu determinante não constitui uma boa prática porque um determinante pequeno não indica necessariamente a ocorrência de um malcondicionamento. Por exemplo, a matriz

$$A = \begin{bmatrix} 0,001 & 0,001 \\ -0,001 & 0,001 \end{bmatrix}$$

tem $\det(A) = 2 \times 10^{-6}$ e, no entanto, é muito bem-condicionada. Seu determinante é pequeno porque seus elementos são pequenos.

O número de condição de uma matriz é mais apropriado para medir o quanto a matriz é malcondicionada. O número de condição é definido como o produto de duas normas, a saber,

$$\text{condição}(A) = \kappa(A) = \|A\| \|A^{-1}\|, \quad (2.42)$$

onde $\|\cdot\|$ significa uma norma matricial qualquer. Portanto, o valor de $\kappa(A)$ depende da norma utilizada. Particularmente, em vista de (2.5), $\kappa_2(A)$ é dado por

$$\kappa_2(A) = \begin{cases} \frac{\lambda_{\max}}{\lambda_{\min}} & \text{se } A = A^T \\ \frac{\sigma_{\max}}{\sigma_{\min}} & \text{se } A \neq A^T \end{cases} \quad (2.43)$$

desde que $\lambda(A^{-1}) = \lambda^{-1}(A)$, conforme mostrado na Seção 2.1.3. Um sistema $Ax = b$ é considerado malcondicionado se o número de condição for grande, $\kappa(A) \gg 1$.

Exemplo 2.57 Calcular $\kappa_2(A)$ e $\kappa_2(B)$ para

$$A = \begin{bmatrix} 1 & 0,99 \\ 0,99 & 0,98 \end{bmatrix} \text{ e } B = \begin{bmatrix} 5 & 3 & 0 \\ -1 & 8 & 6 \\ 4 & 2 & 9 \end{bmatrix}.$$

Por (2.43),

$$\lambda(A) = (1,9801, -5,0504 \times 10^{-5}) \sim$$

$$\kappa_2(A) = \frac{|1,9801|}{|-5,0504 \times 10^{-5}|} = 3,9206 \times 10^4 \text{ e}$$

$$\lambda(B^T B) = (1,7423 \times 10^2, 3,7222 \times 10^1, 2,4548 \times 10^1) \sim$$

$$\kappa_2(B) = \sqrt{\frac{1,7423 \times 10^2}{2,4548 \times 10^1}} = 2,6641.$$

Para este exemplo, um sistema linear com a matriz A será malcondicionado, enquanto um sistema com a matriz B será bem-condicionado. ■

Um exemplo clássico de malcondicionamento é a matriz de Hilbert definida por

$$H_n = \frac{1}{i+j-1}, \quad i, j = 1, 2, \dots, n.$$

Por exemplo,

$$H_2 = \begin{bmatrix} 1 & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{3} \end{bmatrix} \text{ e } H_3 = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{bmatrix}.$$

A Figura 2.13 apresenta um algoritmo para gerar a matriz de Hilbert de ordem n e sua inversa, bem como as respectivas normas- ∞ . O parâmetro de entrada é a ordem n da matriz. Os parâmetros de saída são H , contendo a matriz de Hilbert H_n , $Hinv$ contendo a inversa H_n^{-1} , $NinfH$ com $\|H_n\|_\infty$ e $NinfHinv$ com $\|H_n^{-1}\|_\infty$.

Exemplo 2.58 Gerar a matriz de Hilbert de ordem 4 e sua inversa utilizando o algoritmo descrito na Figura 2.13.

```
% 0 parametro de entrada
n = 4
% produz os resultados
H =
    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
    0.2500    0.2000    0.1667    0.1429
NinfH = 2.0833
Hinv =
    16        -120       240      -140
   -120       1200      -2700      1680
    240      -2700       6480     -4200
   -140      1680      -4200      2800
NinfHinv = 13620
```

Considerando que

$$\|H_4\|_\infty = \frac{25}{12} \approx 2,0833 \text{ e } \|H_4^{-1}\|_\infty = 13620,$$

então H_4 é de fato malcondicionada, desde que, por (2.42)

$$\kappa_\infty(H_4) = \|H_4\|_\infty \|H_4^{-1}\|_\infty = \frac{25}{12} 13620 = 28375. \blacksquare$$

```

Algoritmo Hilbert
{ Objetivo: Gerar uma matriz de Hilbert, sua inversa e as normas- $\infty$  }
parâmetros de entrada  $n$  { ordem da matriz }
parâmetros de saída  $H$ ,  $Hinv$ ,  $NinfH$ ,  $NinfHinv$ 
{  $H_n$ ,  $H_n^{-1}$ ,  $\|H_n\|_\infty$ ,  $\|H_n^{-1}\|_\infty$  }
{ cálculo da matriz de Hilbert e sua norma- $\infty$  }
para  $i \leftarrow 1$  até  $n$  faça
     $H(i,i) \leftarrow 1/(2*i - 1)$ 
    para  $j \leftarrow i + 1$  até  $n$  faça
         $H(i,j) \leftarrow 1/(i+j-1)$ ;  $H(j,i) \leftarrow H(i,j)$ 
    fimpara
fimpara
 $NinfH \leftarrow 0$ 
para  $j \leftarrow 1$  até  $n$  faça
     $NinfH \leftarrow NinfH + H(1,j)$ 
fimpara
{ inversa da matriz de Hilbert e sua norma- $\infty$  }
{ linha 1 }
 $Hinv(1,1) \leftarrow n^2$ ;  $Prod1 \leftarrow 1$ ;  $Soma \leftarrow Hinv(1,1)$ 
para  $j \leftarrow 2$  até  $n$  faça
     $Prod1 \leftarrow Prod1 * (1 - (n/(j-1))^2)$ 
     $Hinv(1,j) \leftarrow n^2/j * Prod1$ ;  $Hinv(j,1) \leftarrow Hinv(1,j)$ 
     $Soma \leftarrow Soma + abs(Hinv(1,j))$ 
fimpara
 $NinfHinv \leftarrow Soma$ 
{ linha  $i > 1$  }
 $Prod2 \leftarrow 1$ 
para  $i \leftarrow 2$  até  $n$  faça
     $Prod1 \leftarrow 1$ ;  $Prod2 \leftarrow Prod2 * ((n/(i-1))^2 - 1)$ 
     $Hinv(i,i) \leftarrow (n * Prod2)^2 / (2 * i - 1)$ ;  $Soma \leftarrow Hinv(i,i)$ 
    para  $j \leftarrow i + 1$  até  $n$  faça
         $Prod1 \leftarrow Prod1 * (1 - (n/(j-1))^2)$ 
         $Hinv(i,j) \leftarrow (n * Prod2)^2 * Prod1 / (i+j-1)$ 
         $Hinv(j,i) \leftarrow Hinv(i,j)$ ;  $Soma \leftarrow Soma + abs(Hinv(i,j))$ 
    fimpara
    para  $j \leftarrow 1$  até  $i - 1$  faça
         $Soma \leftarrow Soma + abs(Hinv(i,j))$ 
    fimpara
    se  $Soma > NinfHinv$  então  $NinfHinv \leftarrow Soma$ , fimse
fimpara
finalgoritmo

```

Figura 2.13 Algoritmo gerador de matriz de Hilbert, sua inversa e normas- ∞ .
(Ver significado da função abs na Tabela 1.1, na página 6.)

A Tabela 2.10 mostra os valores de $\|H_n\|_\infty$, $\|H_n^{-1}\|_\infty$ e $\kappa_\infty(H_n)$ para $n=1, 2, \dots, 10$, obtidos pelo algoritmo da Figura 2.13. À medida que n aumenta, H_n vai se tornando cada vez mais malcondicionada.

Tabela 2.10 Normas- ∞ e número de condição das matrizes de Hilbert.

| n | $\ H_n\ _\infty$ | $\ H_n^{-1}\ _\infty$ | $\kappa_\infty(H_n)$ |
|-----|------------------|--------------------------|--------------------------|
| 1 | 1,00000 | $1,00000 \times 10^0$ | $1,00000 \times 10^0$ |
| 2 | 1,50000 | $1,80000 \times 10^1$ | $2,70000 \times 10^1$ |
| 3 | 1,83333 | $4,08000 \times 10^2$ | $7,48000 \times 10^2$ |
| 4 | 2,08333 | $1,36200 \times 10^4$ | $2,83750 \times 10^4$ |
| 5 | 2,28333 | $4,13280 \times 10^5$ | $9,43656 \times 10^5$ |
| 6 | 2,45000 | $1,18654 \times 10^7$ | $2,90703 \times 10^7$ |
| 7 | 2,59286 | $3,79965 \times 10^8$ | $9,85195 \times 10^8$ |
| 8 | 2,71786 | $1,24631 \times 10^{10}$ | $3,38728 \times 10^{10}$ |
| 9 | 2,82897 | $3,88712 \times 10^{11}$ | $1,09965 \times 10^{12}$ |
| 10 | 2,92897 | $1,20716 \times 10^{13}$ | $3,53574 \times 10^{13}$ |

2.9.3 Sensibilidade da solução

Seja o sistema $Ax = b$ e uma pequena perturbação δb em b . A correspondente modificação δx na solução $x = A^{-1}b$ satisfaz

$$\delta x = A^{-1}\delta b.$$

Pelas propriedades das normas consistentes (ver Seção 2.1.4), tem-se

$$\|A\|\|x\| \geq \|b\| \text{ e } \|\delta x\| \leq \|A^{-1}\|\|\delta b\|,$$

cuja combinação resulta em

$$\frac{\|\delta x\|}{\|x\|} \leq \|A\|\|A^{-1}\| \frac{\|\delta b\|}{\|b\|}.$$

Em vista de (2.42)

$$\frac{\|\delta x\|}{\|x\|} \leq \kappa(A) \frac{\|\delta b\|}{\|b\|}. \quad (2.44)$$

Esta equação fornece um limite superior ao erro relativo na solução x devido à perturbação δb no vetor de termos independentes b . Quanto mais malcondicionado um sistema $Ax = b$ for, maior será o número de condição $\kappa(A)$ e, consequentemente, maior será o erro relativo na solução.

Exemplo 2.59 Verificar (2.44) para o sistema $Ax = b$, com

$$A = \begin{bmatrix} 1 & 0,99 \\ 0,99 & 0,98 \end{bmatrix}, b = \begin{bmatrix} 1,99 \\ 1,97 \end{bmatrix} \text{ e } \delta b = \begin{bmatrix} 0 \\ 0,01 \end{bmatrix}.$$

Sendo $x = [1 \ 1]^T$ a solução exata de $Ax = b$, $\kappa_2(A) = 3,9206 \times 10^4$ (ver Exemplo 2.57), $\|b\|_2 = 2,8002$ e $\|\delta b\|_2 = 10^{-2}$, tem-se que o limite superior ao erro relativo em termos da norma-2 é

$$\frac{\|\delta x\|_2}{\|x\|_2} \leq \kappa_2(A) \frac{\|\delta b\|_2}{\|b\|_2} = 3,9206 \times 10^4 \frac{10^{-2}}{2,8002} \sim \frac{\|\delta x\|_2}{\|x\|_2} \leq 1,4001 \times 10^2.$$

De fato, com a perturbação δb , a solução variou de $[1 \ 1]^T$ para $[100 \ -99]^T$, significando $\delta x = [100-1 \ -99-1]^T \sim \|\delta x\|_2 = 1,4072 \times 10^2$. Considerando que $\|x\|_2 = 1,4142$, na realidade, o erro relativo cometido foi

$$\frac{\|\delta x\|_2}{\|x\|_2} = \frac{1,4072 \times 10^2}{1,4142} = 9,9505 \times 10^1$$

que está dentro do limite previsto por (2.44).

Para considerar a perturbação em A , seja

$$(A + \delta A)(x + \delta x) = b \rightarrow Ax + A\delta x + \delta A x + \delta A \delta x = b \sim A\delta x = -\delta A(x + \delta x).$$

Tomando as normas consistentes, tem-se

$$\|\delta x\| \leq \|A^{-1}\| \|\delta A\| \|x + \delta x\| \text{ ou}$$

$$\frac{\|\delta x\|}{\|x + \delta x\|} \leq \kappa(A) \frac{\|\delta A\|}{\|A\|}. \quad (2.45)$$

Quanto mais malcondicionado um sistema $Ax = b$ for, maior será a influência da perturbação δA em A na solução x . Se, por exemplo, os coeficientes de A forem conhecidos com a precisão de quatro decimais e o número de condição for 10^3 , então x pode ter somente a precisão de uma decimal.

Exemplo 2.60 Verificar (2.45) para o sistema $Ax = b$, com

$$A = \begin{bmatrix} 1 & 0,99 \\ 0,99 & 0,98 \end{bmatrix}, \quad b = \begin{bmatrix} 1,99 \\ 1,97 \end{bmatrix} \text{ e } \delta A = \begin{bmatrix} 0 & 0 \\ 0 & 0,01 \end{bmatrix}.$$

Sendo $\kappa_2(A) = 3,9206 \times 10^4$ (ver Exemplo 2.57), $\|\delta A\|_2 = 10^{-2}$ e $\|A\|_2 = 1,9801$, então o erro relativo em termos da norma-2 é

$$\frac{\|\delta x\|_2}{\|x + \delta x\|_2} \leq \kappa_2(A) \frac{\|\delta A\|_2}{\|A\|_2} = 3,9206 \times 10^4 \frac{10^{-2}}{1,9801} \sim \frac{\|\delta x\|_2}{\|x + \delta x\|_2} \leq 1,9800 \times 10^2.$$

Com a perturbação δA , a solução variou de $x = [1 \ 1]^T$ para $\tilde{x} = [2 \ -1/99]^T$. Assim, a variação na solução foi $\delta x = [2-1 \ -1/99-1]^T$ e o erro relativo real

$$\frac{\|\delta x\|_2}{\|x + \delta x\|_2} = \frac{1,4214}{2,0000} = 7,1070 \times 10^{-1},$$

portanto, está dentro do limite previsto por (2.45).

Os limites aos erros relativos, (2.44) e (2.45), podem ser pessimistas conforme visto neste exemplo. Contudo, essas relações são rigorosas e simples de expressar.

2.10 Exemplos de aplicação

Nesta seção, será mostrado o uso de sistemas de equações lineares para resolver dois problemas práticos, um de Física (cálculo de tensões em um circuito resistivo) e outro de Química (estequiometria de reação química).

2.10.1 Tensões em circuito elétrico

Definição do problema

Calcular as tensões dos nós do circuito elétrico da Figura 2.14.

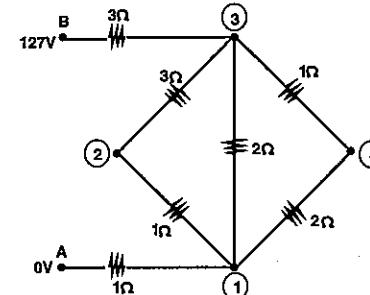


Figura 2.14 Circuito elétrico resistivo.

Modelagem matemática

Pela lei de Kirchhoff, a soma das correntes que passam em cada nó do circuito é nula. Pela lei de Ohm, a corrente elétrica que flui do nó j para o nó k de um circuito é $I_{jk} = \frac{V_j - V_k}{R_{jk}}$, sendo V_j e V_k as tensões nos nós j e k , respectivamente, e R_{jk} a resistência no arco jk . A corrente I é expressa em ampères e a resistência R , em ohms. As duas leis combinadas permitem o cálculo da tensão em cada nó do circuito.

No nó 1, pela lei de Kirchhoff, $I_{A1} + I_{21} + I_{31} + I_{41} = 0$. Utilizando a lei de Ohm,

$$\frac{0 - V_1}{1} + \frac{V_2 - V_1}{1} + \frac{V_3 - V_1}{2} + \frac{V_4 - V_1}{2} = 0 \rightarrow -6V_1 + 2V_2 + V_3 + V_4 = 0.$$

Determinando as equações dos nós 2, 3 e 4, constrói-se um sistema de equações lineares. O vetor solução fornece a tensão em cada nó do circuito elétrico

$$\begin{bmatrix} -6 & 2 & 1 & 1 \\ 3 & -4 & 1 & 0 \\ 3 & 2 & -13 & 6 \\ 1 & 0 & 2 & -3 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ V_4 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ -254 \\ 0 \end{bmatrix}.$$

Solução numérica

Resolvendo o sistema linear $AV = b$, formado pelas equações dos nós 1, 2, 3 e 4, usando decomposição LU com pivotação parcial como no Exemplo 2.33, obtém-se as tensões em cada nó do circuito.

```
% Os valores de entrada
n = 4
A =
-6   2   1   1
 3  -4   1   0
 3   2  -13   6
 1   0   2  -3
% produzem os resultados pela decomposicao LU
A =
-6.0000  2.0000  1.0000  1.0000
-0.5000 -3.0000  1.5000  0.5000
-0.5000 -1.0000 -11.0000  7.0000
-0.1667 -0.1111 -0.2121 -1.2929
Det = 256.0000
Pivot = 1 2 3 4
% vetor de termos independentes
b =
 0
 0
-254
 0
% As substituicoes sucessivas pivotal produzem
y =
 0
 0
-254.0000
-53.8788
% As substituicoes retroativas resultam em
x =
25.7969
31.7500
49.6094
41.6719
As tensões em cada nó do circuito elétrico dado são
V1 = 25,80 V, V2 = 31,75 V, V3 = 49,61 V e V4 = 41,67 V.
```

Análise dos resultados

Nesta etapa, verifica-se a adequação da solução numérica do modelo com relação aos valores possíveis das tensões. Para o circuito dado, a tensão de cada nó deve estar entre 0 e 127 V.

2.10.2 Estequiométria de reação química

Definição do problema

Equilibrar a equação química



Modelagem matemática

O balanceamento de uma equação química é baseado na lei de conservação da massa de Lavoisier: "Em um sistema químico isolado, a massa permanece constante, quaisquer que

sejam as transformações que nele se processem." A lei de Lavoisier também pode ser expressa na forma: "Em uma reação química, a soma das massas dos reagentes é igual à soma das massas dos produtos resultantes." Conclui-se, então, que os elementos têm que estar nos dois membros da equação em igual quantidade.

O método algébrico de balanceamento [19] consiste em atribuir coeficientes literais x_i às substâncias que aparecem na equação, os quais constituem as incógnitas. Aplicando a lei de Lavoisier e comparando os elementos membro a membro, constrói-se um sistema de equações algébricas lineares onde as incógnitas são os coeficientes estequiométricos x_i da reação química. Se houver mais incógnitas do que equações, atribui-se um valor arbitrário a uma delas

$$\begin{array}{l} \text{a: } \text{KMnO}_4 + x_2 \text{H}_2\text{SO}_4 + x_3 \text{NaNO}_2 \longrightarrow x_4 \text{K}_2\text{SO}_4 + x_5 \text{MnSO}_4 + x_6 \text{NaNO}_3 + x_7 \text{H}_2\text{O}, \\ \text{b: } \text{K: } x_1 = 2x_4, \\ \text{c: } \text{Mn: } x_1 = x_5, \\ \text{d: } \text{O: } 4x_1 + 4x_2 + 2x_3 = 4x_4 + 4x_5 + 3x_6 + x_7, \\ \text{e: } \text{H: } 2x_2 = 2x_7, \\ \text{f: } \text{S: } x_2 = x_4 + x_5, \\ \text{g: } \text{Na: } x_3 = x_6, \\ \text{h: } \text{N: } x_3 = x_6. \end{array}$$

Como as duas últimas expressões são iguais, elimina-se uma delas. Deste modo, tem-se um sistema linear com 6 equações e 7 incógnitas. Atribuindo um valor arbitrário a uma delas, por exemplo, $x_7 = 1$, obtém-se o seguinte sistema linear de ordem 6

$$\left[\begin{array}{cccccc} 1 & 0 & 0 & -2 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & 0 \\ 4 & 4 & 2 & -4 & -4 & -3 \\ 0 & 2 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \end{array} \right] \left[\begin{array}{c} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{array} \right] = \left[\begin{array}{c} 0 \\ 0 \\ 1 \\ 2 \\ 0 \\ 0 \end{array} \right].$$

Solução numérica

Resolvendo o sistema linear $Ax = b$, usando decomposição LU com pivotação parcial como no Exemplo 2.33, obtém-se os coeficientes estequiométricos x_i

% Os valores de entrada

```
n = 6
A =
 1   0   0   -2   0   0
 1   0   0   0   -1   0
 4   4   2   -4   -4   -3
 0   2   0   0   0   0
 0   1   0   -1   -1   0
 0   0   1   0   0   -1
```

```
% produzem os resultados pela decomposicao LU
A =
 4.0000  4.0000  2.0000 -4.0000 -4.0000 -3.0000
  0      2.0000    0      0      0      0
  0      0      1.0000    0      0     -1.0000
 0.2500 -0.5000 -0.5000  1.0000    0     0.2500
  0      0.5000    0     -1.0000 -1.0000  0.2500
 0.2500 -0.5000 -0.5000 -1.0000 -1.0000  0.7500
Det = 6
Pivot = 3   4   6   2   5   1
```

% vetor de termos independentes

b =

```
 0
 0
 1
 2
 0
 0
```

% As substituicoes sucessivas pivotal resultam em

```
y =
 1.0000
 2.0000
 0
 0.7500
 -0.2500
 1.2500
```

% As substituicoes sucessivas produzem a solucao

```
x =
 0.6667
 1.0000
 1.6667
 0.3333
 0.6667
 1.6667
```

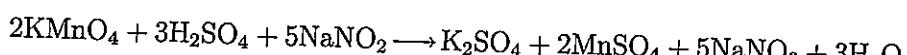
Os coeficientes estequiométricos são obtidos pela solução do sistema acrescido de $x_7 = 1 \rightarrow x = [0,6667 \ 1,0000 \ 1,6667 \ 0,3333 \ 0,6667 \ 1,6667 \ 1]^T$.

Análise dos resultados

Usualmente, os coeficientes estequiométricos são expressos como números inteiros. Pela regra de Cramer, para obter x com valores inteiros basta multiplicá-lo pelo determinante de A , no caso, $\det(A) = 6$. Para mais uma simplificação, divide-se o valor obtido por 2, resultando em

$$x = [2 \ 3 \ 5 \ 1 \ 2 \ 5 \ 3]^T.$$

Portanto, a equação química balanceada torna-se



2.11 Exercícios

Seção 2.1

2.1. Sejam as matrizes e vetores

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

$$x = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} \text{ e } y = \begin{bmatrix} 10 \\ 20 \\ 30 \end{bmatrix}.$$

Calcular

- a) $A + B$.
- b) Ax .
- c) AB .
- d) $x^T y$.
- e) $(xy)^T$.

2.2. Verificar que $(A + B)^T = A^T + B^T$ usando as matrizes A e B do Exercício 2.1.

2.3. Calcular os autovalores das matrizes dadas abaixo, verificar que

$$\text{traço}(A) = \sum_{i=1}^n \lambda_i \text{ e } \det(A) = \prod_{i=1}^n \lambda_i.$$

e observar que os autovalores $\lambda(A) = \lambda(A^T)$.

- a) $A = \begin{bmatrix} 5 & 2 \\ 1 & 4 \end{bmatrix}$.
- b) $A = \begin{bmatrix} 5 & 1 \\ 2 & 4 \end{bmatrix}$.
- c) $A = \begin{bmatrix} -1 & 5 \\ -2 & 5 \end{bmatrix}$.

2.4. Dado o vetor $x = [1 \ 2 \ 3 \ 4 \ 5]^T$, calcular as normas

- a) $\|x\|_1$.
- b) $\|x\|_2$.
- c) $\|x\|_\infty$.

2.5. Calcular as normas da matriz

$$A = \begin{bmatrix} 5 & 4 & -1 \\ 2 & 9 & 3 \\ 8 & -6 & 7 \end{bmatrix}.$$

- a) $\|A\|_1$.
- b) $\|A\|_\infty$.
- c) $\|A\|_F$.

Seção 2.2

2.6. Resolver o sistema triangular inferior por meio das substituições sucessivas

$$\begin{bmatrix} 4 & 0 & 0 \\ -2 & 5 & 0 \\ 1 & 7 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -2 \\ 5 \\ 6 \end{bmatrix}.$$

2.7. Calcular a solução do sistema triangular superior utilizando as substituições retroativas

$$\begin{bmatrix} 6 & -2 & 5 \\ 0 & 4 & -3 \\ 0 & 0 & 7 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 19 \\ 3 \\ -7 \end{bmatrix}.$$

2.8. Achar a solução do sistema triangular superior

$$\begin{bmatrix} 7 & 0 & -3 & 5 \\ 0 & -1 & 6 & 2 \\ 0 & 0 & 4 & -3 \\ 0 & 0 & 0 & 3 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} -9 \\ 12 \\ -3 \\ 5 \end{bmatrix}.$$

2.9. Implementar, em qualquer linguagem de programação, o algoritmo de substituições sucessivas mostrado na Figura 2.3.

2.10. Implementar, em uma linguagem de programação, o algoritmo de substituições retroativas apresentado na Figura 2.4.

Seção 2.3

Resolver os sistemas abaixo pelo método de eliminação de Gauss, com a estratégia indicada e usando 3 casas decimais; verificar também a unicidade e exatidão da solução.

2.11. Sem pivotação parcial

$$\begin{bmatrix} 1 & 2 & 4 \\ -3 & -1 & 4 \\ 2 & 14 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 13 \\ 8 \\ 50 \end{bmatrix}.$$

2.12. Com pivotação parcial

$$\begin{bmatrix} -2 & 3 & 1 \\ 2 & 1 & -4 \\ 4 & 10 & -6 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -5 \\ -9 \\ 2 \end{bmatrix}.$$

2.13. Sem e com pivotação parcial, comparando os resultados

$$\begin{bmatrix} 1 & -3 & 5 & 6 \\ -8 & 4 & -1 & 0 \\ 3 & 2 & -2 & 7 \\ 1 & 2 & 5 & -4 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 17 \\ 29 \\ -11 \\ 7 \end{bmatrix}.$$

2.14. Sem e com pivotação parcial, comparando os resultados

$$\begin{bmatrix} -2 & 3 & 1 & 5 \\ 5 & 1 & -1 & 0 \\ 1 & 6 & 3 & -1 \\ 4 & 5 & 2 & 8 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 2 \\ -1 \\ 0 \\ 6 \end{bmatrix}.$$

2.15. Com pivotação parcial

$$\begin{bmatrix} 0 & 1 & 3 & 2 & 4 \\ 8 & -2 & 9 & -1 & 2 \\ 5 & 1 & 1 & 7 & 2 \\ -2 & 4 & 5 & 1 & 0 \\ 7 & -3 & 2 & -4 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 3 \\ -5 \\ 6 \\ -1 \\ 8 \end{bmatrix}.$$

Seção 2.4

Resolver os sistemas a seguir pelo método da decomposição LU , com a estratégia indicada, e verificar a unicidade e exatidão da solução.

2.16. Efetuar os cálculos utilizando apenas 4 decimais

$$\begin{bmatrix} 2 & 6 & -3 \\ 1 & 3,001 & 2 \\ 4 & -1 & 9 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 5 \\ 9 \\ 29 \end{bmatrix}.$$

a) Sem pivotação.

b) Com pivotação parcial.

2.17. Com pivotação parcial

$$\begin{bmatrix} 1 & 2 & 3 \\ -5 & -1 & 4 \\ 2 & 4 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 17 \\ -2 \\ 24 \end{bmatrix}.$$

2.18. Com pivotação parcial

$$\begin{bmatrix} 4 & -1 & 3 & 8 \\ 1 & 6 & 2 & -3 \\ 5 & 5 & 1 & 0 \\ 2 & 4 & -2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 43 \\ 7 \\ 18 \\ 8 \end{bmatrix}.$$

2.19. Com pivotação parcial

$$\begin{bmatrix} 1 & -4 & -1 \\ -2 & 9 & 3 \\ 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 4 \end{bmatrix}.$$

2.20. Implementar, em uma linguagem de programação, os algoritmos de decomposição LU e de substituições sucessivas pivotal apresentados nas Figuras 2.5 e 2.6.

Seção 2.5

Calcular a solução dos sistemas a seguir utilizando a decomposição de Cholesky e verificar a exatidão e unicidade da solução.

$$\begin{bmatrix} 9 & -6 & 3 \\ -6 & 29 & -7 \\ 3 & -7 & 18 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -3 \\ -8 \\ 33 \end{bmatrix}.$$

$$\begin{bmatrix} 4 & -2 & 4 & 10 \\ -2 & 2 & -1 & -7 \\ 4 & -1 & 14 & 11 \\ 10 & -7 & 11 & 31 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 2 \\ 2 \\ -1 \\ -2 \end{bmatrix}.$$

$$\begin{bmatrix} 1 & 2 & -3 & 0 & 3 \\ 2 & 5 & -1 & 1 & 4 \\ -3 & -1 & 50 & 1 & -19 \\ 0 & 1 & 1 & 6 & 0 \\ 3 & 4 & -19 & 0 & 39 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 17 \\ 41 \\ -45 \\ 30 \\ 51 \end{bmatrix}.$$

2.24. Implementar o algoritmo de decomposição de Cholesky, mostrado na Figura 2.7, em qualquer linguagem de programação.**2.25.** Implementar o algoritmo de decomposição LDL^T , apresentado na Figura 2.8, utilizando qualquer linguagem de programação.

Seção 2.7

2.26. Formular um esquema para refinar a so-

lução de um sistema linear quando a matriz dos coeficientes for simétrica usando a decomposição de Cholesky.

2.27. Mostrar como calcular a inversa de uma matriz não simétrica utilizando a decomposição LU com pivotação parcial.**2.28.** Seja o sistema

$$\begin{bmatrix} 5 & -2 & 3 \\ -2 & 10 & 4 \\ 3 & 4 & 20 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 31 \\ -10 \\ 81 \end{bmatrix}.$$

Considerando que o fator L de Cholesky seja

$$\begin{bmatrix} 2,24 & 0 & 0 \\ -0,89 & 3,03 & 0 \\ 1,34 & 1,71 & 3,91 \end{bmatrix},$$

calcular e refinar o vetor solução até que a condição $\|c\|_\infty < 10^{-3}$ seja satisfeita.**2.29.** Calcular a inversa da matriz A pela decomposição LU com pivotação parcial, sendo

$$A = \begin{bmatrix} 1 & 6 & 4 \\ 2 & -3 & 1 \\ 5 & 5 & 8 \end{bmatrix}.$$

2.30. Mostrar como fazer o refinamento da inversa quando a matriz for simétrica e quando ela for não simétrica.

Seção 2.8

2.31. Resolver o sistema dado a seguir pelos métodos iterativos de Jacobi e de Gauss-Seidel com $\frac{\|x^k - x^{k-1}\|_\infty}{\|x^k\|_\infty} < 10^{-3}$ ou $k_{\max} = 10$

$$\begin{bmatrix} 10 & 2 & -3 & 5 \\ 1 & 8 & -1 & 2 \\ 2 & -1 & -5 & 1 \\ -1 & 2 & 3 & 20 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} 48 \\ 4 \\ -11 \\ 150 \end{bmatrix}.$$

2.32. Seja o sistema linear com a matriz dos coeficientes proposta por Collatz [9]

$$\begin{bmatrix} 1 & 2 & -2 \\ 1 & 1 & 1 \\ 2 & 2 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -1 \\ 6 \\ 9 \end{bmatrix}.$$

Verificar a convergência dos métodos de Jacobi e Gauss-Seidel e justificar os resultados.

2.33. Seja o sistema linear com a matriz dos coeficientes também proposta por Collatz

$$\begin{bmatrix} 1 & -0,5 & 0,5 \\ 1 & 1 & 1 \\ -0,5 & -0,5 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 3 \\ 12 \\ 3 \end{bmatrix}.$$

Verificar a convergência dos métodos de Jacobi e Gauss-Seidel e justificar os resultados.

2.34. Implementar o algoritmo do método de Jacobi mostrado na Figura 2.9 em qualquer linguagem de programação.**2.35.** Implementar, em qualquer linguagem de programação, o algoritmo do método de Gauss-Seidel apresentado na Figura 2.10.

Seção 2.9

2.36. Calcular $\kappa_1(H_2)$, $\kappa_2(H_2)$ e $\kappa_\infty(H_2)$ utilizando (2.42).**2.37.** Implementar o algoritmo da Figura 2.13 em qualquer linguagem de programação.**2.38.** Calcular H_4 e H_4^{-1} usando o algoritmo implementado no Exercício 2.37.**2.39.** Mostrar que $H_4 H_4^{-1} = I_4$.**2.40.** Calcular $\|H_4\|_\infty$, $\|H_4^{-1}\|_\infty$ e $\kappa_\infty(H_4)$ e comparar os resultados com aqueles apresentados na Tabela 2.10.

Gerais

2.41. Calcular o determinante da matriz de Vandermonde de ordem 3

$$M = \begin{bmatrix} 1 & x_0 & x_0^2 \\ 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \end{bmatrix}.$$

2.42. Mostrar que $\|A\|_2 = \max(\sqrt{\lambda(A^T A)})$ para a matriz do Exemplo 2.18.

2.43. Sejam uma matriz A de ordem n , um vetor arbitrário y^0 de tamanho n e os vetores $y^k = Ay^{k-1}$, $k = 1, 2, 3, \dots, n$. O método de Krylov [10] consiste em calcular os coeficientes d_i , $i = n-1, n-2, \dots, 0$ do polinômio característico $D(\lambda) = \lambda^n + d_{n-1}\lambda^{n-1} + d_{n-2}\lambda^{n-2} + \dots + d_1\lambda + d_0$ pela solução do sistema linear $Kd = -y^n$, sendo $K = [y^{n-1} \ y^{n-2} \ \dots \ y^0]$. Se a matriz K for singular, deve-se trocar o vetor arbitrário inicial y^0 . Determinar o polinômio característico das matrizes abaixo pelo método de Krylov

a) $A = \begin{bmatrix} 7 & 14 & -2 \\ -3 & -10 & 2 \\ -12 & -28 & 5 \end{bmatrix}$ (ver Exemplo 2.42),

b) $B = \begin{bmatrix} 5 & -1 & 3 \\ -1 & 4 & 2 \\ 3 & 2 & 1 \end{bmatrix}$.

2.44. Fazer um algoritmo para calcular os coeficientes do polinômio característico de uma matriz pelo método de Krylov.

2.45. Seja a matriz esparsa tridiagonal (todos os elementos fora da faixa compreendendo as três diagonais são nulos)

$$\begin{bmatrix} 4 & -2 & 0 & 0 & 0 \\ -2 & 10 & 6 & 0 & 0 \\ 0 & 6 & 20 & 12 & 0 \\ 0 & 0 & 12 & 34 & -5 \\ 0 & 0 & 0 & -5 & 2 \end{bmatrix}.$$

Fazer a decomposição de Cholesky e verificar que o fator L mantém o padrão de esparsidade de A .

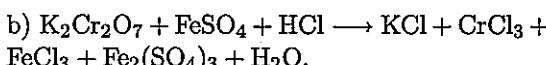
2.46. Resolver os Exercícios 2.21–2.23 utilizando o algoritmo da decomposição de Cholesky implementado no Exercício 2.24.

2.47. Resolver os Exercícios 2.21–2.23 utilizando o algoritmo da fatoração LDL^T implementado no Exercício 2.25.

2.48. Implementar em qualquer linguagem de programação o algoritmo do método SOR mostrado na Figura 2.11.

2.49. Resolver o sistema do Exercício 2.31 usando a implementação do algoritmo SOR, com diversos valores do parâmetro ω .

2.50. Equilibrar as reações químicas



Capítulo 3

Interpolação polinomial

A necessidade de obter um valor intermediário que não consta de uma tabela ocorre comumente. Dados experimentais, tabelas estatísticas e de funções complexas são exemplos desta situação. Neste capítulo, serão apresentados alguns métodos numéricos para resolver este problema de grande utilidade prática.

O conceito de interpolação não é importante somente na obtenção de valores intermediários em tabelas. Ele é fundamental em outros tópicos abordados em **Algoritmos Numéricos**, como, por exemplo, integração numérica (Capítulo 5), cálculo de raízes de equações (Capítulo 6) e solução de equações diferenciais ordinárias (Capítulo 7).

3.1 Polinômios interpoladores

Seja a Tabela 3.1 a seguir

Tabela 3.1 Dados para interpolação.

| | | | |
|-----|-------|-------|-------|
| x | 0,1 | 0,6 | 0,8 |
| y | 1,221 | 3,320 | 4,953 |

O problema consiste em encontrar o valor correspondente de y para um dado x não pertencente à tabela. Um modo de resolver este problema é obter uma função que relaciona as variáveis x e y . Considerando que os polinômios são as funções mais simples e estudadas, então eles são os mais utilizados para determinar esta relação. Um polinômio construído com o intuito de aproximar uma função é denominado polinômio interpolador. Deste modo, para resolver o problema basta avaliar o polinômio obtido no ponto desejado.

Existem vários métodos para construir um polinômio interpolador a partir de um conjunto de pares (x, y) . O esquema mais simples, em termos conceituais, envolve a solução de um sistema de equações lineares.

3.1.1 Interpolação linear

Sejam dois pontos-base (x_0, y_0) e (x_1, y_1) , com $x_0 \neq x_1$, de uma função $y = f(x)$. Para obter uma aproximação de $f(z)$, $z \in (x_0, x_1)$, faz-se

$$f(x) \approx P_1(x) = a_0 + a_1 x,$$

onde $P_1(x)$ é um polinômio interpolador de grau 1. Impondo que o polinômio interpolador passe pelos dois pontos-base, tem-se o seguinte sistema de equações lineares de ordem 2

$$\begin{array}{l} P_1(x_0) = y_0 \\ P_1(x_1) = y_1 \end{array} \longrightarrow \left\{ \begin{array}{l} a_0 + a_1 x_0 = y_0 \\ a_0 + a_1 x_1 = y_1 \end{array} \right. \iff \left[\begin{array}{cc} 1 & x_0 \\ 1 & x_1 \end{array} \right] \left[\begin{array}{c} a_0 \\ a_1 \end{array} \right] = \left[\begin{array}{c} y_0 \\ y_1 \end{array} \right].$$

Aplicando uma operação l-elementar para transformar este sistema linear em um sistema triangular equivalente, obtém-se

$$\begin{bmatrix} 1 & x_0 \\ 0 & x_1 - x_0 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} y_0 \\ y_1 - y_0 \end{bmatrix},$$

cuja solução é

$$a_1 = \frac{y_1 - y_0}{x_1 - x_0} \quad \text{e} \quad a_0 = y_0 - a_1 x_0$$

Portanto, o polinômio interpolador torna-se

$$P_1(x) = a_0 + a_1 x = (y_0 - a_1 x_0) + a_1 x = y_0 + a_1(x - x_0) \text{ e}$$

$$P_1(x) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0). \quad (3.1)$$

Como o determinante da matriz do sistema linear acima é igual a $x_1 - x_0 \neq 0$, então o sistema admite uma única solução, ou seja, por dois pontos passa um único polinômio de grau 1. Deve ser verificado que, por (3.1), $P_1(x_0) = y_0$ e $P_1(x_1) = y_1$, conforme imposto ao se construir o sistema linear. Isso mostra que, de fato, o polinômio interpolador passa pelos dois pontos-base.

Exemplo 3.1 Calcular $P_1(0,2)$ e $P_1(0,3)$ a partir da tabela

| | | |
|-------|-------|-------|
| i | 0 | 1 |
| x_i | 0,1 | 0,6 |
| y_i | 1,221 | 3,320 |

Pelo uso de (3.1),

$$P_1(0,2) = 1,221 + \frac{3,320 - 1,221}{0.6 - 0.1}(0,2 - 0,1) \rightsquigarrow P_1(0,2) = 1,641 \text{ e}.$$

$$P_1(0,3) = 1,221 + \frac{3,320 - 1,221}{0.6 - 0.1}(0.3 - 0.1) \rightsquigarrow P_1(0,3) = 2,061.$$

Sabendo que a função neste exemplo é $f(x) = e^{2x}$, então, os erros cometidos foram

em $x \equiv 0,2$ tem-se $1,641 - e^{2 \times 0,2} = 0,149$ e

em $x \equiv 0.3$ tem-se $2.061 - e^{2 \times 0.3} = 0.239$.

Quanto mais próximo o valor a ser interpolado for de um ponto-base, melhor será o resultado obtido pela interpolação. O resultado da interpolação pode ser melhorado pelo aumento do grau do polinômio interpolador.

A Figura 3.1 mostra os dados da Tabela 3.1 representados por pontos \circ e o polinômio interpolador de grau 1 esboçado por uma linha tracejada $--$. São apresentados também o polinômio interpolador de grau 2, desenhado com uma linha traço-ponto $-\cdot-$, e a função $y(x) = e^{2x}$, representada por uma linha sólida — .

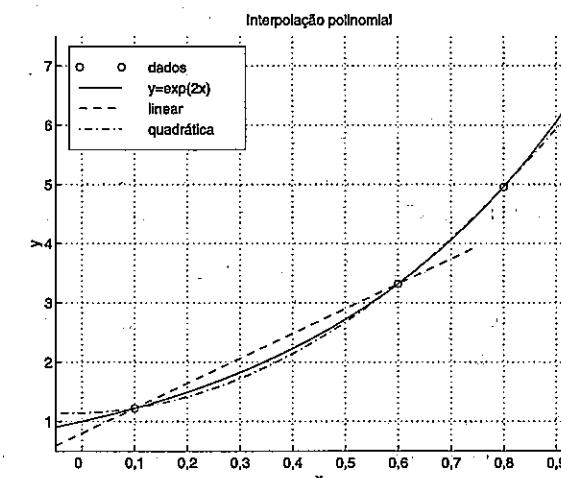


Figura 3.1 Interpretação geométrica da interpolação polinomial.

3.1.2 Interpolacao quadratica

Sejam três pontos-base (x_0, y_0) , (x_1, y_1) e (x_2, y_2) , com x_i distintos, de uma função $y = f(x)$. Para aproximar $f(z)$, $z \in (x_0, x_2)$, faz-se

$$f(x) \approx P_2(x) = a_0 + a_1x + a_2x^2$$

sendo $P_2(x)$ um polinômio interpolador de grau 2. Impondo que o polinômio interpolador passe pelos três pontos-base, tem-se o sistema linear de ordem 3

$$\begin{array}{l} P_2(x_0) = y_0 \\ P_2(x_1) = y_1 \\ P_2(x_2) = y_2 \end{array} \rightarrow \left\{ \begin{array}{l} a_0 + a_1 x_0 + a_2 x_0^2 = y_0 \\ a_0 + a_1 x_1 + a_2 x_1^2 = y_1 \\ a_0 + a_1 x_2 + a_2 x_2^2 = y_2 \end{array} \right. \iff \left[\begin{array}{ccc|c} 1 & x_0 & x_0^2 & a_0 \\ 1 & x_1 & x_1^2 & a_1 \\ 1 & x_2 & x_2^2 & a_2 \end{array} \right] \left[\begin{array}{c} y_0 \\ y_1 \\ y_2 \end{array} \right].$$

A matriz dos coeficientes é a matriz de Vandermonde, e o sistema de equações lineares admite uma única solução, pois $\det(X) = (x_2 - x_0)(x_2 - x_1)(x_1 - x_0) \neq 0$. Consequentemente, por três pontos passa um único polinômio de grau 2. Este fato pode ser generalizado, dizendo-se que por $n + 1$ pontos passa um único polinômio de grau n .

Exemplo 3.2 Calcular $P_2(0,2)$ usando os dados da Tabela 3.1.

Os coeficientes do polinômio interpolador são determinados pela solução do sistema linear

$$\begin{bmatrix} 1 & 0,1 & 0,01 \\ 1 & 0,6 & 0,36 \\ 1 & 0,8 & 0,64 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 1,221 \\ 3,320 \\ 4,953 \end{bmatrix}.$$

Usando a decomposição LU com pivotação parcial, vista na Seção 2.4, têm-se os fatores

$$L = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0,714 & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & 0,1 & 0,01 \\ 0 & 0,7 & 0,63 \\ 0 & 0 & -0,1 \end{bmatrix} \quad \text{e} \quad P = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}.$$

O sistema triangular inferior $Lt = Py$ é resolvido pelas substituições sucessivas (2.7), fornecendo $t = [1,221 \ 3,732 \ -0,567]^T$. Os coeficientes a_i do polinômio interpolador são obtidos pela solução do sistema triangular superior $Ua = t$ usando as substituições retroativas (2.8), o que fornece $a = [1,141 \ 0,231 \ 5,667]^T$. Conseqüentemente, o polinômio tem a forma

$$P_2(x) = 1,141 + 0,231x + 5,667x^2 \rightsquigarrow P_2(0,2) = 1,414.$$

Como o polinômio passa pelos pontos-base, pode ser constatado que $P_2(0,1) = 1,221$, $P_2(0,6) = 3,320$ e $P_2(0,8) = 4,953$.

Como visto, esta metodologia de determinar os coeficientes do polinômio interpolador por meio da resolução de um sistema de equações lineares, apesar de ser conceitualmente simples, requer um certo esforço computacional, pois a decomposição LU tem complexidade da ordem de n^3 , de acordo com a Tabela 2.4. Considerando que por $n+1$ pontos passa um único polinômio de grau n , deve ser procurada uma metodologia alternativa de modo a evitar a solução de um sistema de equações lineares. Serão vistas, a seguir, outras formas de obter um polinômio interpolador, as quais apresentam uma menor ordem de complexidade.

3.2 Polinômios de Lagrange

Sejam dados $n+1$ pontos $(x_0, y_0), (x_1, y_1), \dots, (x_n, y_n)$, sendo x_i distintos, tais que $y_i = f(x_i)$ e $x \in (x_0, x_n)$. Deseja-se construir um polinômio $L_n(x)$ de grau não superior a n e possuindo nos pontos x_i o mesmo valor da função $f(x)$, isto é,

$$L_n(x_i) = y_i, \quad i = 0, 1, 2, \dots, n. \quad (3.2)$$

3.2.1 Fórmula de Lagrange

Inicialmente, sejam os polinômios de grau n , $P_i(x)$, $i = 0, 1, 2, \dots, n$, tais que

$$P_i(x_i) \neq 0 \quad \text{e} \quad P_i(x_j) = 0, \quad \forall i \neq j.$$

Assim,

$$P_0(x) = (x - x_1)(x - x_2)(x - x_3) \dots (x - x_n),$$

$$P_1(x) = (x - x_0)(x - x_2)(x - x_3) \dots (x - x_n),$$

$$P_2(x) = (x - x_0)(x - x_1)(x - x_3) \dots (x - x_n),$$

\vdots

$$P_n(x) = (x - x_0)(x - x_1)(x - x_2) \dots (x - x_{n-1}),$$

(ou seja,

$$P_i(x) = \prod_{\substack{j=0 \\ j \neq i}}^n (x - x_j), \quad i = 0, 1, 2, \dots, n. \quad (3.3)$$

Considerando que $L_n(x)$ é de grau não superior a n , ele pode ser escrito como uma combinação linear dos polinômios $P_i(x)$, isto é,

$$L_n(x) = c_0 P_0(x) + c_1 P_1(x) + c_2 P_2(x) + \dots + c_n P_n(x) \\ L_n(x) = \sum_{i=0}^n c_i P_i(x). \quad (3.4)$$

Comparando (3.2) e (3.4) e em vista de que $P_i(x_j) = 0, \forall i \neq j$, tem-se

$$L_n(x_i) = y_i = c_i P_i(x_i) \longrightarrow c_i = \frac{y_i}{P_i(x_i)}.$$

Substituindo o valor de c_i em (3.4), tem-se

$$L_n(x) = \sum_{i=0}^n \frac{y_i}{P_i(x_i)} P_i(x).$$

Em vista de (3.3), tem-se a fórmula do polinômio interpolador de Lagrange de grau n

$$L_n(x) = \sum_{i=0}^n y_i \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j}. \quad (3.5)$$

Exemplo 3.3 Calcular $L_1(0,2)$ a partir dos dados do Exemplo 3.1

| | | |
|-------|-------|-------|
| i | 0 | 1 |
| x_i | 0,1 | 0,6 |
| y_i | 1,221 | 3,320 |

Usando (3.5) com $n = 1$, tem-se

$$L_1(x) = y_0 \frac{x - x_1}{x_0 - x_1} + y_1 \frac{x - x_0}{x_1 - x_0},$$

$$L_1(0,2) = 1,221 \frac{0,2 - 0,6}{0,1 - 0,6} + 3,320 \frac{0,2 - 0,1}{0,6 - 0,1} \rightsquigarrow L_1(0,2) = 1,641.$$

Este resultado é igual ao obtido no Exemplo 3.1 usando a fórmula de interpolação linear (3.1). De fato, (3.5) com $n = 1$ é idêntica a (3.1)

$$\begin{aligned} P_1(x) &= y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) = \frac{y_0x_1 - y_0x_0 + y_1x - y_1x_0 - y_0x + y_0x_0}{x_1 - x_0}, \\ P_1(x) &= \frac{y_0(x_1 - x) + y_1(x - x_0)}{x_1 - x_0} = y_0 \frac{x - x_1}{x_0 - x_1} + y_1 \frac{x - x_0}{x_1 - x_0} \rightsquigarrow P_1(x) = L_1(x). \end{aligned}$$

Exemplo 3.4 Calcular $L_2(0,2)$ usando os dados da tabela do Exemplo 3.2

| i | 0 | 1 | 2 |
|-------|-------|-------|-------|
| x_i | 0,1 | 0,6 | 0,8 |
| y_i | 1,221 | 3,320 | 4,953 |

A fórmula de Lagrange (3.5) com $n = 2$ torna-se

$$\begin{aligned} L_2(x) &= y_0 \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + y_1 \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} + y_2 \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}, \\ L_2(0,2) &= 1,221 \frac{(0,2-0,6)(0,2-0,8)}{(0,1-0,6)(0,1-0,8)} + 3,320 \frac{(0,2-0,1)(0,2-0,8)}{(0,6-0,1)(0,6-0,8)} + \\ &4,953 \frac{(0,2-0,1)(0,2-0,6)}{(0,8-0,1)(0,8-0,6)} \rightsquigarrow L_2(0,2) = 1,414. \end{aligned}$$

Considerando que $f(0,2) = e^{2 \times 0,2} \approx 1,492$, o erro cometido foi menor do que quando $L_1(0,2)$ foi utilizado. Portanto, quando o grau do polinômio interpolador é aumentado, a exatidão do resultado é melhorada. Obviamente, a interpolação de Lagrange requer um menor esforço computacional do que resolver um sistema de equações lineares, como será mostrado mais adiante.

3.2.2 Dispositivo prático

Um dispositivo prático pode ser construído para facilitar o uso dos polinômios de Lagrange. Para tal, seja a matriz

$$G = \begin{bmatrix} x - x_0 & x_0 - x_1 & x_0 - x_2 & \cdots & x_0 - x_n \\ x_1 - x_0 & x - x_1 & x_1 - x_2 & \cdots & x_1 - x_n \\ x_2 - x_0 & x_2 - x_1 & x - x_2 & \cdots & x_2 - x_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_n - x_0 & x_n - x_1 & x_n - x_2 & \cdots & x - x_n \end{bmatrix}.$$

Acrescentando o termo $(x - x_i)/(x - x_i)$ na fração de (3.5), esta não se modificará numericamente

$$\text{Se } L_n(x) = \sum_{i=0}^n y_i \prod_{\substack{j=0 \\ j \neq i}}^n \frac{x - x_j}{x_i - x_j} \times \frac{x - x_i}{x - x_i} \rightarrow L_n(x) = G_d \sum_{i=0}^n \frac{y_i}{G_i}, \quad (3.6)$$

onde G_d é o produto dos elementos da diagonal principal da matriz G e G_i é o produto dos elementos da $(i+1)$ -ésima linha de G .

Exemplo 3.5 Determinar $L_2(0,2)$ por (3.6) usando os dados do Exemplo 3.4.

A matriz G é

$$(b) \quad G = \begin{bmatrix} 0,2 - 0,1 & 0,1 - 0,6 & 0,1 - 0,8 \\ 0,6 - 0,1 & 0,2 - 0,6 & 0,6 - 0,8 \\ 0,8 - 0,1 & 0,8 - 0,6 & 0,2 - 0,8 \end{bmatrix} = \begin{bmatrix} 0,1 & -0,5 & -0,7 \\ 0,5 & -0,4 & -0,2 \\ 0,7 & 0,2 & -0,6 \end{bmatrix}.$$

Conseqüentemente,

$$G_d = (0,1)(-0,4)(-0,6) = 0,024,$$

$$G_0 = (0,1)(-0,5)(-0,7) = 0,035,$$

$$G_1 = (0,5)(-0,4)(-0,2) = 0,040 \text{ e}$$

$$G_2 = (0,7)(0,2)(-0,6) = -0,084.$$

Usando (3.6) com $n = 2$, tem-se

$$L_2(x) = G_d \left(\frac{y_0}{G_0} + \frac{y_1}{G_1} + \frac{y_2}{G_2} \right),$$

$$L_2(0,2) = 0,024 \left(\frac{1,221}{0,035} + \frac{3,320}{0,040} + \frac{4,953}{-0,084} \right) \rightsquigarrow L_2(0,2) = 1,414.$$

3.2.3 Algoritmo e complexidade computacional

Um algoritmo para interpolação, usando os polinômios de Lagrange, é mostrado na Figura 3.2. Os parâmetros de entrada são o número m de pontos, o vetor x contendo as m abscissas x_i , o vetor y com as m ordenadas y_i e o ponto z a ser interpolado. O parâmetro de saída é a ordenada r do polinômio de Lagrange de grau $m-1$ avaliado no ponto z .

Neste algoritmo, (3.5) é implementada de forma a reduzir o número de operações de divisão, se comparado com o algoritmo da Figura 1.9, na página 18.

```

Algoritmo Polinômio Lagrange
{ Objetivo: Interpolar valor em tabela usando polinômio de Lagrange }
parâmetros de entrada m, x, y, z
{ número de pontos, abscissas, ordenadas e valor a interpolar }
parâmetros de saída r { valor interpolado }

r ← 0
para i ← 1 até m faça
  c ← 1; d ← 1
  para j ← 1 até m faça
    se i ≠ j então
      c ← c * (z - x(j)); d ← d * (x(i) - x(j))
    fimse
  fimpara
  r ← r + y(j) * c/d
fimpara
finalgoritmo

```

Figura 3.2 Interpolação por polinômio de Lagrange.

A complexidade computacional do algoritmo da Figura 3.2 é mostrada na Tabela 3.2, a qual deve ser comparada com aquela apresentada na Tabela 1.6, na página 19.

Tabela 3.2 Complexidade da interpolação de Lagrange.

(n: grau do polinômio interpolador.)

| Operações | Complexidade |
|----------------|-----------------|
| adições | $2n^2 + 3n + 1$ |
| multiplicações | $2n^2 + 3n + 1$ |
| divisões | $n + 1$ |

Exemplo 3.6 Resolver o problema do Exemplo 3.4 utilizando uma implementação do algoritmo da Figura 3.2.

```

% Os parâmetros de entrada
m = 3
x = 0.1000  0.6000  0.8000
y = 1.2210  3.3200  4.9530
z = 0.2000
% fornecem o resultado
r = 1.4141

```

3.3 Polinômios de Newton

Como visto na seção anterior, os polinômios de Lagrange constituem um modo de interpolar sem a necessidade de resolver um sistema de equações lineares. Será mostrado, a seguir, um esquema alternativo para a construção de um polinômio interpolador.

3.3.1 Operador de diferença dividida

Seja a função $y = f(x)$, que passa pelos pontos (x_i, y_i) , $i = 0, 1, 2, \dots, n$. O operador de diferença dividida Δ^i de ordem i é definido como

a) ordem 0: $\Delta^0 y_i = y_i = [x_i]$,

b) ordem 1: $\Delta y_i = \frac{\Delta^0 y_{i+1} - \Delta^0 y_i}{x_{i+1} - x_i} = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} = [x_i, x_{i+1}]$,

c) ordem 2: $\Delta^2 y_i = \frac{\Delta y_{i+1} - \Delta y_i}{x_{i+2} - x_i} = [x_i, x_{i+1}, x_{i+2}]$,

d) ordem n : $\Delta^n y_i = \frac{\Delta^{n-1} y_{i+1} - \Delta^{n-1} y_i}{x_{i+n} - x_i} = [x_i, x_{i+1}, \dots, x_{i+n}]$.

O Teorema 3.1 [3, Teorema 6.9] mostra uma importante propriedade das diferenças divididas de um polinômio. Essa propriedade é fundamental para a construção dos polinômios de Newton.

Teorema 3.1 Se $y = f(x)$ for um polinômio de grau n , então suas diferenças divididas de ordem $n + 1$ são identicamente nulas, isto é,

$$[x, x_0, x_1, \dots, x_n] = 0 \quad \forall x.$$

Exemplo 3.7 Verificar a tabela de diferenças divididas do polinômio $y = 5x^3 - 2x^2 - x + 3$ para alguns pontos x_i no intervalo $[0; 0,9]$

| i | x_i | y_i | Δy_i | $\Delta^2 y_i$ | $\Delta^3 y_i$ | $\Delta^4 y_i$ |
|---|-------|-------|--------------|----------------|----------------|----------------|
| 0 | 0,0 | 3,000 | -1,20 | 0,5 | 5 | 0 |
| 1 | 0,2 | 2,760 | -1,05 | 2,5 | 5 | 0 |
| 2 | 0,3 | 2,655 | -0,55 | 5,0 | 5 | |
| 3 | 0,4 | 2,600 | 1,45 | 8,0 | | |
| 4 | 0,7 | 3,035 | 5,45 | | | |
| 5 | 0,9 | 4,125 | | | | |

3.3.2 Fórmula de Newton

Sejam os $n+1$ pontos (x_i, y_i) , $i = 0, 1, 2, \dots, n$, com x_i distintos, tais que $y_i = P(x_i)$, sendo $P(x)$ um polinômio de grau n . A diferença dividida de ordem 1 é

$$[x, x_0] = \frac{P(x) - P(x_0)}{x - x_0}$$

ou

$$P(x) = P(x_0) + [x, x_0](x - x_0). \quad (3.7)$$

No entanto, a diferença dividida de ordem 2

$$[x, x_0, x_1] = \frac{[x, x_0] - [x_0, x_1]}{x - x_1} \rightsquigarrow [x, x_0] = [x_0, x_1] + [x, x_0, x_1](x - x_1).$$

Substituindo esta equação em (3.7), tem-se

$$P(x) = P(x_0) + [x_0, x_1](x - x_0) + [x, x_0, x_1](x - x_0)(x - x_1). \quad (3.8)$$

Contudo, a diferença dividida de ordem 3

$$\begin{aligned} [x, x_0, x_1, x_2] &= \frac{[x, x_0, x_1] - [x_0, x_1, x_2]}{x - x_2} \rightsquigarrow \\ [x, x_0, x_1] &= [x_0, x_1, x_2] + [x, x_0, x_1, x_2](x - x_2). \end{aligned}$$

Substituindo a equação acima em (3.8), obtém-se

$$\begin{aligned} P(x) &= P(x_0) + [x_0, x_1](x - x_0) + [x_0, x_1, x_2](x - x_0)(x - x_1) + \\ &\quad + [x, x_0, x_1, x_2](x - x_0)(x - x_1)(x - x_2). \end{aligned} \quad (3.9)$$

Continuando o desenvolvimento de $[x, x_0, x_1, x_2]$, chega-se a

$$\begin{aligned} P(x) &= P(x_0) + [x_0, x_1](x - x_0) + [x_0, x_1, x_2](x - x_0)(x - x_1) + \\ &\quad + [x_0, x_1, x_2, x_3](x - x_0)(x - x_1)(x - x_2) + \dots + \\ &\quad + [x_0, x_1, \dots, x_n](x - x_0)(x - x_1) \dots (x - x_{n-1}) + \\ &\quad + [x, x_0, x_1, \dots, x_n](x - x_0)(x - x_1) \dots (x - x_n). \end{aligned}$$

Considerando que $P(x)$ é um polinômio de grau n , então pelo Teorema 3.1 resulta que

$$[x, x_0, x_1, \dots, x_n] = 0.$$

Fazendo $y_0 = P(x_0)$ e usando a notação de diferenças divididas com o operador Δ , tem-se o polinômio de Newton de grau n

$$P_n(x) = y_0 + \Delta y_0(x - x_0) + \Delta^2 y_0(x - x_0)(x - x_1) + \dots + \Delta^n y_0(x - x_0) \dots (x - x_{n-1})$$

$$P_n(x) = y_0 + \sum_{i=1}^n \Delta^i y_0 \prod_{j=0}^{i-1} (x - x_j). \quad (3.10)$$

Uma vantagem dos polinômios de Newton sobre os de Lagrange é que para aumentar o grau basta acrescentar um novo termo em vez de montar todo o polinômio novamente.

Exemplo 3.8 Calcular $P_1(0,2)$ a partir dos dados

| | | |
|-----|-------|-------|
| x | 0,1 | 0,6 |
| y | 1,221 | 3,320 |

Pelo uso de (3.10) com $n = 1$, tem-se que

$$P_1(x) = y_0 + \Delta y_0(x - x_0).$$

A tabela de diferenças divididas é

| i | x_i | y_i | Δy_i |
|-----|-------|-------|--------------|
| 0 | 0,1 | 1,221 | 4,198 |
| 1 | 0,6 | 3,320 | |

$$P_1(0,2) = 1,221 + 4,198(0,2 - 0,1) \rightsquigarrow P_1(0,2) = 1,641.$$

Deve ser observado que este resultado é idêntico aos obtidos nos Exemplos 3.1 e 3.3. É fácil verificar que

$$P_1(x) = y_0 + \Delta y_0(x - x_0) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) \quad (\text{comparar com (3.1)}).$$

Exemplo 3.9 Determinar $P_2(1,2)$ usando os dados

| | | | |
|-----|-------|-------|-------|
| x | 0,9 | 1,1 | 2,0 |
| y | 3,211 | 2,809 | 1,614 |

Por (3.10) com $n = 2$, tem-se que

$$P_2(x) = y_0 + \Delta y_0(x - x_0) + \Delta^2 y_0(x - x_0)(x - x_1).$$

A tabela de diferenças divididas é

| i | x_i | y_i | Δy_i | $\Delta^2 y_i$ |
|-----|-------|-------|--------------|----------------|
| 0 | 0,9 | 3,211 | -2,010 | 0,620 |
| 1 | 1,1 | 2,809 | -1,328 | |
| 2 | 2,0 | 1,614 | | |

$$\begin{aligned}P_2(1,2) &= 3,211 + (-2,010)(1,2 - 0,9) + (0,620)(1,2 - 0,9)(1,2 - 1,1) \sim \\P_2(1,2) &= 2,627.\end{aligned}$$

Exemplo 3.10 Calcular $P_4(0,2)$ a partir de

| | | | | | |
|-----|--------|--------|--------|--------|--------|
| x | 0,1 | 0,3 | 0,4 | 0,6 | 0,7 |
| y | 0,3162 | 0,5477 | 0,6325 | 0,7746 | 0,8367 |

Por (3.10) com $n = 4$, tem-se

$$\begin{aligned}P_4(x) &= y_0 + \Delta y_0(x - x_0) + \Delta^2 y_0(x - x_0)(x - x_1) + \\&\quad \Delta^3 y_0(x - x_0)(x - x_1)(x - x_2) + \Delta^4 y_0(x - x_0)(x - x_1)(x - x_2)(x - x_3).\end{aligned}$$

A tabela de diferenças divididas é

| i | x_i | y_i | Δy_i | $\Delta^2 y_i$ | $\Delta^3 y_i$ | $\Delta^4 y_i$ |
|-----|-------|--------|--------------|----------------|----------------|----------------|
| 0 | 0,1 | 0,3162 | 1,1575 | -1,0317 | 1,1468 | -1,2447 |
| 1 | 0,3 | 0,5477 | 0,8480 | -0,4583 | 0,4000 | |
| 2 | 0,4 | 0,6325 | 0,7105 | -0,2983 | | |
| 3 | 0,6 | 0,7746 | 0,6210 | | | |
| 4 | 0,7 | 0,8367 | | | | |

$$\begin{aligned}P_4(0,2) &= 0,3162 + 1,1575(0,1) + (-1,0317)(0,1)(-0,1) + \\&\quad 1,1468(0,1)(-0,1)(-0,2) + (-1,2447)(0,1)(-0,1)(-0,2)(-0,4) \sim\end{aligned}$$

$$P_4(0,2) = 0,4456.$$

3.3.3 Algoritmo e complexidade computacional

A Figura 3.3 exibe um algoritmo para interpolar um valor z em uma tabela definida pelos vetores x e y usando um polinômio de diferenças divididas de grau n . Para tal, o polinômio (3.10)

$$P_n(x) = y_0 + \sum_{i=1}^n \Delta^i y_0 \prod_{j=0}^{i-1} (x - x_j)$$

é escrito de forma a ser avaliado pelo processo de Horner (ver Figura 6.2, na página 273)

$$P_n(z) = (\dots(\Delta^n y_0(z - x_{n-1}) + \Delta^{n-1} y_0(z - x_{n-2}) + \dots + \Delta^2 y_0(z - x_1) + \Delta y_0(z - x_0) + y_0).$$

Os parâmetros de entrada do algoritmo são o número m de pontos, o vetor x contendo as m abscissas x_i , o vetor y com as m ordenadas y_i e o ponto z a ser interpolado. O parâmetro de saída é o valor r do polinômio de grau $m - 1$ avaliado no ponto z .

Algoritmo Polinômio Newton

```
{ Objetivo: Interpolar valor em tabela usando polinômio de Newton }
parâmetros de entrada m, x, y, z
{ número de pontos, abscissas, ordenadas e valor a interpolar }
parâmetros de saída r { valor interpolado }
para i ← 1 até m faça
  Dely(i) ← y(i)
fimpara
{ construção das diferenças divididas }
para k ← 1 até m - 1 faça
  para i ← m até k + 1 passo -1 faça
    Dely(i) ← ((Dely(i) - Dely(i - 1)) / (x(i) - x(i - k)))
  fimpara
fimpara
{ avaliação do polinômio pelo processo de Horner }
r ← Dely(m)
para i ← m - 1 até 1 passo -1 faça
  r ← r * (z - x(i)) + Dely(i)
fimpara
fimalgoritmo
```

Figura 3.3 Interpolação por polinômio de Newton com diferenças divididas.

Este algoritmo utiliza um vetor auxiliar $Dely$, no qual é construída a tabela de diferenças divididas. Para economizar espaço de memória, os valores de $\Delta^0 y_0$, Δy_0 , ..., $\Delta^n y_0$ são armazenados nas primeiras $n + 1$ posições de $Dely$, conforme o esquema para cinco pontos

| i | x_i | y_i | $Dely_i^{(1)}$ | $Dely_i^{(2)}$ | $Dely_i^{(3)}$ | $Dely_i^{(4)}$ |
|-----|-------|-------|----------------|----------------|----------------|----------------|
| 1 | x_0 | y_0 | $\Delta^0 y_0$ | $\Delta^0 y_0$ | $\Delta^0 y_0$ | $\Delta^0 y_0$ |
| 2 | x_1 | y_1 | Δy_0 | Δy_0 | Δy_0 | Δy_0 |
| 3 | x_2 | y_2 | Δy_1 | $\Delta^2 y_0$ | $\Delta^2 y_0$ | $\Delta^2 y_0$ |
| 4 | x_3 | y_3 | Δy_2 | $\Delta^2 y_1$ | $\Delta^3 y_0$ | $\Delta^3 y_0$ |
| 5 | x_4 | y_4 | Δy_3 | $\Delta^2 y_2$ | $\Delta^3 y_1$ | $\Delta^4 y_0$ |

onde $Dely_i^{(k)}$ significa i -ésima posição do vetor $Dely$ na k -ésima repetição do comando para $k \leftarrow 1$ até $m - 1$ faça. A complexidade computacional do algoritmo da Figura 3.3 é mostrada na Tabela 3.3.

Exemplo 3.11 Calcular $P_4(0,2)$ a partir dos dados da tabela do Exemplo 3.10 usando o algoritmo da Figura 3.3.

Tabela 3.3 Complexidade da interpolação de Newton.

(n: grau do polinômio interpolador.)

| Operações | Complexidade |
|----------------|---------------------------------|
| adições | $n^2 + 4n$ |
| multiplicações | n |
| divisões | $\frac{1}{2}n^2 + \frac{1}{2}n$ |

% Os parametros de entrada

m = 5

x = 0.1000 0.3000 0.4000 0.6000 0.7000
y = 0.3162 0.5477 0.6325 0.7746 0.8367

z = 0.2000

% produzem o resultado

r = 0.4456

A seqüência de vetores $Dely$ produzida pelo algoritmo é

| i | x_i | y_i | $Dely_i^{(1)}$ | $Dely_i^{(2)}$ | $Dely_i^{(3)}$ | $Dely_i^{(4)}$ |
|---|-------|--------|----------------|----------------|----------------|----------------|
| 1 | 0,1 | 0,3162 | 0,3162 | 0,3162 | 0,3162 | 0,3162 |
| 2 | 0,3 | 0,5477 | 1,1575 | 1,1575 | 1,1575 | 1,1575 |
| 3 | 0,4 | 0,6325 | 0,8480 | -1,0317 | -1,0317 | -1,0317 |
| 4 | 0,6 | 0,7746 | 0,7105 | -0,4583 | 1,1468 | 1,1468 |
| 5 | 0,7 | 0,8367 | 0,6210 | -0,2983 | 0,4000 | -1,2447 |

Exemplo 3.12 Calcular $P_1(0,2)$, $P_2(0,2)$ e $P_3(0,2)$ usando os dados da tabela do Exemplo 3.10 e o algoritmo da Figura 3.3.

```
% Calculo de P1(0,2)
m = 2
x = 0.1000 0.3000
y = 0.3162 0.5477
z = 0.2000
r = 0.4320
% Calculo de P2(0,2)
m = 3
x = 0.1000 0.3000 0.4000
y = 0.3162 0.5477 0.6325
z = 0.2000
r = 0.4423
% Calculo de P3(0,2)
m = 4
x = 0.1000 0.3000 0.4000 0.6000
y = 0.3162 0.5477 0.6325 0.7746
z = 0.2000
r = 0.4446
```

Considerando que $y = \sqrt{x}$, portanto, o valor exato é $\sqrt{0,2} \approx 0,4472$. A partir dos resultados acima e mais o valor de $P_4(0,2)$ obtido no Exemplo 3.11, pode-se verificar que a diferença entre o valor interpolado e o exato diminui à medida que o grau do polinômio interpolador aumenta

| n | $P_n(0,2)$ | $ P_n(0,2) - \sqrt{0,2} $ |
|---|------------|---------------------------|
| 1 | 0,4320 | 0,0152 |
| 2 | 0,4423 | 0,0049 |
| 3 | 0,4446 | 0,0026 |
| 4 | 0,4456 | 0,0016 |

3.4 Polinômios de Gregory-Newton

Quando os valores das abscissas x_i forem igualmente espaçados, a fórmula de Newton pode ser simplificada, resultando na fórmula de Gregory-Newton. Portanto, o polinômio de Gregory-Newton é um caso particular do polinômio de Newton para pontos igualmente espaçados.

3.4.1 Operador de diferença finita ascendente

Seja a função $y = f(x)$ que passa pelos pontos (x_i, y_i) , $i = 0, 1, 2, \dots, n$, sendo $x_{i+1} - x_i = h \forall i$. O operador de diferença finita ascendente Δ^i de ordem i é definido como

a) ordem 0: $\Delta^0 y_i = y_i$,b) ordem 1: $\Delta y_i = \Delta^0 y_{i+1} - \Delta^0 y_i = y_{i+1} - y_i$,c) ordem 2: $\Delta^2 y_i = \Delta y_{i+1} - \Delta y_i$,d) ordem n : $\Delta^n y_i = \Delta^{n-1} y_{i+1} - \Delta^{n-1} y_i$.

Exemplo 3.13 Verificar a tabela de diferenças finitas

| i | x_i | y_i | Δy_i | $\Delta^2 y_i$ | $\Delta^3 y_i$ | $\Delta^4 y_i$ |
|---|-------|-------|--------------|----------------|----------------|----------------|
| 0 | 3,5 | 9,82 | 1,09 | 0,05 | -0,10 | 2,11 |
| 1 | 4,0 | 10,91 | 1,14 | -0,05 | 2,01 | |
| 2 | 4,5 | 12,05 | 1,09 | 1,96 | | |
| 3 | 5,0 | 13,14 | 3,05 | | | |
| 4 | 5,5 | 16,19 | | | | |

A relação entre os operadores de diferença finita é dividida é dada pela expressão

$$\Delta^n y_i = \frac{\Delta^n y_i}{n! h^n} \quad (3.11)$$

Exemplo 3.14 Para a tabela do Exemplo 3.13,

$$\Delta y_0 = \frac{\Delta y_0}{1!h} \sim \frac{10,91 - 9,82}{4,0 - 3,5} = \frac{1,09}{1!0,5} = 2,18 \text{ e}$$

$$\Delta^2 y_1 = \frac{\Delta^2 y_1}{2!h^2} \sim$$

$$\frac{\Delta y_2 - \Delta y_1}{x_3 - x_1} = \frac{\frac{y_3 - y_2}{x_3 - x_2} - \frac{y_2 - y_1}{x_2 - x_1}}{x_3 - x_1} = \frac{\frac{13,14 - 12,05}{5,0 - 4,5} - \frac{12,05 - 10,91}{4,5 - 4,0}}{5,0 - 4,0} = \frac{-0,05}{2!0,5^2} = -0,10.$$

3.4.2 Fórmula de Gregory-Newton

Seja a fórmula do polinômio interpolador de Newton (3.10) na forma

$$P_n(x) = y_0 + \Delta y_0(x - x_0) + \Delta^2 y_0(x - x_0)(x - x_1) + \dots + \Delta^n y_0(x - x_0)\dots(x - x_{n-1})$$

e uma variável auxiliar

$$u_x = u(x) = \frac{x - x_0}{h},$$

da qual verifica-se que

$$x - x_0 = hu_x,$$

$$x - x_1 = x - (x_0 + h) = x - x_0 - h = hu_x - h \sim x - x_1 = h(u_x - 1),$$

$$x - x_2 = x - (x_0 + 2h) = x - x_0 - 2h = hu_x - 2h \sim x - x_2 = h(u_x - 2),$$

⋮

$$x - x_{n-1} = x - (x_0 + (n-1)h) = x - x_0 - (n-1)h \sim x - x_{n-1} = h(u_x - n+1).$$

Substituindo esses valores na fórmula de Newton e aplicando a relação (3.11) entre operadores, tem-se

$$\begin{aligned} P_n(x) &= y_0 + \frac{\Delta y_0}{1!h} hu_x + \frac{\Delta^2 y_0}{2!h^2} hu_x h(u_x - 1) + \dots + \\ &\quad \frac{\Delta^n y_0}{n!h^n} hu_x h(u_x - 1) \dots h(u_x - n+1). \end{aligned}$$

Assim,

$$P_n(x) = y_0 + \Delta y_0 u_x + \frac{\Delta^2 y_0}{2!} u_x(u_x - 1) + \dots + \frac{\Delta^n y_0}{n!} u_x(u_x - 1) \dots (u_x - n+1)$$

$$P_n(x) = y_0 + \sum_{i=1}^n \frac{\Delta^i y_0}{i!} \prod_{j=0}^{i-1} (u_x - j).$$

Exemplo 3.15 Calcular $P_1(0,2)$, usando os dados da tabela do Exemplo 3.1.

Usando (3.12) com $n = 1$, obtém-se

$$P_1(x) = y_0 + \Delta y_0 u_x.$$

A tabela de diferenças finitas é

| i | x_i | y_i | Δy_i |
|-----|-------|-------|--------------|
| 0 | 0,1 | 1,221 | 2,099 |
| 1 | 0,6 | 3,320 | |

E a variável $u_x = \frac{x - x_0}{h} = \frac{0,2 - 0,1}{0,5} = 0,2$. Então

$$P_1(0,2) = 1,221 + 2,099(0,2) \sim P_1(0,2) = 1,641.$$

Este resultado é o mesmo obtido por (3.1) no Exemplo 3.1, por (3.5) no Exemplo 3.3 e por (3.10) no Exemplo 3.8. De fato,

$$P_1(x) = y_0 + \Delta y_0 u_x = y_0 + (y_1 - y_0) \frac{x - x_0}{x_1 - x_0} \quad (\text{comparar com (3.1)}).$$

Exemplo 3.16 Calcular $P_2(115)$ a partir da tabela

| x | 110 | 120 | 130 |
|-----|-------|-------|-------|
| y | 2,041 | 2,079 | 2,114 |

Usando (3.12) com $n = 2$, tem-se

$$P_2(x) = y_0 + \Delta y_0 u_x + \frac{\Delta^2 y_0}{2!} u_x(u_x - 1).$$

A tabela de diferenças finitas é

| i | x_i | y_i | Δy_i | $\Delta^2 y_i$ |
|-----|-------|-------|--------------|----------------|
| 0 | 110 | 2,041 | 0,038 | -0,003 |
| 1 | 120 | 2,079 | 0,035 | |
| 2 | 130 | 2,114 | | |

E a variável $u_x = \frac{x - x_0}{h} = \frac{115 - 110}{10} = 0,5$. Assim,

$$P_2(115) = 2,041 + (0,038)(0,5) + \frac{-0,003}{2}(0,5)(0,5 - 1) \sim P_2(115) = 2,060.$$

3.4.3 Algoritmo e complexidade computacional

Um algoritmo para interpolar um valor z em uma tabela definida pelos vetores x e y , de dimensões m , usando um polinômio de Gregory-Newton de diferenças finitas de grau $m-1$, é mostrado na Figura 3.4. Os parâmetros de entrada são o número m de pontos, o vetor x contendo as m abscissas x_i , o vetor y com as m ordenadas y_i e o ponto z a ser interpolado. O parâmetro de saída é o valor r do polinômio de Gregory-Newton de grau $m-1$ avaliado no ponto z .

```

Algoritmo Polinômio_Gregory-Newton
{ Objetivo: Interpolar valor em tabela usando polinômio de Gregory-Newton }
parâmetros de entrada m, x, y, z
{ número de pontos, abscissas, ordenadas e valor a interpolar }
parâmetros de saída r { valor interpolado }
para i ← 1 até m faça
    Dely(i) ← y(i)
fimpara
{ construção das diferenças finitas }
para k ← 1 até m - 1 faça
    para i ← m até k + 1 passo -1 faça
        Dely(i) ← Dely(i) - Dely(i - 1)
    fimpara
fimpara
{ avaliação do polinômio pelo processo de Horner }
u ← (z - x(1)) / (x(2) - x(1))
r ← Dely(m)
para i ← m - 1 até 1 passo -1 faça
    r ← r + ((u - i + 1) / i + Dely(i))
fimpara
finalgoritmo

```

Figura 3.4 Interpolação por polinômio de Gregory-Newton com diferenças finitas.

Também neste caso, o polinômio (3.12)

$$P_n(x) = y_0 + \sum_{i=1}^n \frac{\Delta^i y_0}{i!} \prod_{j=0}^{i-1} (u_x - j)$$

pode ser escrito de forma a ser avaliado pelo processo de Horner

$$P_n(x) = \left(\left(\left(\dots \left(\Delta^n y_0 \frac{u_x - n+1}{n} + \dots + \Delta^2 y_0 \right) \frac{u_x - 1}{2} + \Delta y_0 \right) \frac{u_x - 0}{1} \right) + y_0 \right).$$

De forma análoga ao método de Newton, este algoritmo utiliza um vetor auxiliar $Dely$, no qual é construída a tabela de diferenças finitas. Os valores de $\Delta^0 y_0, \Delta y_0, \dots, \Delta^n y_0$ são

armazenados nas primeiras $n+1$ posições de $Dely$, para economizar espaço de memória, conforme o esquema para quatro pontos

| i | x_i | y_i | $Dely_i^{(1)}$ | $Dely_i^{(2)}$ | $Dely_i^{(3)}$ |
|-----|-------|-------|----------------|----------------|----------------|
| 1 | x_0 | y_0 | $\Delta^0 y_0$ | $\Delta^0 y_0$ | $\Delta^0 y_0$ |
| 2 | x_1 | y_1 | Δy_0 | Δy_0 | Δy_0 |
| 3 | x_2 | y_2 | Δy_1 | $\Delta^2 y_0$ | $\Delta^2 y_0$ |
| 4 | x_3 | y_3 | Δy_2 | $\Delta^2 y_1$ | $\Delta^3 y_0$ |

onde $Dely_i^{(k)}$ significa i -ésima posição do vetor $Dely$ na k -ésima repetição do comando para $k \leftarrow 1$ até $m-1$ faça. A Tabela 3.4 mostra a complexidade computacional do algoritmo da Figura 3.4. Comparando com a complexidade do algoritmo de Newton, dado na Tabela 3.3, verifica-se que o algoritmo de Gregory-Newton apresenta uma complexidade menor em relação à operação de divisão.

Tabela 3.4 Complexidade da interpolação de Gregory-Newton.

(n : grau do polinômio interpolador.)

| Operações | Complexidade |
|----------------|----------------|
| adições | $n^2 + 4n + 2$ |
| multiplicações | n |
| divisões | $n + 1$ |

Exemplo 3.17 Calcular $P_2(115)$ com os dados da tabela do Exemplo 3.16 usando o algoritmo da Figura 3.4.

```

% Os parametros de entrada
n = 3
x = 110 120 130
y = 2.0410 2.0790 2.1140
z = 115
% produzem o resultado
r = 2.0604

```

Além disso, a sequência de vetores $Dely$ produzida pelo algoritmo é

| i | x_i | y_i | $Dely_i^{(1)}$ | $Dely_i^{(2)}$ |
|-----|-------|-------|----------------|----------------|
| 1 | 110 | 2,041 | 2,041 | 2,041 |
| 2 | 120 | 2,079 | 0,038 | 0,038 |
| 3 | 130 | 2,114 | 0,035 | -0,003 |

3.5 Escolha dos pontos para interpolação

Uma questão importante que surge no uso da interpolação é como fazer a escolha dos pontos a serem utilizados. Até agora foram vistos exemplos nos quais todos os pontos da tabela eram usados. No entanto, na prática, faz-se necessário escolher $n+1$ pontos dentre os m valores de uma tabela, sendo $m > n+1$, para que seja construído um polinômio interpolador de grau n . Esta escolha é importante, pois não se devem construir polinômios de grau elevado por causa do erro de arredondamento, e deve-se evitar uma extrapolação na qual o valor de z esteja fora do intervalo dos pontos utilizados para construir o polinômio.

Exemplo 3.18 Interpolar $z = 1,4$ usando um polinômio de terceiro grau com os dados

| | | | | | | | |
|---|-------|-------|-------|-------|-------|--------|--------|
| x | 0,7 | 1,2 | 1,3 | 1,5 | 2,0 | 2,3 | 2,6 |
| y | 0,043 | 1,928 | 2,497 | 3,875 | 9,000 | 13,467 | 19,176 |

Para determinar um polinômio interpolador de grau 3 são necessários 4 pontos. O ponto a ser interpolado deve ser o mais próximo possível destes 4 pontos. Inicialmente são escolhidos 2 pontos, devendo o valor a ser interpolado (1,4) estar entre eles, a saber 1,3 e 1,5. O terceiro ponto será 1,2 e não 2,0, porque $1,4 - 1,2 < 2,0 - 1,4$. Analogamente, o quarto ponto será 2,0 e não 0,7, pois $2,0 - 1,4 < 1,4 - 0,7$. Assim, a interpolação cúbica utilizará os quatro pontos

| i | 0 | 1 | 2 | 3 |
|-------|-------|-------|-------|-------|
| x_i | 1,2 | 1,3 | 1,5 | 2,0 |
| y_i | 1,928 | 2,497 | 3,875 | 9,000 |

Usando o polinômio de Lagrange, a matriz G é

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

Por isso,

$$\begin{aligned} G_d &= (0,2)(0,1)(-0,1)(-0,6) = 1,2 \times 10^{-3}, \\ G_0 &= (0,2)(-0,1)(-0,3)(-0,8) = -4,8 \times 10^{-3}, \\ G_1 &= (0,1)(0,1)(-0,2)(-0,7) = 1,4 \times 10^{-3}, \\ G_2 &= (0,3)(0,2)(-0,1)(-0,5) = 3,0 \times 10^{-3} \text{ e} \\ G_3 &= (0,8)(0,7)(0,5)(-0,6) = -1,68 \times 10^{-1}. \end{aligned}$$

Usando (3.6) com $n = 3$, tem-se

$$\begin{aligned} L_3(x) &= G_d \left(\frac{y_0}{G_0} + \frac{y_1}{G_1} + \frac{y_2}{G_2} + \frac{y_3}{G_3} \right), \\ L_3(1,4) &= 1,2 \times 10^{-3} \left(\frac{1,928}{-4,8 \times 10^{-3}} + \frac{2,497}{1,4 \times 10^{-3}} + \frac{3,875}{3,0 \times 10^{-3}} + \frac{9,000}{-1,68 \times 10^{-1}} \right) \rightsquigarrow \\ L_3(1,4) &= 3,144. \end{aligned}$$

3.6 Erro de truncamento da interpolação polinomial

Erro de truncamento é o erro cometido ao aproximar uma função $f(x)$ por um polinômio interpolador. Pode ser mostrado que se $P_n(x)$ for um polinômio interpolador de grau n de Lagrange, Newton ou Gregory-Newton, então o erro de truncamento [22] é dado pela expressão

$$T_n(x) = \frac{f^{n+1}(\xi)}{(n+1)!} \prod_{i=0}^n (x - x_i), \quad x_0 < \xi < x_n. \quad (3.13)$$

Sendo $f(x)$ definida no intervalo $[a, b]$ que contém os pontos x_0, x_1, \dots, x_n , e supondo que a derivada $f^{n+1}(x)$ existe e é contínua no intervalo (a, b) , então, devido à dificuldade de localizar ξ , na prática ele é tomado como o ponto no intervalo $[x_0, x_n] \subset (a, b)$, onde $f^{n+1}(x)$ apresenta o maior valor em módulo. Deste modo, (3.13) fornecerá a cota máxima do erro de truncamento cometido.

Exemplo 3.19 Sendo $f(x) = 2x^4 + 3x^2 + 1$, calcular $P_2(0,1)$ e $T_2(0,1)$ a partir da tabela

| | | | |
|---|--------|--------|--------|
| x | 0,0 | 0,2 | 0,4 |
| y | 1,0000 | 1,1232 | 1,5312 |

Cálculo de $P_2(0,1)$:

Utilizando o polinômio de Gregory-Newton de grau 2, dado por (3.12), tem-se

$$P_2(x) = y_0 + \Delta y_0 u_x + \frac{\Delta^2 y_0}{2} u_x(u_x - 1).$$

A tabela de diferenças finitas é

| i | x_i | y_i | Δy_i | $\Delta^2 y_i$ |
|---|-------|--------|--------------|----------------|
| 0 | 0,0 | 1,0000 | 0,1232 | 0,2848 |
| 1 | 0,2 | 1,1232 | 0,4080 | |
| 2 | 0,4 | 1,5312 | | |

e a variável $u_x = \frac{x - x_0}{h} = \frac{0,1 - 0,0}{0,2} \rightsquigarrow u_x = 0,5$. Deste modo,

$$P_2(0,1) = 1,0000 + 0,1232(0,5) + \frac{0,2848}{2}(0,5)(0,5 - 1) \rightsquigarrow P_2(0,1) = 1,0260.$$

Cálculo de $T_2(0,1)$:

$$f(x) = 2x^4 + 3x^2 + 1, \quad f'(x) = 8x^3 + 6x, \quad f''(x) = 24x^2 + 6, \quad f'''(x) = 48x \rightsquigarrow \xi = 0,4.$$

Logo,

$$T_2(x) = \frac{f'''(\xi)}{3!} (x - x_0)(x - x_1)(x - x_2) \text{ e}$$

$$T_2(0,1) = \frac{48(0,4)}{6}(0,1 - 0,0)(0,1 - 0,2)(0,1 - 0,4) \rightsquigarrow T_2(0,1) = 0,0096.$$

Pode ser verificado que, de fato, neste exemplo, (3.13) fornece a cota máxima do erro de truncamento, pois o erro real cometido foi

$$|f(0,1) - P_2(0,1)| = |1,0302 - 1,0260| = 0,0042 < T_2(0,1).$$

Apesar de (3.13) ser de maior interesse na análise teórica da interpolação, ela mostra claramente que o erro de truncamento é diretamente proporcional ao produto das distâncias entre o valor interpolado e os pontos-base. Portanto, os pontos escolhidos para construir o polinômio interpolador devem ser os mais próximos do ponto a ser interpolado (ver Seção 3.5), conforme será mostrado no próximo exemplo.

Exemplo 3.20 Seja a tabela da função $f(x) = e^x - x^2 - x$

| | | | | | | | |
|---|--------|--------|--------|--------|--------|--------|---------|
| x | 1,1 | 1,4 | 1,9 | 2,1 | 2,5 | 3,0 | 3,2 |
| y | 0,6942 | 0,6952 | 1,1759 | 1,6562 | 3,4325 | 8,0855 | 11,0925 |

Para calcular $P_2(2,2)$, são necessários 3 pontos. A fim de garantir que não seja efetuada uma extrapolação, devem ser usados os pontos de abscissas $x = 2,1$ e $x = 2,5$. O terceiro ponto pode ser escolhido entre $x_a = 1,9$ e $x_b = 3,0$. Como os pontos não são igualmente espaçados, o método de Lagrange ou de Newton deve ser utilizado.

Cálculo de $P_{2,a}(2,2)$ pelo método de Newton utilizando os pontos

| | | | |
|---|--------|--------|--------|
| x | 1,9 | 2,1 | 2,5 |
| y | 1,1759 | 1,6562 | 3,4325 |

Por (3.10) com $n = 2$,

$$P_2(x) = y_0 + \Delta y_0(x - x_0) + \Delta^2 y_0(x - x_0)(x - x_1).$$

A tabela de diferenças divididas é

| i | x_i | y_i | Δy_i | $\Delta^2 y_i$ |
|---|-------|--------|--------------|----------------|
| 0 | 1,9 | 1,1759 | 2,4015 | 3,3988 |
| 1 | 2,1 | 1,6562 | 4,4408 | |
| 2 | 2,5 | 3,4325 | | |

$$P_{2,a}(2,2) = 1,1759 + 2,4015(2,2 - 1,9) + 3,3988(2,2 - 1,9)(2,2 - 2,1) \rightsquigarrow$$

$$P_{2,a}(2,2) = 1,9983.$$

O erro de truncamento é, por (3.13),

$$T_{2,a}(x) = \frac{f'''(\xi)}{3!}(x - x_0)(x - x_1)(x - x_2), \text{ para algum } \xi, x_0 < \xi < x_2.$$

Como $f'''(x) = e^x$, então para $\xi = 2,5$, $|T_{2,a}(2,2)|$ é a cota máxima do erro de truncamento

$$T_{2,a}(2,2) = \frac{e^{2,5}}{6}(2,2 - 1,9)(2,2 - 2,1)(2,2 - 2,5) \rightsquigarrow T_{2,a}(2,2) = -0,0183,$$

onde o sinal negativo indica que a interpolação foi por excesso, isto é, $P_{2,a}(2,2) > f(2,2)$. O erro real cometido é

$$|f(2,2) - P_{2,a}(2,2)| = |1,9850 - 1,9983| = 0,0133 < |T_{2,a}(2,2)|.$$

Cálculo de $P_{2,b}(2,2)$ pelo método de Newton utilizando os pontos

| | | | |
|---|--------|--------|--------|
| x | 2,1 | 2,5 | 3,0 |
| y | 1,6562 | 3,4325 | 8,0855 |

A tabela de diferenças divididas é

| i | x_i | y_i | Δy_i | $\Delta^2 y_i$ |
|---|-------|--------|--------------|----------------|
| 0 | 2,1 | 1,6562 | 4,4408 | 5,4058 |
| 1 | 2,5 | 3,4325 | 9,3060 | |
| 2 | 3,0 | 8,0855 | | |

$$P_{2,b}(2,2) = 1,6562 + 4,4408(2,2 - 2,1) + 5,4058(2,2 - 2,1)(2,2 - 2,5) \rightsquigarrow$$

$$P_{2,b}(2,2) = 1,9381, \text{ e para } \xi = 3,0, \text{ tem-se}$$

$$T_{2,b}(2,2) = \frac{e^{3,0}}{6}(2,2 - 2,1)(2,2 - 2,5)(2,2 - 3,0) \rightsquigarrow T_{2,b}(2,2) = 0,0803.$$

Neste caso, o valor positivo do erro de truncamento indica que a interpolação foi por falta, ou seja, $P_{2,b}(2,2) < f(2,2)$. O erro real é

$$|f(2,2) - P_{2,b}(2,2)| = |1,9850 - 1,9381| = 0,0469 < |T_{2,b}(2,2)|.$$

Conseqüentemente, como o ponto-base $x_a = 1,9$ está mais próximo do valor interpolado $x = 2,2$ do que $x_b = 3,0$, tem-se que

$$|f(2,2) - P_{2,a}(2,2)| < |f(2,2) - P_{2,b}(2,2)| \text{ e } |T_{2,a}(2,2)| < |T_{2,b}(2,2)|.$$

3.7 Comparação das complexidades

A Tabela 3.5 apresenta uma compilação das Tabelas 3.2, 3.3 e 3.4 referentes às complexidades dos algoritmos de interpolação utilizando polinômios de Lagrange, Newton e Gregory-Newton, respectivamente.

Tabela 3.5 Complexidade dos algoritmos de interpolação.

(n : grau do polinômio interpolador.)

| Polinômio | Adições | Multiplicações | Divisões |
|----------------|-----------------|-----------------|---------------------------------|
| Lagrange | $2n^2 + 3n + 1$ | $2n^2 + 3n + 1$ | $n + 1$ |
| Newton | $n^2 + 4n$ | n | $\frac{1}{2}n^2 + \frac{1}{2}n$ |
| Gregory-Newton | $n^2 + 4n + 2$ | n | $n + 1$ |

A Figura 3.5 exibe os gráficos dos polinômios de complexidade da interpolação para as operações de adição (a) e divisão (b).

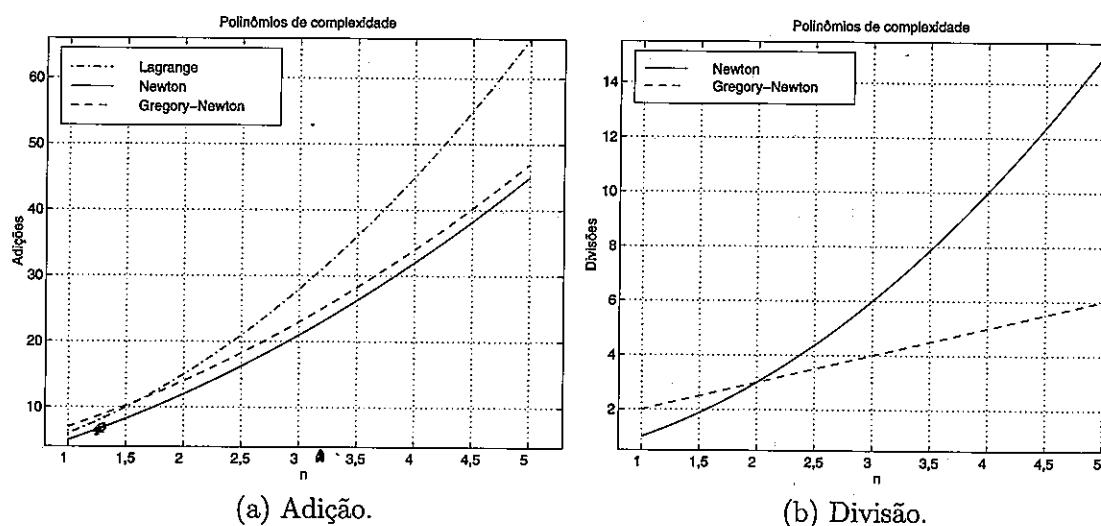


Figura 3.5 Valores dos polinômios de complexidade para interpolação.

Pela Figura 3.5(a), os polinômios de Newton são os que apresentam menor complexidade para a operação de adição. No que diz respeito à operação de divisão, as complexidades dos algoritmos de Lagrange e Gregory-Newton são lineares, enquanto o de Newton é quadrático. Entretanto, pela Figura 3.5(b), o polinômio de Newton é o mais indicado quando $n \leq 2$. É óbvio, pela Tabela 3.5, que os polinômios de Lagrange são os menos eficientes em termos da operação de multiplicação. Resumindo, para polinômios de grau $n \leq 2$ é preferível utilizar a fórmula de Newton; caso contrário, um polinômio de Gregory-Newton é o mais indicado se os pontos forem igualmente espaçados.

3.8 Splines cúbicos

Sejam $n + 1$ pontos (x_i, y_i) , $i = 0, 1, 2, \dots, n$, com $x_0 < x_1 < \dots < x_{n-1} < x_n$. Deseja-se construir n polinômios interpoladores cúbicos $s_i(x)$, denominados *splines*¹ cúbicos, que passem por dois pontos sucessivos (x_i, y_i) e (x_{i+1}, y_{i+1}) , ou seja, cada polinômio é utilizado no intervalo $[x_i, x_{i+1}]$, sendo da forma

$$s_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i, \quad i = 0, 1, 2, \dots, n-1, \quad (3.14)$$

e que satisfaçam às condições

$$s_i(x_i) = y_i, \quad i = 0, 1, 2, \dots, n-1 \text{ e } s_{n-1}(x_n) = y_n, \quad (3.15)$$

$$s_i(x_{i+1}) = s_{i+1}(x_{i+1}), \quad i = 0, 1, 2, \dots, n-2, \quad (3.16)$$

de modo a garantir que os *splines* cúbicos passem pelos pontos (x_i, y_i) e sejam contínuos. Para garantir também que as inclinações e curvaturas sejam contínuas, impõe-se que

$$s'_i(x_{i+1}) = s'_{i+1}(x_{i+1}), \quad i = 0, 1, 2, \dots, n-2, \quad (3.17)$$

$$s''_i(x_{i+1}) = s''_{i+1}(x_{i+1}), \quad i = 0, 1, 2, \dots, n-2. \quad (3.18)$$

Desta forma, obtém-se de (3.14) n equações com $4n$ incógnitas a_i , b_i , c_i e d_i . No entanto, as condições (3.15) a (3.18) fornecem apenas $4n - 2$ equações, sendo necessárias mais duas equações para calcular todas as $4n$ incógnitas.

3.8.1 Cálculo dos coeficientes

Para $x = x_i$ em (3.14) e comparando com (3.15)

$$\begin{aligned} s_i(x_i) &= d_i, \\ d_i &= y_i, \quad i = 0, 1, 2, \dots, n-1. \end{aligned} \quad (3.19)$$

Para $x = x_{i+1}$ em (3.14) e comparando com (3.16) em vista de (3.15)

$$\begin{aligned} s_i(x_{i+1}) &= s_{i+1}(x_{i+1}) = y_{i+1}, \\ a_i(x_{i+1} - x_i)^3 + b_i(x_{i+1} - x_i)^2 + c_i(x_{i+1} - x_i) + d_i &= y_{i+1}. \end{aligned}$$

Definindo

$$h_i = x_{i+1} - x_i, \quad (3.20)$$

e substituindo (3.19), tem-se

$$a_i h_i^3 + b_i h_i^2 + c_i h_i + y_i = y_{i+1}. \quad (3.21)$$

¹O termo inglês *splines* significa longas tiras de madeira. Elas eram usadas antigamente para interpolar, de modo suave, pontos das estruturas de navios e aviões.

Por outro lado, as derivadas de (3.14) são

$$s'_i(x) = 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i, \quad (3.22)$$

$$s''_i(x) = 6a_i(x - x_i) + 2b_i. \quad (3.23)$$

Para $x = x_i$ em (3.23)

$$\begin{aligned} s''_i(x_i) &= 6a_i(x_i - x_i) + 2b_i, \\ b_i &= \frac{s''_i(x_i)}{2}, \quad i = 0, 1, 2, \dots, n-1. \end{aligned} \quad (3.24)$$

Para $x = x_{i+1}$ em (3.23)

$$s''_i(x_{i+1}) = 6a_i(x_{i+1} - x_i) + 2b_i.$$

Em vista de (3.18) e substituindo (3.20) e (3.24)

$$\begin{aligned} s''_{i+1}(x_{i+1}) &= 6a_i h_i + 2 \frac{s''_i(x_i)}{2}, \\ a_i &= \frac{s''_{i+1}(x_{i+1}) - s''_i(x_i)}{6h_i}, \quad i = 0, 1, 2, \dots, n-1. \end{aligned} \quad (3.25)$$

Substituindo (3.19), (3.24) e (3.25) em (3.21)

$$\frac{s''_{i+1}(x_{i+1}) - s''_i(x_i)}{6h_i} h_i^3 + \frac{s''_i(x_i)}{2} h_i^2 + c_i h_i + y_i = y_{i+1},$$

obtém-se

$$c_i = \Delta y_i - \frac{s''_{i+1}(x_{i+1}) + 2s''_i(x_i)}{6} h_i, \quad i = 0, 1, 2, \dots, n-1, \quad (3.26)$$

sendo o operador de diferença dividida Δy_i definido na Seção 3.3

$$\Delta y_i = \frac{y_{i+1} - y_i}{h_i}. \quad (3.27)$$

3.8.2 Sistema linear subdeterminado

Impondo a condição (3.17) que as inclinações de dois splines cúbicos adjacentes $s_{i-1}(x)$ e $s_i(x)$ sejam iguais no ponto comum (x_i, y_i)

$$s'_{i-1}(x_i) = s'_i(x_i)$$

e em vista de (3.22), tem-se

$$3a_{i-1}(x_i - x_{i-1})^2 + 2b_{i-1}(x_i - x_{i-1}) + c_{i-1} = 3a_i(x_i - x_i)^2 + 2b_i(x_i - x_i) + c_i.$$

Substituindo (3.20), (3.24), (3.25) e (3.26)

$$\begin{aligned} &3 \frac{s''_i(x_i) - s''_{i-1}(x_{i-1})}{6h_{i-1}} h_{i-1}^2 + 2 \frac{s''_{i-1}(x_{i-1})}{2} h_{i-1} + \frac{y_i - y_{i-1}}{h_{i-1}} - \frac{s''_i(x_i) + 2s''_{i-1}(x_{i-1})}{6} h_{i-1} \\ &= \frac{y_{i+1} - y_i}{h_i} - \frac{s''_{i+1}(x_{i+1}) + 2s''_i(x_i)}{6} h_i, \end{aligned}$$

e simplificando, obtém-se a i -ésima equação para $i = 1, 2, 3, \dots, n-1$

$$h_{i-1}s''_{i-1}(x_{i-1}) + 2(h_{i-1} + h_i)s''_i(x_i) + h_i s''_{i+1}(x_{i+1}) = 6(\Delta y_i - \Delta y_{i-1}). \quad (3.28)$$

que é um sistema linear subdeterminado com $n-1$ equações e $n+1$ incógnitas $s''_i(x_i)$, $i = 0, 1, 2, \dots, n$. O sistema linear (3.28) é da forma

$$\left[\begin{array}{cccccc} h_0 & 2(h_0+h_1) & h_1 & & & \\ & h_1 & 2(h_1+h_2) & h_2 & & \\ & & h_2 & 2(h_2+h_3) & h_3 & \\ & & & \ddots & \ddots & \\ & & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) \\ & & & & & h_{n-1} \end{array} \right] \left[\begin{array}{c} s''_0(x_0) \\ s''_1(x_1) \\ s''_2(x_2) \\ \vdots \\ s''_{n-1}(x_{n-1}) \\ s''_n(x_n) \end{array} \right] = 6 \left[\begin{array}{c} \Delta y_1 - \Delta y_0 \\ \Delta y_2 - \Delta y_1 \\ \vdots \\ \Delta y_{n-1} - \Delta y_{n-2} \end{array} \right].$$

Duas incógnitas podem ser eliminadas de modo a obter um sistema linear com matriz quadrada de ordem $n-1$. Esta eliminação pode ocorrer de várias formas, conforme será visto mais adiante.

3.9 Splines cúbicos naturais

A forma mais simples e freqüentemente usada de eliminar duas incógnitas do sistema (3.28) consiste em atribuir

$$\left. \begin{array}{l} s''_0(x_0) = 0, \\ s''_n(x_n) = 0 \end{array} \right\} \quad (3.29)$$

3.9.1 Cálculo das derivadas

Substituindo o valor de $s''_0(x_0)$ na primeira equação do sistema (3.28) e $s''_n(x_n)$ na última equação, obtém-se o sistema linear tridiagonal simétrico (3.30), cuja solução fornece as derivadas $s''_i(x_i)$, $i = 1, 2, 3, \dots, n-1$

$$\left[\begin{array}{cccccc} 2(h_0+h_1) & h_1 & & & & \\ h_1 & 2(h_1+h_2) & h_2 & & & \\ & h_2 & 2(h_2+h_3) & h_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \end{array} \right] \left[\begin{array}{c} s''_1(x_1) \\ s''_2(x_2) \\ s''_3(x_3) \\ \vdots \\ s''_{n-1}(x_{n-1}) \end{array} \right] = 6 \left[\begin{array}{c} \Delta y_1 - \Delta y_0 \\ \Delta y_2 - \Delta y_1 \\ \vdots \\ \Delta y_{n-1} - \Delta y_{n-2} \end{array} \right]. \quad (3.30)$$

Com essas derivadas têm-se os chamados *splines* cúbicos naturais, que devem ser usados quando $y = f(x)$ apresentar um comportamento linear nas proximidades dos pontos finais x_0 e x_n .

Exemplo 3.21 Dados os pontos $(1, 2)$, $(2, 4)$, $(4, 1)$, $(6, 3)$ e $(7, 3)$, calcular as segundas derivadas $s_i''(x_i)$, $i = 0, 1, 2, 3, 4$ dos *splines* cúbicos naturais.

Por (3.20)

$$h_0 = x_1 - x_0 = 2 - 1 \rightsquigarrow h_0 = 1, \quad h_1 = x_2 - x_1 = 4 - 2 \rightsquigarrow h_1 = 2,$$

$$h_2 = x_3 - x_2 = 6 - 4 \rightsquigarrow h_2 = 2, \quad h_3 = x_4 - x_3 = 7 - 6 \rightsquigarrow h_3 = 1.$$

Usando (3.27)

$$\Delta y_0 = \frac{y_1 - y_0}{h_0} = \frac{4 - 2}{1} \rightsquigarrow \Delta y_0 = 2, \quad \Delta y_1 = \frac{y_2 - y_1}{h_1} = \frac{1 - 4}{2} \rightsquigarrow \Delta y_1 = -1,5;$$

$$\Delta y_2 = \frac{y_3 - y_2}{h_2} = \frac{3 - 1}{2} \rightsquigarrow \Delta y_2 = 1, \quad \Delta y_3 = \frac{y_4 - y_3}{h_3} = \frac{3 - 3}{1} \rightsquigarrow \Delta y_3 = 0.$$

Substituindo os valores acima em (3.30), tem-se

$$\begin{bmatrix} 2(1+2) & 2 & 0 \\ 2 & 2(2+2) & 2 \\ 0 & 2 & 2(2+1) \end{bmatrix} \begin{bmatrix} s_1''(x_1) \\ s_2''(x_2) \\ s_3''(x_3) \end{bmatrix} = 6 \begin{bmatrix} -1,5 - 2 \\ 1 - (-1,5) \\ 0 - 1 \end{bmatrix} \rightsquigarrow s''(x) = \begin{bmatrix} -4,7 \\ 3,6 \\ -2,2 \end{bmatrix}.$$

A partir de (3.29) e da solução do sistema acima, obtém-se as segundas derivadas

$$s_0''(x_0) = 0; \quad s_1''(x_1) = -4,7; \quad s_2''(x_2) = 3,6; \quad s_3''(x_3) = -2,2; \quad s_4''(x_4) = 0.$$

Exemplo 3.22 A partir dos pontos do Exemplo 3.21, determinar as equações dos quatro *splines* cúbicos naturais na forma (3.14).

Determinação do *spline* $s_0(x)$

$$a_0 = \frac{s_1''(x_1) - s_0''(x_0)}{6h_0} = \frac{-4,7 - 0}{6 \times 1} \rightsquigarrow a_0 = -\frac{47}{60};$$

$$b_0 = \frac{s_0''(x_0)}{2} = \frac{0}{2} \rightsquigarrow b_0 = 0;$$

$$c_0 = \Delta y_0 - \frac{s_1''(x_1) + 2s_0''(x_0)}{6} h_0 = 2 - \frac{-4,7 + 2 \times 0}{6} \times 1 \rightsquigarrow c_0 = \frac{167}{60};$$

$$d_0 = y_0 \rightsquigarrow d_0 = 2,$$

$$s_0(x) = -\frac{47}{60}(x-1)^3 + 0(x-1)^2 + \frac{167}{60}(x-1) + 2.$$

Determinação do *spline* $s_1(x)$

$$a_1 = \frac{s_2''(x_2) - s_1''(x_1)}{6h_1} = \frac{3,6 - (-4,7)}{6 \times 2} \rightsquigarrow a_1 = \frac{83}{120};$$

$$b_1 = \frac{s_1''(x_1)}{2} = \frac{-4,7}{2} \rightsquigarrow b_1 = -\frac{47}{20};$$

$$c_1 = \Delta y_1 - \frac{s_2''(x_2) + 2s_1''(x_1)}{6} h_1 = -1,5 - \frac{3,6 + 2 \times -4,7}{6} \times 2 \rightsquigarrow c_1 = \frac{13}{30};$$

$$d_1 = y_1 \rightsquigarrow d_1 = 4,$$

$$s_1(x) = \frac{83}{120}(x-2)^3 - \frac{47}{20}(x-2)^2 + \frac{13}{30}(x-2) + 4.$$

Determinação do *spline* $s_2(x)$

$$a_2 = \frac{s_3''(x_3) - s_2''(x_2)}{6h_2} = \frac{-2,2 - 3,6}{6 \times 2} \rightsquigarrow a_2 = -\frac{29}{60};$$

$$b_2 = \frac{s_2''(x_2)}{2} = \frac{3,6}{2} \rightsquigarrow b_2 = \frac{9}{5};$$

$$c_2 = \Delta y_2 - \frac{s_3''(x_3) + 2s_2''(x_2)}{6} h_2 = 1 - \frac{-2,2 + 2 \times 3,6}{6} \times 2 \rightsquigarrow c_2 = -\frac{2}{3};$$

$$d_2 = y_2 \rightsquigarrow d_2 = 1,$$

$$s_2(x) = -\frac{29}{60}(x-4)^3 + \frac{9}{5}(x-4)^2 - \frac{2}{3}(x-4) + 1.$$

Determinação do *spline* $s_3(x)$

$$a_3 = \frac{s_4''(x_4) - s_3''(x_3)}{6h_3} = \frac{0 - (-2,2)}{6 \times 1} \rightsquigarrow a_3 = \frac{11}{30};$$

$$b_3 = \frac{s_3''(x_3)}{2} = \frac{-2,2}{2} \rightsquigarrow b_3 = -\frac{11}{10};$$

$$c_3 = \Delta y_3 - \frac{s_4''(x_4) + 2s_3''(x_3)}{6} h_3 = 0 - \frac{0 + 2 \times -2,2}{6} \times 1 \rightsquigarrow c_3 = \frac{11}{15};$$

$$d_3 = y_3 \rightsquigarrow d_3 = 3,$$

$$s_3(x) = \frac{11}{30}(x-6)^3 - \frac{11}{10}(x-6)^2 + \frac{11}{15}(x-6) + 3.$$

As derivadas dos *splines* naturais são

$$s_0'(x) = -\frac{47}{20}(x-1)^2 + \frac{167}{60} \quad \text{e} \quad s_0''(x) = -\frac{47}{10}(x-1),$$

$$s_1'(x) = \frac{83}{40}(x-2)^2 - \frac{47}{10}(x-2) + \frac{13}{30} \quad \text{e} \quad s_1''(x) = \frac{83}{20}(x-2) - \frac{47}{10},$$

$$s_2'(x) = -\frac{29}{20}(x-4)^2 + \frac{18}{5}(x-4) - \frac{2}{3} \text{ e } s_2''(x) = -\frac{29}{10}(x-4) + \frac{18}{5},$$

$$s_3'(x) = \frac{11}{10}(x-6)^2 - \frac{11}{5}(x-6) + \frac{11}{15} \text{ e } s_3''(x) = \frac{11}{5}(x-6) - \frac{11}{5}.$$

Pela condição (3.16), os *splines* são contínuos

$$s_i(x_{i+1}) = s_{i+1}(x_{i+1}); s_0(2) = s_1(2) = 4; s_1(4) = s_2(4) = 1 \text{ e } s_2(6) = s_3(6) = 3.$$

As primeiras derivadas, pela condição (3.17)

$$s_i'(x_{i+1}) = s_{i+1}'(x_{i+1}); s_0'(2) = s_1'(2) = \frac{13}{30}; s_1'(4) = s_2'(4) = -\frac{2}{3} \text{ e } s_2'(6) = s_3'(6) = \frac{11}{15},$$

e as segundas derivadas, pela condição (3.18)

$$s_i''(x_{i+1}) = s_{i+1}''(x_{i+1}); s_0''(2) = s_1''(2) = -\frac{47}{10}; s_1''(4) = s_2''(4) = \frac{18}{5} \text{ e } s_2''(6) = s_3''(6) = -\frac{11}{5},$$

também são contínuas. ■

Exemplo 3.23 Interpolar os valores $z = 1,2; 2,9; 5,2$ e $6,7$ usando os *splines* cúbicos naturais obtidos no Exemplo 3.22.

$$s_0(1,2) = -\frac{47}{60}(1,2-1)^3 + 0(1,2-1)^2 + \frac{167}{60}(1,2-1) + 2 = 2,5504;$$

$$s_1(2,9) = \frac{83}{120}(2,9-2)^3 - \frac{47}{20}(2,9-2)^2 + \frac{13}{30}(2,9-2) + 4 = 2,9907;$$

$$s_2(5,2) = -\frac{29}{60}(5,2-4)^3 + \frac{9}{5}(5,2-4)^2 - \frac{2}{3}(5,2-4) + 1 = 1,9568;$$

$$s_3(6,7) = \frac{11}{30}(6,7-6)^3 - \frac{11}{10}(6,7-6)^2 + \frac{11}{15}(6,7-6) + 3 = 3,1001.$$

A Figura 3.6 mostra os quatro *splines* cúbicos naturais. Os pares (x_i, y_i) são representados por pontos \circ , os *splines* $s_0(x)$ e $s_2(x)$ são esboçados por uma linha tracejada $--$, $s_1(x)$ e $s_3(x)$ por uma linha sólida $-$, enquanto os valores interpolados são representados por Δ . Os *splines* $s_i(x)$ foram esboçados além de seus intervalos de utilização $[x_i, x_{i+1}]$, apenas para visualizar os seus comportamentos. ■

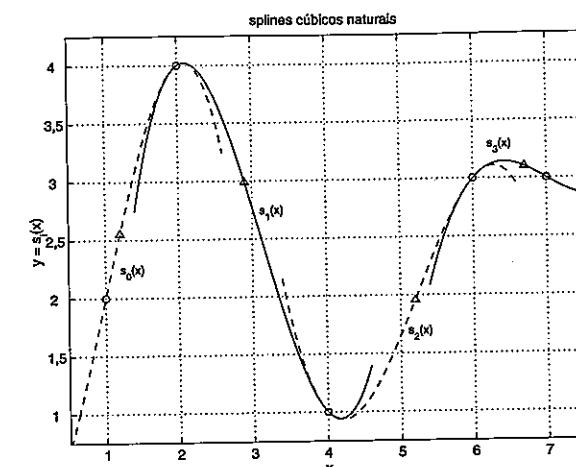


Figura 3.6 Splines cúbicos naturais.

3.9.2 Algoritmo e complexidade

A Figura 3.7 mostra um algoritmo para calcular as derivadas $s_i''(x_i)$ dos *splines* cúbicos naturais. Os parâmetros de entrada são o número de pontos dados n , o vetor x com as abscissas x_i em ordem crescente e o vetor y das ordenadas y_i . Os parâmetros de saída são o vetor solução $s2$ contendo as segundas derivadas e a condição de erro *CondErro*. A condição *CondErro* = 1 significa que o número de pontos $n < 3$, situação em que o algoritmo não pode ser executado. São utilizados dois vetores de trabalho d e e , de dimensões $n-2$, para armazenarem a diagonal e a subdiagonal do sistema linear tridiagonal simétrico. Para economizar espaço de memória o vetor de termos independentes é armazenado, temporariamente, no vetor solução $s2$.

Apesar de o sistema linear ser tridiagonal e simétrico, o algoritmo é baseado no método de eliminação de Gauss. Não se faz necessário o uso da estratégia da pivotação parcial porque a matriz é diagonal estritamente dominante, fazendo com que os pivôs já estejam na diagonal principal.

A Tabela 3.6 mostra a complexidade do algoritmo da Figura 3.7. Como pode ser observado, o algoritmo tem complexidade linear para todas as operações aritméticas.

Tabela 3.6 Complexidade da construção de *splines* cúbicos naturais.

(n : número de pontos, $n > 3$)

| Operações | Complexidade |
|----------------|--------------|
| adições | $20n - 47$ |
| multiplicações | $5n - 13$ |
| divisões | $3n - 7$ |

Algoritmo Splines naturais

{ Objetivo: Calcular as segundas derivadas para os splines cúbicos naturais }
 parâmetros de entrada n, x, y
 { número de pontos dados, abscissas em ordem crescente e ordenadas }
 parâmetros de saída $s2, CondErro$
 { segundas derivadas e condição de erro }
 se $n < 3$ então
 CondErro $\leftarrow 1$, abandone
 fimse
 CondErro $\leftarrow 0$
 { construção do sistema tridiagonal simétrico }
 $m \leftarrow n - 2$; $Ha \leftarrow x(2) - x(1)$; $Deltaa \leftarrow (y(2) - y(1))/Ha$
 para $i \leftarrow 1$ até m faça
 $Hb \leftarrow x(i+2) - x(i+1)$
 $Deltaab \leftarrow (y(i+2) - y(i+1))/Hb$
 $e(i) \leftarrow Hb$
 $d(i) \leftarrow 2 * (Ha + Hb)$
 $s2(i+1) \leftarrow 6 * (Deltaab - Deltaa)$
 $Ha \leftarrow Hb$; $Deltaa \leftarrow Deltaab$
 fimpara
 { eliminação de Gauss }
 para $i \leftarrow 2$ até m faça
 $t \leftarrow e(i-1)/d(i-1)$
 $d(i) \leftarrow d(i) - t * e(i-1)$
 $s2(i+1) \leftarrow s2(i+1) - t * s2(i)$
 fimpara
 { solução por substituições retroativas }
 $s2(m+1) \leftarrow s2(m+1)/d(m)$
 para $i \leftarrow m$ até 2 passo -1 faça
 $s2(i) \leftarrow (s2(i) - e(i-1) * s2(i+1))/d(i-1)$
 fimpara
 $s2(1) \leftarrow 0$
 $s2(m+2) \leftarrow 0$
 finalgoritmo

Figura 3.7 Algoritmo para cálculo das derivadas $s_i''(x_i)$ dos splines cúbicos naturais.

Exemplo 3.24 Resolver o Exemplo 3.23 usando o algoritmo da Figura 3.7.

```
% Os parametros de entrada
n = 5
x = 1    2    4    6    7
y = 2    4    1    3    3
% produzem os resultados
s2 = 0   -4.7000  3.6000 -2.2000      0
CondErro = 0
```

3.10 Splines cúbicos extrapolados

Uma outra forma de eliminar duas incógnitas do sistema linear (3.28) é impor a condição

$$s_0'''(x_1) = s_1'''(x_1) \text{ e } s_{n-2}'''(x_{n-1}) = s_{n-1}'''(x_{n-1}), \quad (3.31)$$

sendo $s_i'''(x)$ obtido da derivação de (3.23). Em vista de (3.25)

$$s_i'''(x) = \frac{s_{i+1}''(x_{i+1}) - s_i''(x_i)}{h_i}, \quad i = 0, 1, 2, \dots, n-1. \quad (3.32)$$

3.10.1 Cálculo das derivadas

Considerando em (3.31) que

$$s_0'''(x_1) = s_1'''(x_1),$$

é avaliando em (3.32)

$$\begin{aligned} \frac{s_1''(x_1) - s_0''(x_0)}{h_0} &= \frac{s_2''(x_2) - s_1''(x_1)}{h_1}, \\ s_0''(x_0) &= \frac{(h_0 + h_1)s_1''(x_1) - h_0 s_2''(x_2)}{h_1}. \end{aligned}$$

Do mesmo modo, a partir da condição de (3.31)

$$s_{n-2}'''(x_{n-1}) = s_{n-1}'''(x_{n-1}),$$

tem-se

$$\begin{aligned} \frac{s_{n-1}''(x_{n-1}) - s_{n-2}''(x_{n-2})}{h_{n-2}} &= \frac{s_n''(x_n) - s_{n-1}''(x_{n-1})}{h_{n-1}}, \\ s_n''(x_n) &= \frac{(h_{n-1} + h_{n-2})s_{n-1}''(x_{n-1}) - h_{n-1}s_{n-2}''(x_{n-2})}{h_{n-2}}. \end{aligned}$$

Substituindo o valor acima de $s_0''(x_0)$ na primeira equação de (3.28) e $s_n''(x_n)$ na última, tem-se o sistema linear tridiagonal não simétrico (3.33)

$$\left[\begin{array}{ccccc} \frac{(h_0+h_1)(h_0+2h_1)}{h_1} & \frac{h_1^2-h_0^2}{h_1} & & & \\ h_1 & 2(h_1+h_2) & h_2 & & \\ & h_2 & 2(h_2+h_3) & h_3 & \\ & & & & \\ & & & & \end{array} \right] \begin{bmatrix} s_1''(x_1) \\ s_2''(x_2) \\ \vdots \\ s_{n-2}''(x_{n-2}) \\ s_{n-1}''(x_{n-1}) \end{bmatrix} = \left[\begin{array}{c} \\ \\ \\ \frac{h_2^2-h_1^2}{h_1} \\ \frac{(h_{n-1}+h_{n-2})(h_{n-1}+2h_{n-2})}{h_{n-2}} \end{array} \right] \quad (3.33)$$

$$= 6 \begin{bmatrix} \Delta y_1 - \Delta y_0 \\ \Delta y_2 - \Delta y_1 \\ \Delta y_3 - \Delta y_2 \\ \vdots \\ \Delta y_{n-1} - \Delta y_{n-2} \end{bmatrix},$$

a partir do qual obtém-se as derivadas $s''_i(x_i)$, $i = 1, 2, 3, \dots, n - 1$. As derivadas $s''_0(x_0)$ e $s''_n(x_n)$ são dadas pelas expressões deduzidas acima

$$\left. \begin{aligned} s''_0(x_0) &= \frac{(h_0 + h_1)s''_1(x_1) - h_0 s''_2(x_2)}{h_1}, \\ s''_n(x_n) &= \frac{(h_{n-1} + h_{n-2})s''_{n-1}(x_{n-1}) - h_{n-1} s''_{n-2}(x_{n-2})}{h_{n-2}} \end{aligned} \right\} \quad (3.34)$$

A derivada $s''_0(x_0)$ é uma interpolação linear de $s''_1(x_1)$ e $s''_2(x_2)$, enquanto $s''_n(x_n)$ é uma interpolação linear de $s''_{n-2}(x_{n-2})$ e $s''_{n-1}(x_{n-1})$. Com essas condições, os splines cúbicos se ajustam perfeitamente a $y = f(x)$ quando a função $f(x)$ é cúbica. Os splines cúbicos extrapolados devem ser usados quando $y = f(x)$ apresentar um ponto de inflexão nas proximidades dos pontos finais x_0 e x_n . Estes splines são também conhecidos como *not-a-knot*.

Exemplo 3.25 Dados os pontos $(1, 2)$, $(2, 4)$, $(4, 1)$, $(6, 3)$ e $(7, 3)$, calcular as segundas derivadas $s''_i(x_i)$, $i = 0, 1, 2, 3, 4$ dos splines cúbicos extrapolados.

Por (3.20)

$$\begin{aligned} h_0 &= x_1 - x_0 = 2 - 1 \rightsquigarrow h_0 = 1, \quad h_1 = x_2 - x_1 = 4 - 2 \rightsquigarrow h_1 = 2, \\ h_2 &= x_3 - x_2 = 6 - 4 \rightsquigarrow h_2 = 2, \quad h_3 = x_4 - x_3 = 7 - 6 \rightsquigarrow h_3 = 1. \end{aligned}$$

Usando (3.27)

$$\begin{aligned} \Delta y_0 &= \frac{y_1 - y_0}{h_0} = \frac{4 - 2}{1} \rightsquigarrow \Delta y_0 = 2, \quad \Delta y_1 = \frac{y_2 - y_1}{h_1} = \frac{1 - 4}{2} \rightsquigarrow \Delta y_1 = -1,5; \\ \Delta y_2 &= \frac{y_3 - y_2}{h_2} = \frac{3 - 1}{2} \rightsquigarrow \Delta y_2 = 1, \quad \Delta y_3 = \frac{y_4 - y_3}{h_3} = \frac{3 - 3}{1} \rightsquigarrow \Delta y_3 = 0. \end{aligned}$$

Substituindo os valores acima em (3.33), tem-se

$$\begin{bmatrix} \frac{(1+2)(1+2 \times 2)}{2} & \frac{2^2 - 1^2}{2} & 0 \\ 2 & 2(2+2) & 2 \\ 0 & \frac{2^2 - 1^2}{2} & \frac{(1+2)(1+2 \times 2)}{2} \end{bmatrix} \begin{bmatrix} s''_1(x_1) \\ s''_2(x_2) \\ s''_3(x_3) \end{bmatrix} = 6 \begin{bmatrix} -1,5 - 2 \\ 1 - (-1,5) \\ 0 - 1 \end{bmatrix} \rightsquigarrow$$

$$s''(x) = \begin{bmatrix} -41/12 \\ 37/12 \\ -17/12 \end{bmatrix}.$$

Por (3.34)

$$\begin{aligned} s''_0(x_0) &= \frac{(1+2) \times -41/12 - 1 \times 37/12}{2} = -20/3; \\ s''_4(x_4) &= \frac{(1+2) \times -17/12 - 1 \times 37/12}{2} = -11/3. \end{aligned}$$

Portanto, as segundas derivadas são

$$s''_0(x_0) = -\frac{20}{3}; \quad s''_1(x_1) = -\frac{41}{12}; \quad s''_2(x_2) = \frac{37}{12}; \quad s''_3(x_3) = -\frac{17}{12}; \quad s''_4(x_4) = -\frac{11}{3}. \quad \blacksquare$$

Exemplo 3.26 A partir dos pontos do Exemplo 3.25, determinar as equações dos quatro splines cúbicos extrapolados na forma (3.14).

Determinação do spline $s_0(x)$

$$a_0 = \frac{s''_1(x_1) - s''_0(x_0)}{6h_0} = \frac{-41/12 - (-20/3)}{6 \times 1} \rightsquigarrow a_0 = \frac{13}{24};$$

$$b_0 = \frac{s''_0(x_0)}{2} = \frac{-20/3}{2} \rightsquigarrow b_0 = -\frac{10}{3};$$

$$c_0 = \Delta y_0 - \frac{s''_1(x_1) + 2s''_0(x_0)}{6} h_0 = 2 - \frac{-41/12 + 2 \times -20/3}{6} \times 1 \rightsquigarrow c_0 = \frac{115}{24};$$

$$d_0 = y_0 \rightsquigarrow d_0 = 2,$$

$$s_0(x) = \frac{13}{24}(x-1)^3 - \frac{10}{3}(x-1)^2 + \frac{115}{24}(x-1) + 2.$$

Determinação do spline $s_1(x)$

$$a_1 = \frac{s''_2(x_2) - s''_1(x_1)}{6h_1} = \frac{37/12 - (-41/12)}{6 \times 2} \rightsquigarrow a_1 = \frac{13}{24};$$

$$b_1 = \frac{s''_1(x_1)}{2} = \frac{-41/12}{2} \rightsquigarrow b_1 = -\frac{41}{24};$$

$$c_1 = \Delta y_1 - \frac{s''_2(x_2) + 2s''_1(x_1)}{6} h_1 = -1,5 - \frac{37/12 + 2 \times -41/12}{6} \times 2 \rightsquigarrow c_1 = -\frac{1}{4};$$

$$d_1 = y_1 \rightsquigarrow d_1 = 4,$$

$$s_1(x) = \frac{13}{24}(x-2)^3 - \frac{41}{24}(x-2)^2 - \frac{1}{4}(x-2) + 4.$$

Determinação do spline $s_2(x)$

$$a_2 = \frac{s''_3(x_3) - s''_2(x_2)}{6h_2} = \frac{-17/12 - 37/12}{6 \times 2} \rightsquigarrow a_2 = -\frac{3}{8};$$

$$b_2 = \frac{s''_2(x_2)}{2} = \frac{37/12}{2} \rightsquigarrow b_2 = \frac{37}{24};$$

$$c_2 = \Delta y_2 - \frac{s''_3(x_3) + 2s''_2(x_2)}{6} h_2 = 1 - \frac{-17/12 + 2 \times 37/12}{6} \times 2 \rightsquigarrow c_2 = -\frac{7}{12};$$

$$d_2 = y_2 \rightsquigarrow d_2 = 1,$$

$$s_2(x) = -\frac{3}{8}(x-4)^3 + \frac{37}{24}(x-4)^2 - \frac{7}{12}(x-4) + 1.$$

Determinação do spline $s_3(x)$

$$a_3 = \frac{s''_4(x_4) - s''_3(x_3)}{6h_3} = \frac{-11/3 - (-17/12)}{6 \times 1} \rightsquigarrow a_3 = -\frac{3}{8};$$

$$b_3 = \frac{s''_3(x_3)}{2} = \frac{-17/12}{2} \rightsquigarrow b_3 = -\frac{17}{24};$$

$$c_3 = \Delta y_3 - \frac{s''_4(x_4) + 2s''_3(x_3)}{6} h_3 = 0 - \frac{-11/3 + 2 \times -17/12}{6} \times 1 \rightsquigarrow c_3 = \frac{13}{12};$$

$$d_3 = y_3 \rightsquigarrow d_3 = 3,$$

$$s_3(x) = -\frac{3}{8}(x-6)^3 - \frac{17}{24}(x-6)^2 + \frac{13}{12}(x-6) + 3.$$

As derivadas dos splines extrapolados são

$$s'_0(x) = \frac{13}{8}(x-1)^2 - \frac{20}{3}(x-1) + \frac{115}{24}; \quad s''_0(x) = \frac{13}{4}(x-1) - \frac{20}{3} \text{ e } s'''_0(x) = \frac{13}{4},$$

$$s'_1(x) = \frac{13}{8}(x-2)^2 - \frac{41}{12}(x-2) - \frac{1}{4}; \quad s''_1(x) = \frac{13}{4}(x-2) - \frac{41}{12} \text{ e } s'''_1(x) = \frac{13}{4},$$

$$s'_2(x) = -\frac{9}{8}(x-4)^2 + \frac{37}{12}(x-4) - \frac{7}{12}; \quad s''_2(x) = -\frac{9}{4}(x-4) + \frac{37}{12} \text{ e } s'''_2(x) = -\frac{9}{4},$$

$$s'_3(x) = -\frac{9}{8}(x-6)^2 - \frac{17}{12}(x-6) + \frac{13}{12}; \quad s''_3(x) = -\frac{9}{4}(x-6) - \frac{17}{12} \text{ e } s'''_3(x) = -\frac{9}{4}.$$

Pela condição (3.16), os splines são contínuos

$$s_i(x_{i+1}) = s_{i+1}(x_{i+1}); \quad s_0(2) = s_1(2) = 4; \quad s_1(4) = s_2(4) = 1 \text{ e } s_2(6) = s_3(6) = 3.$$

As primeiras derivadas, pela condição (3.17)

$$s'_i(x_{i+1}) = s'_{i+1}(x_{i+1}); \quad s'_0(2) = s'_1(2) = -\frac{1}{4}; \quad s'_1(4) = s'_2(4) = -\frac{7}{12} \text{ e } s'_2(6) = s'_3(6) = \frac{13}{12},$$

as segundas derivadas, pela condição (3.18)

$$s''_i(x_{i+1}) = s''_{i+1}(x_{i+1}); \quad s''_0(2) = s''_1(2) = -\frac{41}{12}; \quad s''_1(4) = s''_2(4) = \frac{37}{12} \text{ e } s''_2(6) = s''_3(6) = -\frac{17}{12},$$

e as terceiras derivadas, pela condição (3.31)

$$s'''_0(x_1) = s'''_1(x_1); \quad s'''_0(2) = s'''_1(2) = \frac{13}{4},$$

$$s'''_2(x_3) = s'''_3(x_3); \quad s'''_2(6) = s'''_3(6) = -\frac{9}{4},$$

também são contínuas.

Exemplo 3.27 Interpolar os valores $z = 1,2; 2,9; 5,2; 6,7$ usando os splines cúbicos extrapolados obtidos no Exemplo 3.26.

$$s_0(1,2) = \frac{13}{24}(1,2-1)^3 - \frac{10}{3}(1,2-1)^2 + \frac{115}{24}(1,2-1) + 2 = 2,8293;$$

$$s_1(2,9) = \frac{13}{24}(2,9-2)^3 - \frac{41}{24}(2,9-2)^2 - \frac{1}{4}(2,9-2) + 4 = 2,7861;$$

$$s_2(5,2) = -\frac{3}{8}(5,2-4)^3 + \frac{37}{24}(5,2-4)^2 - \frac{7}{12}(5,2-4) + 1 = 1,8720;$$

$$s_3(6,7) = -\frac{3}{8}(6,7-6)^3 - \frac{17}{24}(6,7-6)^2 + \frac{13}{12}(6,7-6) + 3 = 3,2826.$$

São apresentados na Figura 3.8 os quatro splines cúbicos extrapolados. Os pares (x_i, y_i) são representados por pontos \circ , os splines $s_0(x)$ e $s_2(x)$ são esboçados por uma linha tracejada $--$, $s_1(x)$ e $s_3(x)$ por uma linha sólida $-$, sendo os valores interpolados representados por Δ . Aqui também os splines $s_i(x)$ foram esboçados além de seus intervalos de utilização $[x_i, x_{i+1}]$ para visualizar os seus comportamentos. ■

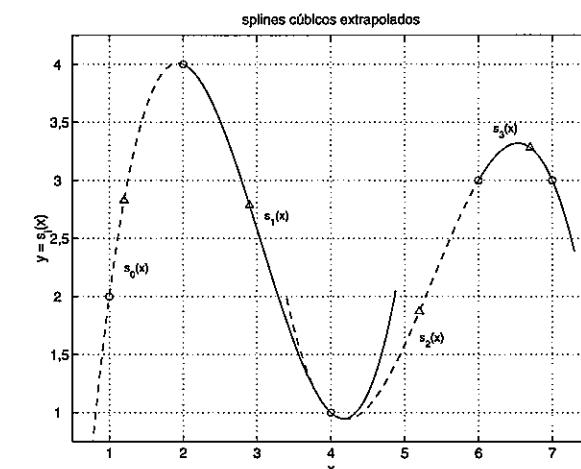


Figura 3.8 Splines cúbicos extrapolados.

3.10.2 Algoritmo e complexidade

Um algoritmo para calcular as derivadas $s''_i(x_i)$ dos splines cúbicos extrapolados é apresentado na Figura 3.9. Os parâmetros de entrada são o número de pontos dados n , o vetor x com as abscissas x_i em ordem crescente e o vetor y das ordenadas y_i . Os parâmetros de saída são o vetor solução $s2$ contendo as segundas derivadas e a condição de erro $CondErro$. A condição $CondErro = 1$ significa que o número de pontos $n < 4$, situação em que o algoritmo não pode ser executado. São usados três vetores de trabalho c , d e e , de dimensões $n-2$, para armazenar as diagonais do sistema linear tridiagonal não simétrico. Para economizar memória o vetor de termos independentes é salvo, temporariamente, no vetor solução $s2$.

```

Algoritmo Splines_extrapolados
{ Objetivo: Calcular as segundas derivadas para os splines cúbicos extrapolados }
parâmetros de entrada  $n, x, y$ 
{ número de pontos dados, abscissas em ordem crescente e ordenadas }
parâmetros de saída  $s2, CondErro$ 
{ segundas derivadas e condição de erro }
se  $n < 4$  então
    CondErro ← 1, abandone
fimse
CondErro ← 0
{ construção do sistema tridiagonal não simétrico }
 $m \leftarrow n - 2$ 
 $Ha \leftarrow x(2) - x(1); Deltaaa \leftarrow (y(2) - y(1))/Ha$ 
 $Hb \leftarrow x(3) - x(2); Deltab \leftarrow (y(3) - y(2))/Hb$ 
 $d(1) \leftarrow (Ha + Hb) * (Ha + 2 * Hb)/Hb$ 
 $c(2) \leftarrow (Hb^2 - Ha^2)/Hb$ 
 $s2(2) \leftarrow 6 * (Deltab - Deltaaa)$ 
para  $i \leftarrow 2$  até  $m - 1$  faça
     $Ha \leftarrow Hb; Deltaaa \leftarrow Deltab$ 
     $Hb \leftarrow x(i+2) - x(i+1)$ 
     $Deltab \leftarrow (y(i+2) - y(i+1))/Hb$ 
     $d(i) \leftarrow 2 * (Ha + Hb)$ 
     $c(i-1) \leftarrow Ha$ 
     $c(i+1) \leftarrow Hb$ 
     $s2(i+1) \leftarrow 6 * (Deltab - Deltaaa)$ 
     $Ha \leftarrow Hb; Deltaaa \leftarrow Deltab$ 
fimpara
 $Ha \leftarrow Hb; Deltaaa \leftarrow Deltab$ 
 $Hb \leftarrow x(n) - x(n-1); Deltab \leftarrow (y(n) - y(n-1))/Hb$ 
 $d(m) \leftarrow (Ha + Hb) * (Hb + 2 * Ha)/Ha$ 
 $c(m-1) \leftarrow (Ha^2 - Hb^2)/Ha$ 
 $s2(m+1) \leftarrow 6 * (Deltab - Deltaaa)$ 
{ eliminação de Gauss }
para  $i \leftarrow 2$  até  $m$  faça
     $t \leftarrow c(i-1)/d(i-1)$ 
     $d(i) \leftarrow d(i) - t * c(i)$ 
     $s2(i+1) \leftarrow s2(i+1) - t * s2(i)$ 
fimpara
{ solução por substituições retroativas }
 $s2(m+1) \leftarrow s2(m+1)/d(m)$ 
para  $i \leftarrow m$  até 2 passo -1 faça
     $s2(i) \leftarrow (s2(i) - c(i) * s2(i+1))/d(i-1)$ 
fimpara
 $Ha \leftarrow x(2) - x(1); Hb \leftarrow x(3) - x(2)$ 
 $s2(1) \leftarrow (((Ha + Hb) * s2(2) - Ha * s2(3))/Hb$ 
 $Ha \leftarrow x(n-1) - x(n-2); Hb \leftarrow x(n) - x(n-1)$ 
 $s2(m+2) \leftarrow (((Ha + Hb) * s2(m+1) - Hb * s2(m))/Hb$ 
finalgoritmo

```

Figura 3.9 Algoritmo para cálculo das derivadas $s_i''(x_i)$ dos splines cúbicos extrapolados.

O sistema linear é tridiagonal não simétrico e o algoritmo é baseado no método de eliminação de Gauss. Nele também não se faz necessário o uso da estratégia da pivotação parcial porque a matriz é diagonal estritamente dominante, fazendo com que os pivôs já estejam na diagonal. A Tabela 3.7 mostra a complexidade do algoritmo da Figura 3.9. O algoritmo tem complexidade linear para todas as operações aritméticas. A operação de potenciação (h^2) foi contabilizada como uma multiplicação.

Tabela 3.7 Complexidade da construção de splines cúbicos extrapolados.

(n : número de pontos, $n > 4$.)

| Operações | Complexidade |
|----------------|--------------|
| adições | $20n - 37$ |
| multiplicações | $5n - 3$ |
| divisões | $3n$ |

Exemplo 3.28 Resolver o Exemplo 3.27 usando o algoritmo da Figura 3.9.

% Os parâmetros de entrada

$n = 5$

$x = 1 \quad 2 \quad 4 \quad 6 \quad 7$

$y = 2 \quad 4 \quad 1 \quad 3 \quad 3$

% produzem os resultados

$s2 = -6.6667 \quad -3.4167 \quad 3.0833 \quad -1.4167 \quad -3.6667$

CondErro = 0

3.11 Avaliação dos splines cúbicos

Uma vez calculadas as derivadas $s_i''(x_i)$, os splines cúbicos da forma (3.14),

$$s_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i, \quad i = 0, 1, 2, \dots, n-1,$$

têm seus coeficientes obtidos a partir de (3.25), (3.24), (3.26) e (3.19)

$$\left. \begin{aligned} a_i &= \frac{s_{i+1}''(x_{i+1}) - s_i''(x_i)}{6h_i}, \\ b_i &= \frac{s_i''(x_i)}{2}, \\ c_i &= \Delta y_i - \frac{s_{i+1}''(x_{i+1}) + 2s_i''(x_i)}{6}h_i, \\ d_i &= y_i, \end{aligned} \right\} i = 0, 1, 2, \dots, n-1,$$

sendo

$$\left. \begin{aligned} h_i &= x_{i+1} - x_i, \\ \Delta y_i &= \frac{y_{i+1} - y_i}{h_i}, \end{aligned} \right\} i = 0, 1, 2, \dots, n-1,$$

dados por (3.20) e (3.27).

3.11.1 Algoritmo e complexidade

Um algoritmo para avaliar os *splines* cúbicos é apresentado na Figura 3.10. Os parâmetros de entrada são o número de pontos dados n , o vetor x com as abscissas x_i em ordem crescente, o vetor y das ordenadas y_i , o número de pontos a interpolar m , o vetor z com os valores a serem interpolados e o tipo de *splines* a ser utilizado ts . Se $ts = 0$, então os *splines* naturais serão usados, e se $ts \neq 0$, então os *splines* extrapolados é que serão. Os parâmetros de saída são o vetor sz contendo os valores interpolados e a condição de erro $CondErro$. A variável $CondErro$ conta o número de vezes em que o valor a ser interpolado $z(j)$ está fora do intervalo $[x(1) x(n)]$, situação em que é atribuído o valor 0 à variável $sz(j)$. O algoritmo utiliza uma pesquisa binária para localizar o intervalo que contenha o valor a ser interpolado e, assim, definir qual *spline* será avaliado.

A Tabela 3.8 apresenta a complexidade do algoritmo para avaliar os *splines*, sem considerar as operações da pesquisa binária.

Tabela 3.8 Complexidade da avaliação de *splines*.

(m : número de pontos a serem avaliados sem considerar as operações da pesquisa binária.)

| Operações | Complexidade |
|----------------|--------------|
| adições | $9m$ |
| multiplicações | $7m$ |
| divisões | $3m$ |

3.11.2 Avaliação

Será mostrado por meio de dois exemplos como utilizar os algoritmos das Figuras 3.7, 3.9 e 3.10 para avaliar os *splines* naturais e extrapolados.

Exemplo 3.29 Dados os pontos $(1, 2), (2, 4), (4, 1), (6, 3)$ e $(7, 3)$, interpolar os valores $z = 1,2; 0,1; 2,9; 5,2$ e $6,7$ usando os *splines* cúbicos naturais.

```
% Os parametros de entrada
n = 5
x = 1    2    4    6    7
y = 2    4    1    3    3
m = 5
z = 1.2000  0.1000  2.9000  5.2000  6.7000
ts = 0
% produzem os resultados
spline  a_i      b_i      c_i      d_i      z_j      sz_j
s_0: -0.7833  0.0000  2.7833  2.0000  1.2000  2.5504
s_1:  0.6917 -2.3500  0.4333  4.0000  2.9000  2.9907
s_2: -0.4833  1.8000 -0.6667  1.0000  5.2000  1.9568
s_3:  0.3667 -1.1000  0.7333  3.0000  6.7000  3.1001
sz = 2.5504      0  2.9907  1.9568  3.1001
CondErro = 1
```

Algoritmo Splines_avaliar

```
{ Objetivo: Avaliar os splines cúbicos naturais e extrapolados }
parâmetros de entrada n, x, y, m, z, ts
{ número de pontos dados, abscissas em ordem crescente, ordenadas, }
{ número de pontos a interpolar, valores a interpolar e tipo de splines }
parâmetros de saída sz, CondErro
{ valores interpolados e condição de erro }

se ts = 0 então
  [sz, CondErro] ← Splines_naturais(n, x, y) (ver Figura 3.7)
senão
  [sz, CondErro] ← Splines_extrapolados(n, x, y) (ver Figura 3.9)
fimse
se CondErro ≠ 0 então abandone, fimse
CondErro ← 0
para j ← 1 até m faça
  se z(j) ≥ x(1) e z(j) ≤ x(n) então
    { pesquisa binária para localizar o intervalo }
    inf ← 1; sup ← n
    repita
      se sup - inf ≤ 1 então interrompa, fimse
      ind ← trunc(((inf + sup)/2))
      se x(ind) > z(j) então
        sup ← ind
      senão
        inf ← ind
      fimse
    fimrepita
    { avaliação do spline pelo processo de Horner }
    h ← x(sup) - x(ind)
    a ← ((s2(sup) - s2(ind)) / (6 * h))
    b ← s2(ind) * 0,5
    c ← (y(sup) - y(ind)) / h - ((s2(sup) + 2 * s2(ind)) * h / 6)
    d ← y(ind)
    h ← z(j) - x(ind)
    sz(j) ← (((a * h + b) * h + c) * h + d)
    escreva inf-1, a, b, c, d, z(j), sz(j)
  senão
    sz(j) ← 0
    CondErro ← CondErro + 1
  fimse
fimpara
finalgoritmo
```

Figura 3.10 Algoritmo para avaliar *splines* cúbicos.

(Ver significado da função trunc na Tabela 1.1, na página 6.)

O resultado $\text{CondErro} = 1$ mostra que um valor a ser interpolado $z(2)$ está fora do intervalo $[1, 7]$, fazendo com que o valor 0 seja atribuído a $\text{sz}(2)$. Comparar os resultados acima com aqueles obtidos no Exemplo 3.23.

Exemplo 3.30 Dados os pontos $(1, 2), (2, 4), (4, 1), (6, 3)$ e $(7, 3)$, interpolar os valores $z = 1,2; 2,9; 0,5; 5,2; 6,7$ e $8,3$ utilizando os *splines* cúbicos extrapolados.

```
% Os parametros de entrada
n = 5
x = 1    2    4    6    7
y = 2    4    1    3    3
m = 6
z = 1.2000  2.9000  0.5000  5.2000  6.7000  8.3000
ts = 1
% produzem os resultados
spline a_i   b_i   c_i   d_i   z_j   sz_j
s_0:  0.5417 -3.3333  4.7917  2.0000  1.2000  2.8293
s_1:  0.5417 -1.7083 -0.2500  4.0000  2.9000  2.7861
s_2: -0.3750  1.5417 -0.5833  1.0000  5.2000  1.8720
s_3: -0.3750 -0.7083  1.0833  3.0000  6.7000  3.2826
sz = 2.8293  2.7861   0  1.8720  3.2826   0
CondErro = 2
```

Neste caso, $\text{CondErro} = 2$ indicando que dois valores a serem interpolados $z(3)$ e $z(6)$ estão fora do intervalo $[1, 7]$, fazendo com que $\text{sz}(3)$ e $\text{sz}(6)$ recebam o valor 0. Comparar estes resultados com aqueles obtidos no Exemplo 3.27.

3.12 Comparação dos *splines* cúbicos

Seja a função

$$f(x) = \begin{cases} e^x, & -2 \leq x \leq 0, \\ x \operatorname{sen}(5x) + 1, & 0 \leq x \leq 4, \end{cases}$$

definida por n pontos distintos $(-2, f(-2)), (-2+h, f(-2+h)), (-2+2h, f(-2+2h)), \dots, (4, f(4))$, sendo $h = (4 - (-2))/(n-1)$. Deseja-se reconstruir a curva de $y = f(x)$ por meio dos *splines* cúbicos, utilizando os algoritmos das Figuras 3.7, 3.9 e 3.10, com $m = 121$ pontos. A Figura 3.11 mostra como os *splines* cúbicos aproximam a curva de $y = f(x)$, dados diferentes números de pontos da função.

A Tabela 3.9 apresenta uma comparação entre os *splines* cúbicos naturais e extrapolados para 6 dos 121 valores de z . Ela compila o módulo da diferença entre o valor interpolado pelo *spline* e o valor exato quando diferentes números de pontos n são fornecidos. Para $n = 7$ pontos nenhum dos dois *splines* apresentou um bom resultado, conforme mostram as Figuras 3.11 (a) e (b). Quando $n = 13$ os *splines* cúbicos naturais foram melhores, como visto nas Figuras 3.11 (c) e (d), enquanto para $n = 25$ os extrapolados apresentaram melhores resultados. Já para $n = 61$, os resultados são praticamente iguais. Obviamente, um único exemplo não é suficiente para definir qual *spline* é mais indicado; no entanto, vale a regra geral: quanto mais pontos, melhor a interpolação.

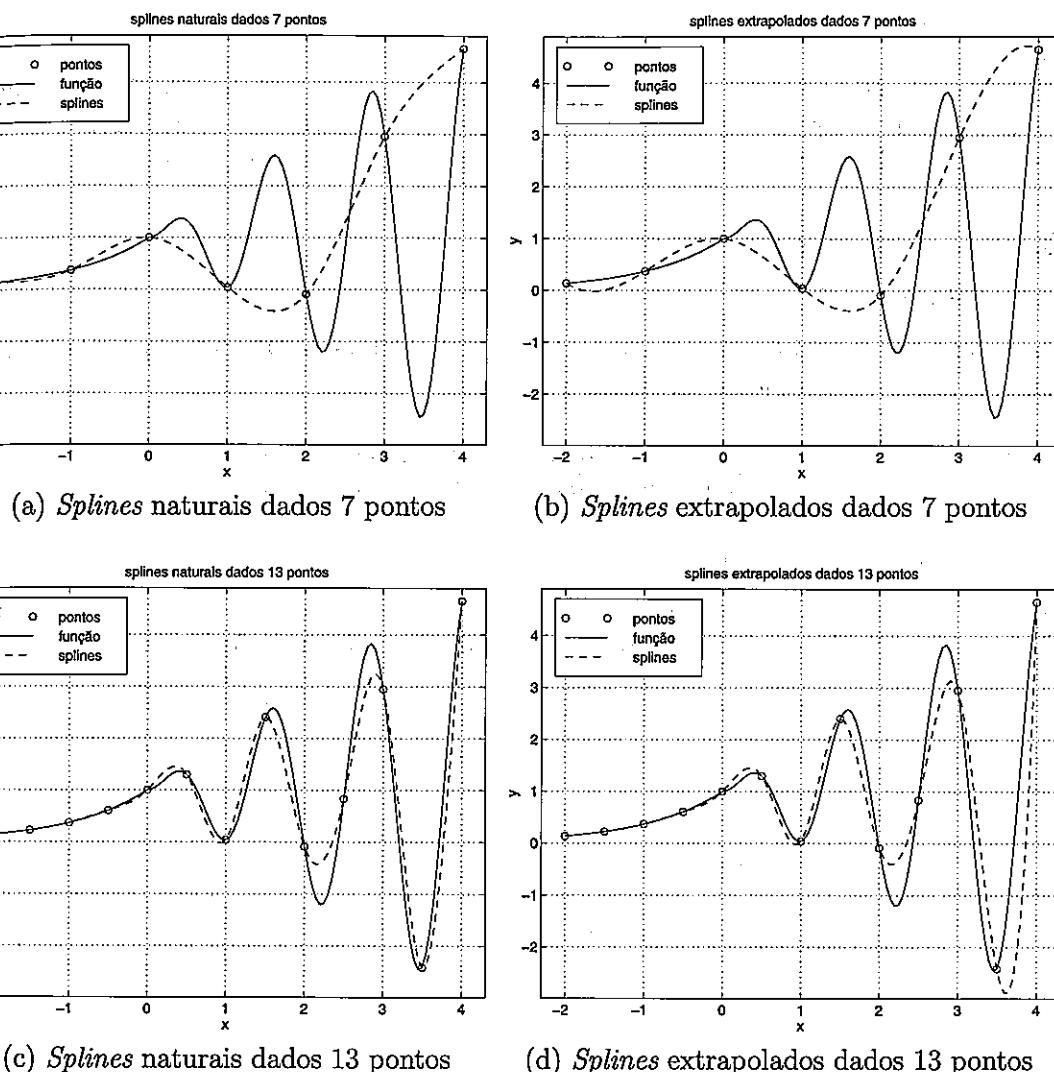


Figura 3.11 *Splines* cúbicos com diferentes números de pontos.

Tabela 3.9 Comparação de *splines* cúbicos, com $m = 121$ pontos interpolados.

| z | $n = 7 (h = 1)$ | | $n = 13 (h = 0,5)$ | | $n = 25 (h = 0,25)$ | | $n = 61 (h = 0,1)$ | |
|-------|-----------------|----------|--------------------|----------|---------------------|----------|--------------------|----------|
| | Natural | Extrapol | Natural | Extrapol | Natural | Extrapol | Natural | Extrapol |
| -1,95 | 0,00625 | 0,05198 | 0,00105 | 0,00189 | 0,00033 | 0,00001 | 0,00006 | 0,00000 |
| -0,95 | 0,01625 | 0,02866 | 0,00216 | 0,00222 | 0,00002 | 0,00003 | 0,00000 | 0,00000 |
| 0,05 | 0,02107 | 0,02496 | 0,06784 | 0,06788 | 0,03382 | 0,03382 | 0,01022 | 0,01022 |
| 1,05 | 0,11802 | 0,11486 | 0,09564 | 0,09614 | 0,00626 | 0,00626 | 0,00023 | 0,00023 |
| 2,05 | 0,51399 | 0,50526 | 0,23972 | 0,24657 | 0,00956 | 0,00956 | 0,00016 | 0,00016 |
| 3,05 | 0,73943 | 0,77121 | 0,20128 | 0,29666 | 0,00100 | 0,00062 | 0,00036 | 0,00036 |

3.13 Exemplos de aplicação

Serão apresentados dois problemas que podem ser resolvidos por interpolação. O primeiro é como determinar o ponto de equivalência de uma curva de titulação ácido-base, e o segundo mostra o uso de interpolação inversa para calcular a raiz de uma equação.

3.13.1 Curva de titulação

Definição do problema

Determinar o ponto de equivalência da titulação de 100 ml de HCl 0,1 N por NaOH 0,1 N, cujo resultado é apresentado na tabela a seguir

| | | | | | | | | |
|----------|------|------|------|------|-------|-------|-------|-------|
| Vol (ml) | 50,0 | 80,0 | 98,0 | 99,8 | 100,2 | 102,0 | 120,0 | 150,0 |
| pH | 1,48 | 1,95 | 3,00 | 4,00 | 10,00 | 11,00 | 11,96 | 12,30 |

Modelagem matemática

A Figura 3.12(a) mostra a curva de titulação ácido-base construída a partir da tabela acima. O ponto de equivalência é um ponto de inflexão da curva de titulação Vol × pH. Conseqüen-

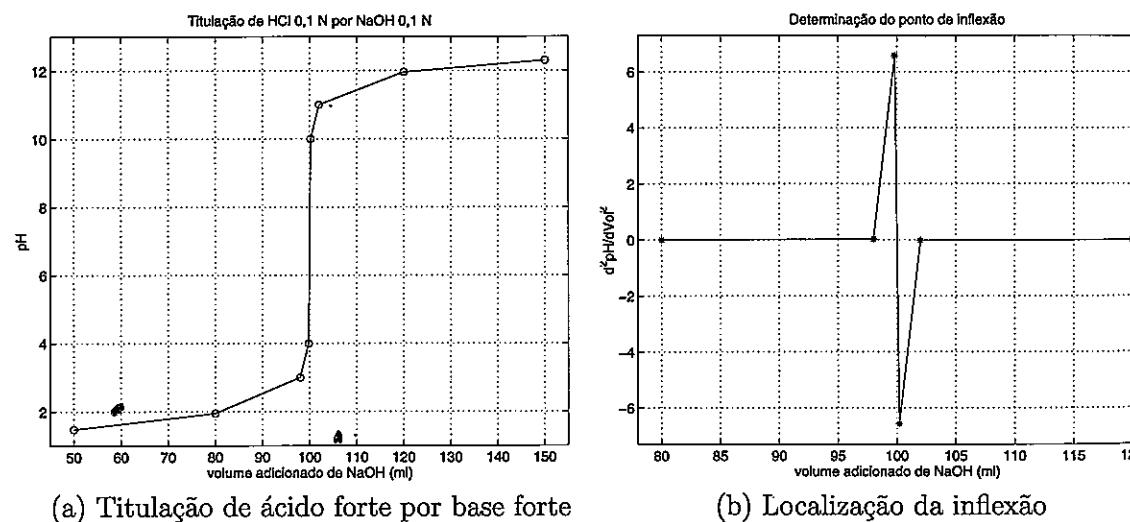


Figura 3.12 Determinação do ponto de equivalência de titulação ácido-base.

tamente, a região em torno da inflexão pode ser aproximada por um polinômio interpolador $P_n(x)$. Como a inflexão de uma curva é o ponto onde a derivada segunda da função se anula, então uma raiz de $P''(x) = 0$ fornece uma aproximação do ponto de equivalência procurado. Deve ser utilizado um polinômio interpolador de Newton ou Lagrange, pois a variação de volume não é constante. Será usado um polinômio de Newton de grau 3, pois sua derivada segunda será de grau 1

$$P_3(x) = y_0 + \Delta y_0(x - x_0) + \Delta^2 y_0(x - x_0)(x - x_1) + \Delta^3 y_0(x - x_0)(x - x_1)(x - x_2).$$

Também poderia ser utilizado um polinômio de grau superior para uma maior exatidão; no entanto, isto acarretaria o cálculo de raiz de equação algébrica que será abordado somente no Capítulo 6. O polinômio $P_3(x)$ pode ser escrito na forma de potências

$$P_3(x) = a_3 x^3 + a_2 x^2 + a_1 x + a_0,$$

sendo

$$a_3 = \Delta^3 y_0,$$

$$a_2 = \Delta^2 y_0 - \Delta^3 y_0(x_0 + x_1 + x_2),$$

$$a_1 = \Delta y_0 - \Delta^2 y_0(x_0 + x_1) + \Delta^3 y_0(x_0 x_1 + x_0 x_2 + x_1 x_2) \text{ e}$$

$$a_0 = y_0 - \Delta y_0(x_0) + \Delta^2 y_0(x_0 x_1) - \Delta^3 y_0(x_0 x_1 x_2).$$

Como a derivada segunda é

$$P''_3(x) = P_1(x) = 6a_3 x + 2a_2,$$

então a abscissa x_{eq} , onde a derivada se anula, é

$$P_1(x_{eq}) = 6a_3 x_{eq} + 2a_2 = 0 \rightarrow x_{eq} = -\frac{a_2}{3a_3} \quad (3.35)$$

e a ordenada do ponto de equivalência é

$$P_3(x_{eq}) = a_3 x_{eq}^3 + a_2 x_{eq}^2 + a_1 x_{eq} + a_0. \quad (3.36)$$

Solução numérica

A tabela de diferenças divididas de $x_i = \text{Vol}_i$ e $y_i = \text{pH}_i$ é mostrada a seguir

| i | x_i | y_i | Δy_i | $\Delta^2 y_i$ | $\Delta^3 y_i$ |
|-----|-------|-------|--------------|----------------|----------------|
| 0 | 50,0 | 1,48 | 0,0157 | 0,0009 | 0,0005 |
| 1 | 80,0 | 1,95 | 0,0583 | 0,0251 | 0,3238 |
| 2 | 98,0 | 3,00 | 0,5556 | 6,5656 | -3,2828 |
| 3 | 99,8 | 4,00 | 15,0000 | -6,5656 | 0,3238 |
| 4 | 100,2 | 10,00 | 0,5556 | -0,0254 | 0,0005 |
| 5 | 102,0 | 11,00 | 0,0533 | -0,0009 | |
| 6 | 120,0 | 11,96 | 0,0113 | | |
| 7 | 150,0 | 12,30 | | | |

Considerando que $\frac{d^2y}{dx^2}(x_i) \approx \Delta^2 y_{i-1}$, que $\Delta^2 y_2 > 0$ e $\Delta^2 y_3 < 0$, então a inflexão ocorre entre os pontos $x_3 = 99,8$ e $x_4 = 100,2$, conforme pode ser verificado na Figura 3.12(b). Deste modo, os quatro pontos selecionados para construir o polinômio interpolador $P_3(x)$ são aqueles em torno da inflexão mostrados na tabela a seguir. Nesta tabela, $x_i = \text{Vol}_i - 98$ para reduzir os erros de arredondamento quando da avaliação dos coeficientes de $P_3(x)$.

| i | x_i | y_i | Δy_i | $\Delta^2 y_i$ | $\Delta^3 y_i$ |
|-----|-------|-------|--------------|----------------|----------------|
| 0 | 0,0 | 3,00 | 0,5556 | 6,5656 | -3,2828 |
| 1 | 1,8 | 4,00 | 15,0000 | -6,5656 | 0,3238 |
| 2 | 2,2 | 10,00 | 0,5556 | -0,0254 | 0,0005 |
| 3 | 4,0 | 11,00 | 0,0533 | -0,0009 | |

Os coeficientes de $P_3(x)$ são

$$a_3 = \Delta^3 y_0 \rightarrow a_3 = -3,2828,$$

$$a_2 = \Delta^2 y_0 - \Delta^3 y_0(x_0 + x_1 + x_2)$$

$$= 6,5656 + 3,2828(0,0 + 1,8 + 2,2) \rightarrow a_2 = 19,6968,$$

$$a_1 = \Delta y_0 - \Delta^2 y_0(x_0 + x_1) + \Delta^3 y_0(x_0 x_1 + x_0 x_2 + x_1 x_2)$$

$$= 0,5556 - 6,5656(0,0 + 1,8) - 3,2828((0,0)(1,8) + (0,0)(2,2) + (1,8)(2,2))$$

$$\rightarrow a_1 = -24,2624 \text{ e}$$

$$a_0 = y_0 - \Delta y_0(x_0) + \Delta^2 y_0(x_0 x_1) - \Delta^3 y_0(x_0 x_1 x_2)$$

$$= 3,00 - 0,5556(0,0) + 6,5656(0,0)(1,8) + 3,2828(0,0)(1,8)(2,2)$$

$$\rightarrow a_0 = 3,0000.$$

Por (3.35), o ponto de equivalência é

$$x_{eq} = -\frac{a_2}{3a_3} = -\frac{19,6968}{3(-3,2828)} \rightarrow x_{eq} = 2,0000.$$

O pH no ponto de equivalência é dado por (3.36)

$$P_3(x_{eq}) = a_3 x_{eq}^3 + a_2 x_{eq}^2 + a_1 x_{eq} + a_0,$$

$$P_3(2) = -3,2828(2)^3 + 19,6968(2)^2 - 24,2624(2) + 3,0000 \rightarrow P_3(2) = 7,0000.$$

Como os valores das abscissas foram reduzidos por 98, então o ponto de equivalência, de fato, é

$$\text{Vol}_{eq} = 2,0000 + 98 = 100,0000 \text{ ml.}$$

Análise dos resultados

É sabido da Química Analítica que, teoricamente, em uma titulação de 100 ml de HCl 0,1 N com NaOH 0,1 N, o volume de equivalência é igual a 100 ml e o pH = 7.

3.13.2 Interpolação inversa

Definição do problema

Calcular uma aproximação da raiz ψ de $f(x) = 0,05x^3 - 0,4x^2 + 3 \operatorname{sen}(x)x = 0$, sabendo que $11 \leq \psi \leq 12$.

Modelagem matemática

Este problema pode ser resolvido por interpolação inversa quadrática, que consiste em aproximar a função inversa $f^{-1}(x)$ por um polinômio de grau 2, tomando $f(x)$ como abscissa e x como ordenada, conforme a tabela a seguir

| i | 0 | 1 | 2 |
|----------|----------|----------|----------|
| $f(x_i)$ | $f(x_0)$ | $f(x_1)$ | $f(x_2)$ |
| x_i | x_0 | x_1 | x_2 |

A raiz ψ de $f(x) = 0$ é tal que $f^{-1}(0) = \psi$.

Solução numérica

Como são necessários $m = 3$ pontos para construir um polinômio de grau $n = 2$ e $11 \leq \psi \leq 12$, então constrói-se a tabela

| i | 0 | 1 | 2 |
|----------|----------|---------|--------|
| $f(x_i)$ | -14,8497 | -7,0594 | 9,4834 |
| x_i | 11,0 | 11,5 | 12,0 |

Utilizando esses dados no algoritmo Polinômio_Newton da Figura 3.3

$m = 3$
 $x = -14,8497 \quad -7,0594 \quad 9,4834$
 $y = 11,0000 \quad 11,5000 \quad 12,0000$
 $z = 0$

Obtém-se o resultado

$$r = 11,8068$$

Desse modo, $\psi \approx 11,8068$, sendo $f(11,8068) = 2,1426$. Embora esse valor seja mais próximo de zero do que os valores da tabela acima, uma melhor aproximação pode ser obtida substituindo $(-14,8497; 11,0)$ pelo par $(2,1426; 11,8068)$, visto ser este o ponto da ordenada mais distante de zero. A nova tabela será

| i | 0 | 1 | 2 |
|----------|---------|---------|--------|
| $f(x_i)$ | -7,0594 | 2,1426 | 9,4834 |
| x_i | 11,5 | 11,8068 | 12,0 |

Usando esses dados no algoritmo Polinômio_Newton da Figura 3.3, obtém-se o resultado

```
m = 3
x = -7.0594  2.1426  9.4834
y = 11.5000  11.8068 12.0000
z = 0
r = 11.7418
```

Assim, uma melhor aproximação é $\psi \approx 11,7418$. De fato, $f(11,7418) = -0,0704$. Repetindo mais uma vez o processo, tem-se

```
m = 3
x = -7.0594  -0.0704  2.1426
y = 11.5000  11.7418  11.8068
z = 0
r = 11.7440
```

Esta é, sem dúvida, a melhor das três aproximações, visto que $f(11,7440) = 0,0023$.

Análise dos resultados

Usando a interpolação inversa quadrática foi possível obter as coordenadas $(11,8068; 2,1426)$, $(11,7418; -0,0704)$ e $(11,7440; 0,0023)$, que cada vez mais se aproximam da raiz ($\psi f(\psi)$).

O robusto e eficiente método de van Wijngaarden-Dekker-Brent para cálculo de raiz de equação, visto na Seção 6.4.2, é baseado em interpolação inversa quadrática (ver Exemplo 6.29, na página 306).

3.14 Exercícios

Seção 3.1

Calcular a partir da tabela de $y = \sin(x)$

| x | y |
|-----|--------|
| 0,3 | 0,2955 |
| 0,4 | 0,3894 |
| 0,5 | 0,4794 |

os valores de

3.1. $P_1(0,33)$.

3.2. $P_2(0,33)$.

3.3. $P_1(0,38)$.

3.4. $P_2(0,38)$.

3.5. Comparar cada valor interpolado acima com o resultado exato.

Seção 3.2

Seja a tabela

| x | y |
|-----|--------|
| 1,0 | 0,8415 |
| 1,3 | 1,2526 |
| 1,7 | 1,6858 |
| 2,0 | 1,8186 |

3.6. Calcular $L_1(1,1)$.

3.7. Avaliar $L_2(1,1)$ por (3.5).

3.8. Calcular $L_2(1,1)$ por (3.6).

3.9. Estimar $L_3(1,2)$.

3.10. Implementar o algoritmo do polinômio de Lagrange apresentado na Figura 3.2, utilizando qualquer linguagem de programação, e resolver os Exercícios 3.8–3.9 por meio do programa.

Seção 3.3

Considere a tabela

| x | y |
|-----|--------|
| 2,0 | 0,9803 |
| 2,2 | 1,1695 |
| 2,4 | 1,3563 |
| 2,5 | 1,4488 |
| 2,7 | 1,6321 |
| 2,9 | 1,8131 |

3.11. Calcular $P_1(2,1)$, $P_2(2,1)$ e $P_3(2,1)$ por intermédio de (3.10).

3.12. Comparar os três resultados acima com o valor exato $f(2,1) = 1,0752$.

3.13. Implementar, usando qualquer linguagem de programação, o algoritmo do polinômio de Newton dado na Figura 3.3.

3.14. Resolver o Exercício 3.11 usando o programa escrito no Exercício 3.13.

3.15. Mostrar que o polinômio $P_1(x)$ de Newton é igual ao $L_1(x)$ de Lagrange.

Seção 3.4

Seja a tabela

| x | y |
|-----|--------|
| 2,1 | 0,3693 |
| 2,2 | 0,5137 |
| 2,3 | 0,6732 |
| 2,4 | 0,8424 |

3.16. Construir a tabela de diferenças finitas ascendentes e a de diferenças divididas e verificar a relação (3.11).

3.17. Calcular $P_1(2,15)$, $P_2(2,15)$ e $P_3(2,15)$ por meio de (3.12) e comparar com o resultado exato $f(2,15) = 0,4393$.

3.18. Implementar o algoritmo do polinômio de Gregory-Newton mostrado na Figura 3.4 em qualquer linguagem de programação.

3.19. Resolver o Exercício 3.17 usando o programa escrito no Exercício 3.18.

3.20. Mostrar que $P_1(x)$ de Gregory-Newton é igual ao $L_1(x)$ de Lagrange.

Seção 3.5

Seja a tabela

| x | y |
|-----|--------|
| 1,0 | 0,8415 |
| 1,1 | 0,9803 |
| 1,3 | 1,2526 |
| 1,4 | 1,3796 |
| 1,7 | 1,6858 |
| 1,8 | 1,7529 |
| 2,0 | 1,8186 |

Escolher as abscissas dos pontos para calcular

3.21. $L_1(1,11)$.

3.22. $L_2(1,35)$.

3.23. $L_3(1,5)$.

3.24. $P_2(1,6)$.

3.25. $P_2(1,9)$.

Seção 3.6

Considere a tabela da função $f(x) = x\sqrt{x}$

| x | f(x) |
|-----|---------|
| 2,0 | 2,8284 |
| 2,5 | 3,9528 |
| 3,2 | 5,7243 |
| 3,9 | 7,7019 |
| 4,1 | 8,3019 |
| 5,0 | 11,1803 |

3.26. Calcular $P_2(3,5)$, utilizando os pontos de abscissas $x = 2,5, 3,2$ e $3,9$.

3.27. Avaliar $T_2(3,5)$ usando os três pontos do Exercício 3.26.

3.28. Calcular $P_2(3,5)$ utilizando os pontos de abscissas $x = 3,2, 3,9$ e $4,1$.

3.29. Avaliar $T_2(3,5)$ usando os três pontos do Exercício 3.28.

3.30. Comparar os resultados acima com o valor exato $f(3,5) = 6,5479$.

Seção 3.9

3.31. Mostrar que a substituição dos valores das derivadas $s''_0(x_0) = 0$ e $s''_n(x_n) = 0$ no sistema linear subdeterminado (3.28) resulta no sistema tridiagonal simétrico (3.30).

3.32. Resolver o sistema linear do Exemplo 3.21 usando decomposição de Cholesky e comparar o resultado com o vetor $s''(x)$.

3.33. Achar a solução do sistema linear do Exemplo 3.21 usando eliminação de Gauss, sem pivotação parcial, e comparar o resultado com o vetor $s''(x)$.

3.34. Implementar o algoritmo para construção dos splines cúbicos naturais apresentado na Figura 3.7 em uma linguagem de programação.

3.35. Calcular as derivadas $s''_i(x_i)$ do Exemplo 3.21 usando o programa implementado acima. Listar os valores intermediários dos vetores d e $s2$ nas etapas *eliminação de Gauss* e *solução por substituições retroativas* e comparar com os valores calculados no Exercício 3.33.

Seção 3.10

3.36. Substituir os valores das derivadas $s''_0(x_0)$ e $s''_n(x_n)$, dadas por (3.34), no sistema linear subdeterminado (3.28) e mostrar que resulta no sistema tridiagonal não simétrico (3.33).

3.37. Calcular a solução do sistema linear do Exemplo 3.25 usando a eliminação de Gauss, sem pivotação parcial, e comparar o resultado com o vetor $s''(x)$.

3.38. Implementar o algoritmo para obtenção dos splines cúbicos extrapolados dado na Figura 3.9 usando qualquer linguagem de programação.

3.39. Utilizando o programa implementado acima, achar as derivadas $s''_i(x_i)$ do Exemplo 3.25. Exhibir os valores intermediários dos vetores d e $s2$ nas etapas *eliminação de Gauss* e *solução por substituições retroativas* e comparar com os valores obtidos no Exercício 3.37.

3.40. Fazer um estudo comparativo entre as complexidades computacionais dos splines naturais e extrapolados baseado nos valores compilados nas Tabelas 3.6 e 3.7.

Seção 3.11

3.41. Implementar, usando qualquer linguagem de programação, o algoritmo para avaliação dos splines cúbicos dado na Figura 3.10.

Seja a tabela da função $f(x) = \sqrt{x}$

| x | f(x) |
|---|--------|
| 1 | 1 |
| 2 | 1,4142 |
| 4 | 2 |
| 5 | 2,2361 |
| 8 | 2,8284 |
| 9 | 3 |

3.42. Dados $z = [1,1; 2,2; 4,3; 5,7; 8,8; 8,9]$, interpolar usando o programa acima, com splines cúbicos naturais, e comparar com os valores exatos.

3.43. Dados $z = [1,1; 2,2; 4,3; 5,7; 8,8; 8,9]$, interpolar usando o programa acima, com splines cúbicos extrapolados. Comparar com os valores exatos e com os resultados obtidos pelos splines cúbicos naturais.

Seja a função definida na Seção 3.12

$$f(x) = \begin{cases} e^x, & -2 \leq x \leq 0, \\ x \operatorname{sen}(5x) + 1, & 0 \leq x \leq 4, \end{cases}$$

definida por n pontos distintos $(-2, f(-2)), (-2+h, f(-2+h)), (-2+2h, f(-2+2h)), \dots, (4, f(4))$, sendo $h = (4 - (-2))/(n - 1)$.

3.44. Gerar $n = 7$ pares $(x_i, f(x_i))$ da função e, dado $z = [-1,95; -0,95; 0,05; 1,05; 2,05; 3,05]$, interpolar usando o programa acima com splines cúbicos naturais. Verificar a diferença com o valor exato e comparar com os resultados da Tabela 3.9.

3.45. Dados o vetor z do exercício anterior e $n = 13$ pares $(x_i, f(x_i))$ da função, interpolar utilizando o programa do Exercício 3.41 com splines cúbicos extrapolados. Verificar a diferença com o valor exato e comparar com os resultados da Tabela 3.9.

Gerais

3.46. A tabela a seguir relaciona o calor específico da água com a temperatura [30, Tabela A.3(b)]

| t, °C | c_p , kcal/(kg·°C) |
|-------|----------------------|
| 200 | 1,075 |
| 220 | 1,102 |
| 240 | 1,136 |
| 260 | 1,183 |
| 280 | 1,250 |

Calcular a capacidade calorífica c_p da água à temperatura $t = 250^\circ\text{C}$, por meio de interpolação cúbica.

3.47. Seja a tabela relacionando a temperatura com a densidade do mercúrio (Hg) [30, Tabela A.3(c)]

| t, °C | ρ , g/cm³ |
|-------|----------------|
| -20 | 13,645 |
| 20 | 13,546 |
| 100 | 13,352 |
| 200 | 13,115 |
| 300 | 12,881 |

Determinar a densidade ρ do mercúrio à tempe-

ratura $t = 25^\circ\text{C}$ usando um polinômio interpolador de segundo grau.

3.48. Seja a função de distribuição de probabilidade normal padrão definida por

$$N(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^z \exp\left(-\frac{t^2}{2}\right) dt,$$

cujos valores são mostrados na tabela abaixo [1]

| z | $N(z)$ |
|-----|---------|
| 0,0 | 0,50000 |
| 0,5 | 0,69146 |
| 1,0 | 0,84134 |
| 1,5 | 0,93319 |
| 2,0 | 0,97725 |
| 2,5 | 0,99379 |
| 3,0 | 0,99865 |

a) Calcular $P_n(0,3)$ utilizando polinômios interpoladores de graus $n = 1, 2, 3, 4, 5$ e 6 .

b) Interpolar $z = 0,3$ usando *splines* cúbicos naturais.

c) Utilizando *splines* cúbicos extrapolados, interpolar $z = 0,3$.

d) Comparar os resultados obtidos nos itens (a), (b) e (c) com o valor tabelado $N(0,3) = 0,61791$.

3.49. Por meio de interpolação inversa cúbica e dos valores da tabela do Exercício 3.48, determinar para qual valor de z , $N(z) = 0,8$.

3.50. Usando interpolação inversa quadrática, calcular uma aproximação da raiz de $f(x) = 2x^3 - \cos(x+1) - 3 = 0$, pertencente ao intervalo $[-1, 2]$. Comparar com o resultado do Exemplo 6.24, na página 299.

Capítulo 4

Ajuste de curvas

As relações entre as variáveis envolvidas em um experimento podem ser classificadas em três tipos: determinísticas, semideterminísticas e empíricas. Nas relações determinísticas, as variáveis estão freqüentemente relacionadas entre si por algum tipo de lei que pode ser expressa por intermédio de uma fórmula matemática precisa e qualquer variação nas observações é atribuída a erros experimentais.

Já nas relações semideterminísticas, alguma teoria prescreve uma forma para a relação entre as variáveis, mas não os valores particulares dos parâmetros que aparecem na relação. Então, faz-se necessário realizar experimentos para obter informações acerca desses parâmetros. Quando as relações entre as variáveis envolvidas não são conhecidas, têm-se as chamadas relações empíricas. O que se busca, neste caso, é justamente determinar uma fórmula matemática que relate essas variáveis. Um gráfico feito com valores observados dessas variáveis fornece uma idéia da relação entre elas com algumas variações aleatórias. Somente após ter suficiente conhecimento sobre uma relação empírica é possível desenvolver uma teoria que conduza a uma fórmula matemática, e, portanto, a um caso semideterminístico.

A precisão limitada dos instrumentos de medida, as perturbações incontroláveis das condições experimentais e outros fatores introduzem erros nos dados e normalmente causam alguma perturbação na verdadeira relação. A variação das leituras de uma variável está associada, além dos erros de medida experimentais, a outras variáveis cujos valores se alteram durante um experimento. O que se busca é relacionar, por meio de um modelo matemático, a variável resposta (ou dependente) com o conjunto de variáveis explicativas (ou independentes) para ter controle, determinar algum parâmetro ou mesmo fazer previsão acerca do comportamento da variável resposta.

Existe uma área da Estatística denominada *Análise de regressão* que consiste em um conjunto de métodos, tais como estimativa de parâmetros, análise de variância e de resíduos, testes de hipóteses, que lidam com a formulação de modelos matemáticos que descrevem relações entre

variáveis e o uso desses modelos com o propósito de predição e outras inferências estatísticas [13, 23]. No entanto, uma *análise de regressão* está além do escopo deste texto, que irá se ater somente à determinação de parâmetros de modelos semideterminísticos.

4.1 Regressão linear simples

As relações mais simples entre duas variáveis são as relações lineares. A variável independente ou explicativa x é relacionada com a variável dependente ou resposta y por meio de um modelo linear, por exemplo, $y = b_0 + b_1x$.

4.1.1 Diagrama de dispersão

Uma etapa preliminar importante, ao analisar a relação entre duas variáveis, é esboçar os dados em um gráfico de coordenadas cartesianas denominado diagrama de dispersão. Este diagrama mostra a natureza da relação intrínseca entre as duas variáveis estudadas. Sejam, por exemplo, os dados da Tabela 4.1 relacionando as variáveis x e y .

Tabela 4.1 Variáveis explicativas x e as respostas y .

| | | | | | |
|-----|-----|-----|-----|-----|-----|
| x | 0,3 | 2,7 | 4,5 | 5,9 | 7,8 |
| y | 1,8 | 1,9 | 3,1 | 3,9 | 3,3 |

O diagrama de dispersão dos dados da Tabela 4.1 é mostrado na Figura 4.1. Pode-se observar uma relação aproximadamente linear entre as variáveis explicativas x e as respostas y .

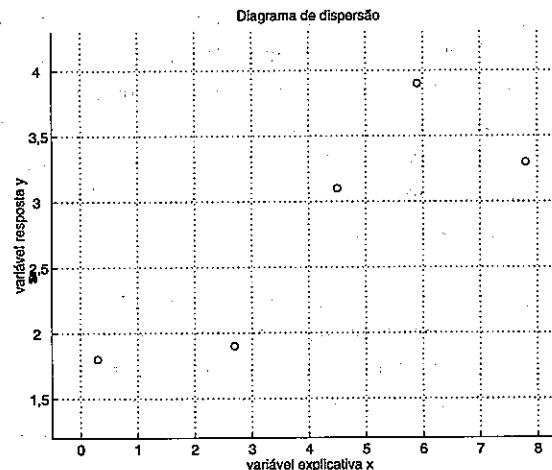


Figura 4.1 Diagrama de dispersão.

4.1.2 Retas de regressão

Um modelo simples que relaciona duas variáveis x e y é

$$y = \beta_0 + \beta_1 x + \epsilon,$$

onde β_0 e β_1 são os parâmetros a serem estimados e ϵ contém os componentes desconhecidos e aleatórios de erro que se sobrepõem à verdadeira relação linear. A questão que se coloca é como estimar os parâmetros β_0 e β_1 .

A primeira tentativa (modelo 1) pode ser obtida por intermédio de um polinômio interpolador linear. Pela Figura 4.1, é possível perceber que não se pode traçar uma única reta que passe por todos os pontos simultaneamente. Por isso, a reta será esboçada a partir de dois pontos quaisquer, por exemplo o primeiro e o último, mostrados a seguir

| | | |
|-----|-----|-----|
| x | 0,3 | 7,8 |
| y | 1,8 | 3,3 |

A equação da reta $u(x)$ que passa por estes dois pontos, por (3.1), é

$$u(x) = y_0 + \frac{y_1 - y_0}{x_1 - x_0} (x - x_0) = 1,8 + \frac{3,3 - 1,8}{7,8 - 0,3} (x - 0,3) = 1,8 + 0,2(x - 0,3) \rightsquigarrow \\ u(x) = 1,74 + 0,2x.$$

A Figura 4.2(a) mostra a reta $u = 1,74 + 0,2x$ traçada entre os pontos do diagrama de dispersão. A distância vertical d_i entre o i -ésimo ponto dado y_i e o ponto $u_i = 1,74 + 0,2x_i$ de mesma abscissa x_i e pertencente à reta é

$$d_i = y_i - u_i.$$

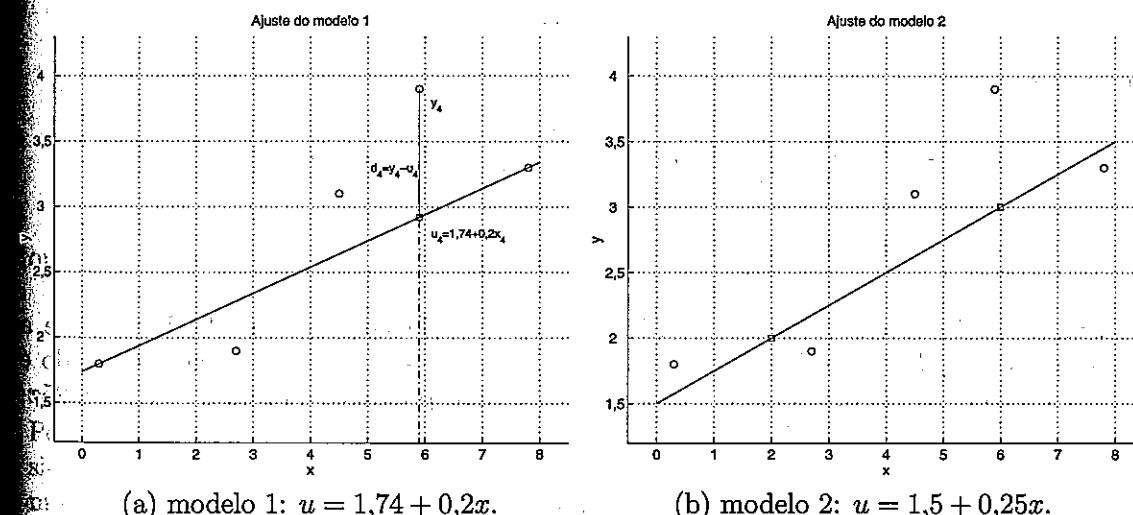


Figura 4.2 Dois exemplos de ajustes lineares.

Um modo de verificar a qualidade do ajuste é calculando a soma de todas as n distâncias verticais de y_i aos pontos da reta $u_i = 1,74 + 0,2x_i$, considerando valores positivos de d_i

$$D(b_0, b_1) = \sum_{i=1}^n (y_i - u_i)^2 = \sum_{i=1}^n (y_i - b_0 - b_1 x_i)^2 = \sum_{i=1}^n d_i^2,$$

$$D(1,74; 0,2) = \sum_{i=1}^5 (y_i - 1,74 - 0,2x_i)^2.$$

A Tabela 4.2 mostra os resultados do ajuste pelo modelo 1: $u = 1,74 + 0,2x$.

Tabela 4.2 Resultados do ajuste pelo modelo 1.

| i | x_i | y_i | u_i | d_i |
|-------------------------|-------|-------|-------|-------|
| 1 | 0,3 | 1,8 | 1,80 | 0,00 |
| 2 | 2,7 | 1,9 | 2,28 | -0,38 |
| 3 | 4,5 | 3,1 | 2,64 | 0,46 |
| 4 | 5,9 | 3,9 | 2,92 | 0,98 |
| 5 | 7,8 | 3,3 | 3,30 | 0,00 |
| $D(1,74; 0,2) = 1,3164$ | | | | |

A segunda tentativa (modelo 2) para obter um ajuste consiste em traçar a reta por dois pontos quaisquer, porém que não pertençam, necessariamente, ao diagrama de dispersão. Por exemplo, se forem escolhidos os pontos

| | | |
|-----|---|---|
| x | 2 | 6 |
| y | 2 | 3 |

a reta será

$$u(x) = y_0 + \frac{y_1 - y_0}{x_1 - x_0}(x - x_0) = 2 + \frac{3 - 2}{6 - 2}(x - 2) = 2 + 0,25(x - 2) \sim$$

$$u(x) = 1,5 + 0,25x.$$

A Figura 4.2(b) mostra a reta $u = 1,5 + 0,25x$ esboçada no diagrama de dispersão e a Tabela 4.3 compila os resultados do modelo 2. Entre os dois modelos, o mais adequado é o 2, pois os pontos do diagrama de dispersão estão mais próximos da reta $u = 1,5 + 0,25x$, visto que $D(1,5; 0,25) = 1,2300 < D(1,74; 0,2) = 1,3164$.

Tabela 4.3 Resultados do ajuste pelo modelo 2.

| i | x_i | y_i | u_i | d_i |
|-------------------------|-------|-------|-------|--------|
| 1 | 0,3 | 1,8 | 1,575 | 0,225 |
| 2 | 2,7 | 1,9 | 2,175 | -0,275 |
| 3 | 4,5 | 3,1 | 2,625 | 0,475 |
| 4 | 5,9 | 3,9 | 2,975 | 0,925 |
| 5 | 7,8 | 3,3 | 3,450 | -0,150 |
| $D(1,5; 0,25) = 1,2300$ | | | | |

4.1.3 Método dos quadrados mínimos

Pelos resultados das Tabelas 4.2 e 4.3, é claro que a qualidade do ajuste depende da equação da reta escolhida. O fato de a reta não passar por dois pontos entre aqueles do diagrama de dispersão produziu um resultado melhor. A questão é por onde se deve traçar a reta de modo a obter o menor valor do desvio D .

O método dos quadrados mínimos consiste justamente em encontrar uma estimativa da reta $u = \beta_0 + \beta_1 x$ de modo a produzir o menor valor possível do desvio

$$D(\beta_0, \beta_1) = \sum_{i=1}^n (y_i - u_i)^2 = \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2,$$

cujas derivadas parciais são

$$\frac{\partial D(\beta_0, \beta_1)}{\partial \beta_0} = -2 \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i) \text{ e}$$

$$\frac{\partial D(\beta_0, \beta_1)}{\partial \beta_1} = -2 \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i) x_i.$$

Os valores para os quais a função $D(\beta_0, \beta_1)$ possui um mínimo são aqueles onde as derivadas parciais se anulam, isto é, se (b_0, b_1) for o ponto de mínimo de $D(\beta_0, \beta_1)$, então

$$-2 \sum_{i=1}^n (y_i - b_0 - b_1 x_i) = 0 \rightarrow \sum_{i=1}^n b_0 + \sum_{i=1}^n b_1 x_i = \sum_{i=1}^n y_i \text{ e}$$

$$-2 \sum_{i=1}^n (y_i - b_0 - b_1 x_i) x_i = 0 \rightarrow \sum_{i=1}^n b_0 x_i + \sum_{i=1}^n b_1 x_i^2 = \sum_{i=1}^n x_i y_i,$$

ou na forma matricial e simplificando a notação $\sum_{i=1}^n$ para \sum

$$\begin{bmatrix} n & \sum x_i \\ \sum x_i & \sum x_i^2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \end{bmatrix}. \quad (4.1)$$

Portanto, os valores em que $D(\beta_0, \beta_1)$ apresenta um mínimo são obtidos pela solução do sistema linear (4.1), denominado equações normais. Utilizando as operações l-elementares, mostradas na Seção 2.3, obtém-se o seguinte sistema equivalente

$$\begin{bmatrix} n & \sum x_i \\ 0 & -\frac{1}{n}(\sum x_i)^2 + \sum x_i^2 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \begin{bmatrix} \sum y_i \\ -\frac{1}{n} \sum x_i \sum y_i + \sum x_i y_i \end{bmatrix},$$

cuja solução é

$$b_1 = \frac{\sum x_i \sum y_i - n \sum x_i y_i}{(\sum x_i)^2 - n \sum x_i^2} \text{ e } b_0 = \frac{\sum y_i - b_1 \sum x_i}{n}. \quad (4.2)$$

Exemplo 4.1 Calcular a reta de quadrados mínimos utilizando os dados da Tabela 4.1.

Os valores dos somatórios necessários para resolver o sistema linear (4.1) pelas expressões (4.2) estão mostrados na Tabela 4.4.

Tabela 4.4 Ajuste linear por quadrados mínimos.

| i | x_i | y_i | x_i^2 | $x_i y_i$ | y_i^2 |
|--------|-------|-------|---------|-----------|---------|
| 1 | 0,3 | 1,8 | 0,09 | 0,54 | 3,24 |
| 2 | 2,7 | 1,9 | 7,29 | 5,13 | 3,61 |
| 3 | 4,5 | 3,1 | 20,25 | 13,95 | 9,61 |
| 4 | 5,9 | 3,9 | 34,81 | 23,01 | 15,21 |
| 5 | 7,8 | 3,3 | 60,84 | 25,74 | 10,89 |
| \sum | 21,2 | 14,0 | 123,28 | 68,37 | 42,56 |

A solução de quadrados mínimos é

$$b_1 = \frac{\sum x_i \sum y_i - n \sum x_i y_i}{(\sum x_i)^2 - n \sum x_i^2} = \frac{21,2 \times 14,0 - 5 \times 68,37}{(21,2)^2 - 5 \times 123,28} \sim b_1 = 0,2698 \text{ e}$$

$$b_0 = \frac{\sum y_i - b_1 \sum x_i}{n} = \frac{14,0 - 0,2698 \times 21,2}{5} \sim b_0 = 1,6560.$$

A Figura 4.3 apresenta a reta de quadrados mínimos $u = 1,6560 + 0,2698x$, traçada no diagrama de dispersão, e a Tabela 4.5 lista os resultados do ajuste.

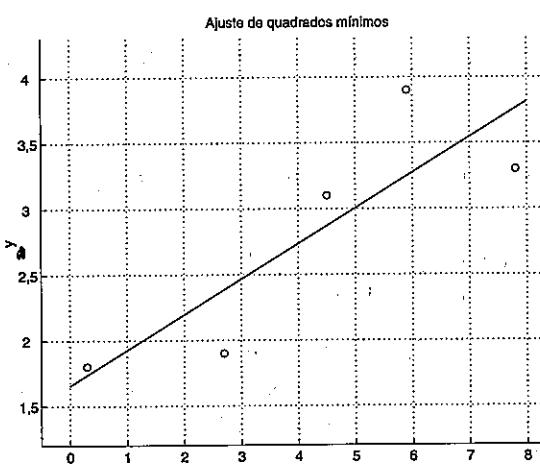


Figura 4.3 Ajuste de quadrados mínimos $u = 1,6560 + 0,2698x$.

Considerando que $D(1,6560; 0,2698) = 0,9289 < D(1,5; 0,25) = 1,2300 < D(1,74; 0,2) = 1,3164$, então o ajuste de quadrados mínimos é, sem dúvida, o melhor dos três modelos propostos por apresentar o menor desvio D .

Tabela 4.5 Resultados do ajuste por quadrados mínimos.

| i | x_i | y_i | u_i | d_i |
|------------------------------|-------|-------|--------|---------|
| 1 | 0,3 | 1,8 | 1,7369 | 0,0631 |
| 2 | 2,7 | 1,9 | 2,3845 | -0,4845 |
| 3 | 4,5 | 3,1 | 2,8701 | 0,2299 |
| 4 | 5,9 | 3,9 | 3,2478 | 0,6522 |
| 5 | 7,8 | 3,3 | 3,7604 | -0,4604 |
| $D(1,6560; 0,2698) = 0,9289$ | | | | |

4.2 Qualidade do ajuste

Nesta seção, serão apresentados dois parâmetros para aferir a qualidade do ajuste obtido pela regressão linear: o coeficiente de determinação r^2 e a variância residual σ^2 .

4.2.1 Coeficiente de determinação

Seja a expressão para o i -ésimo ponto

$$y_i - \bar{y} = (y_i - u_i) + (u_i - \bar{y}),$$

sendo $u_i = b_0 + b_1 x_i$ e $\bar{y} = \frac{1}{n} \left(\sum_{i=1}^n y_i \right)$. Tomando o quadrado em ambos os termos da igualdade, tem-se

$$(y_i - \bar{y})^2 = (y_i - u_i)^2 + (u_i - \bar{y})^2 + 2(y_i - u_i)(u_i - \bar{y}).$$

Calculando o somatório para $i = 1, 2, \dots, n$, obtém-se

$$\sum_{i=1}^n (y_i - \bar{y})^2 = \sum_{i=1}^n (y_i - u_i)^2 + \sum_{i=1}^n (u_i - \bar{y})^2 + 2 \sum_{i=1}^n (y_i - u_i)(u_i - \bar{y}). \quad (4.3)$$

No entanto,

$$\begin{aligned} \sum_{i=1}^n (y_i - u_i)(u_i - \bar{y}) &= \sum_{i=1}^n d_i(b_0 + b_1 x_i - \bar{y}) \sim \\ \sum_{i=1}^n (y_i - u_i)(u_i - \bar{y}) &= (b_0 - \bar{y}) \sum_{i=1}^n d_i + b_1 \sum_{i=1}^n x_i d_i. \end{aligned} \quad (4.4)$$

Por outro lado,

$$\sum_{i=1}^n d_i = \sum_{i=1}^n (y_i - b_0 - b_1 x_i) = \sum_{i=1}^n y_i - nb_0 - b_1 \sum_{i=1}^n x_i.$$

Em vista da expressão de b_0 em (4.2)

$$\sum_{i=1}^n d_i = 0, \quad (4.5)$$

isto é, a soma dos desvios obtidos por quadrados mínimos é zero. Além disso,

$$\sum_{i=1}^n x_i d_i = \sum_{i=1}^n (x_i(y_i - b_0 - b_1 x_i)) = \sum_{i=1}^n (x_i y_i - b_0 x_i - b_1 x_i^2).$$

Substituindo o valor de b_0 dado em (4.2), tem-se que

$$\begin{aligned} \sum_{i=1}^n x_i d_i &= \sum_{i=1}^n x_i y_i - \frac{1}{n} \left(\sum_{i=1}^n y_i - b_1 \sum_{i=1}^n x_i \right) \sum_{i=1}^n x_i - b_1 \sum_{i=1}^n x_i^2, \\ &= \frac{1}{n} \left(n \sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \sum_{i=1}^n y_i + b_1 \left(\sum_{i=1}^n x_i \right)^2 - nb_1 \sum_{i=1}^n x_i^2 \right) \sim \\ \sum_{i=1}^n x_i d_i &= \frac{1}{n} \left(n \sum_{i=1}^n x_i y_i - \sum_{i=1}^n x_i \sum_{i=1}^n y_i + b_1 \left(\left(\sum_{i=1}^n x_i \right)^2 - n \sum_{i=1}^n x_i^2 \right) \right). \end{aligned}$$

Em vista do valor de b_1 dado em (4.2),

$$\sum_{i=1}^n x_i d_i = 0. \quad (4.6)$$

Substituindo (4.5) e (4.6) em (4.4), tem-se que

$$\sum_{i=1}^n (y_i - u_i)(u_i - \bar{y}) = 0.$$

Conseqüentemente, (4.3) torna-se

$$\sum_{i=1}^n (y_i - \bar{y})^2 = \sum_{i=1}^n (y_i - u_i)^2 + \sum_{i=1}^n (u_i - \bar{y})^2,$$

sendo

$$\sum_{i=1}^n (y_i - \bar{y})^2 = \text{SQTot} \text{ (soma de quadrados total)},$$

$$\sum_{i=1}^n (y_i - u_i)^2 = \text{SQRes} \text{ (soma de quadrados residual)} \text{ e}$$

$$\sum_{i=1}^n (u_i - \bar{y})^2 = \text{SQReg} \text{ (soma de quadrados devido à regressão)}.$$

Uma maneira de avaliar a qualidade do ajuste do modelo $u = b_0 + b_1 x$ aos dados é tomado a razão entre a SQReg e a SQTot,

$$r^2 = \frac{\text{SQReg}}{\text{SQTot}} = \frac{\text{SQTot} - \text{SQRes}}{\text{SQTot}} \sim r^2 = 1 - \frac{\text{SQRes}}{\text{SQTot}},$$

sendo r^2 denominado coeficiente de determinação e satisfazendo $0 \leq r^2 \leq 1$. Quanto mais próximo r^2 for da unidade, melhor será o ajuste. Considerando que

$$D(b_0, b_1) = \sum_{i=1}^n (y_i - u_i)^2 = \sum_{i=1}^n d_i^2 \quad (4.7)$$

e que

$$\sum_{i=1}^n (y_i - \bar{y})^2 = \sum_{i=1}^n y_i^2 - 2\bar{y} \sum_{i=1}^n y_i + n\bar{y}^2 \sim \sum_{i=1}^n (y_i - \bar{y})^2 = \sum_{i=1}^n y_i^2 - \frac{1}{n} \left(\sum_{i=1}^n y_i \right)^2,$$

então

$$r^2 = 1 - \frac{D(b_0, b_1)}{\sum y_i^2 - \frac{1}{n} (\sum y_i)^2}. \quad (4.8)$$

O coeficiente de determinação r^2 pode ser visto como a proporção da variação total dos dados em torno da média \bar{y} que é explicada pelo modelo de regressão.

4.2.2 Variância residual

Um outro parâmetro importante para aferir a qualidade do ajuste é a variância residual σ^2 definida por

$$\sigma^2 = \frac{D(b_0, b_1)}{n - p}, \quad (4.9)$$

onde $D(b_0, b_1)$ é dado por (4.7), n é o número de pontos e p é o número de parâmetros estimados. No caso de regressão linear simples $u = b_0 + b_1 x$, tem-se que $p = 2$.

Tanto o numerador quanto o denominador de (4.9) irão diminuir se forem introduzidos mais parâmetros no modelo, como será visto mais adiante. Todavia, será a redução global de σ^2 que definirá se mais parâmetros devem ou não ser incorporados ao modelo.

Exemplo 4.2 Calcular o coeficiente de determinação e a variância residual, utilizando os dados das Tabelas 4.4 e 4.5. Por (4.7) e (4.8),

$$r^2 = 1 - \frac{0,9289}{42,56 - (14,0)^2/5} \sim r^2 = 0,7235.$$

A variância residual dada por (4.9) é

$$\sigma^2 = \frac{0,9289}{5 - 2} \sim \sigma^2 = 0,3096.$$

Exemplo 4.3 Calcular a reta de quadrados mínimos a partir dos dados da Tabela 4.6.

Tabela 4.6 Dados para ajuste por regressão linear simples.

| x | 1,2 | 2,5 | 3,0 | 4,1 | 6,2 | 7,1 | 8,8 | 9,5 |
|---|-----|-----|-----|-----|------|------|------|------|
| y | 6,8 | 6,1 | 9,9 | 9,7 | 12,1 | 17,9 | 18,0 | 21,5 |

Inicialmente, monta-se o dispositivo prático, mostrado na Tabela 4.7, para facilitar a obtenção dos somatórios necessários ao cálculo dos parâmetros de quadrados mínimos.

Tabela 4.7 Dispositivo para regressão linear simples por quadrados mínimos.

| i | x_i | y_i | x_i^2 | $x_i y_i$ | y_i^2 | u_i | d_i | d_i^2 |
|--------|-------|-------|---------|-----------|---------|----------|---------|---------|
| 1 | 1,2 | 6,8 | 1,44 | 8,16 | 46,24 | 5,4037 | 1,3963 | 1,9497 |
| 2 | 2,5 | 6,1 | 6,25 | 15,25 | 37,21 | 7,7330 | -1,6330 | 2,6667 |
| 3 | 3,0 | 9,9 | 9,00 | 29,70 | 98,01 | 8,6289 | 1,2711 | 1,6157 |
| 4 | 4,1 | 9,7 | 16,81 | 39,77 | 94,09 | 10,5999 | -0,8999 | 0,8098 |
| 5 | 6,2 | 12,1 | 38,44 | 75,02 | 146,41 | 14,3627 | -2,2627 | 5,1198 |
| 6 | 7,1 | 17,9 | 50,41 | 127,09 | 320,41 | 15,9753 | 1,9247 | 3,7045 |
| 7 | 8,8 | 18,0 | 77,44 | 158,40 | 324,00 | 19,0213 | -1,0213 | 1,0431 |
| 8 | 9,5 | 21,5 | 90,25 | 204,25 | 462,25 | 20,2756 | 1,2244 | 1,4992 |
| \sum | 42,4 | 102,0 | 290,04 | 657,64 | 1528,62 | 102,0003 | -0,0004 | 18,4085 |

Usando valores da Tabela 4.7 em (4.2),

$$b_1 = \frac{\sum x_i \sum y_i - n \sum x_i y_i}{(\sum x_i)^2 - n \sum x_i^2} = \frac{42,4 \times 102,0 - 8 \times 657,64}{(42,4)^2 - 8 \times 290,04} \rightsquigarrow b_1 = 1,7918$$

e

$$b_0 = \frac{\sum y_i - b_1 \sum x_i}{n} = \frac{102,0 - 1,7918 \times 42,4}{8} \rightsquigarrow b_0 = 3,2535.$$

O coeficiente de determinação, dado por (4.7) e (4.8), é

$$r^2 = 1 - \frac{D(b_0, b_1)}{\sum y_i^2 - \frac{1}{n} (\sum y_i)^2} = 1 - \frac{18,4085}{1528,62 - (102,0)^2 / 8} \rightsquigarrow r^2 = 0,9193.$$

A variância residual, dada por (4.9), é

$$\sigma^2 = \frac{18,4085}{8 - 2} \rightsquigarrow \sigma^2 = 3,0681.$$

A equação de quadrados mínimos $u = 3,2535 + 1,7918x$ traçada no diagrama de dispersão dos dados da Tabela 4.6 é mostrada na Figura 4.4.

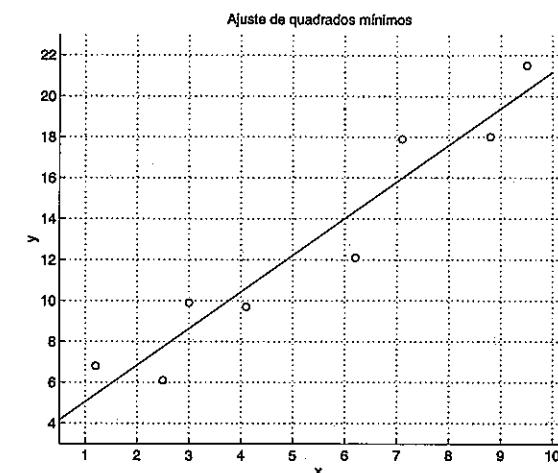


Figura 4.4 Equação de quadrados mínimos $u = 3,2535 + 1,7918x$.

4.3 Regressão linear múltipla

Um modelo mais completo que relaciona a variável resposta y com as p variáveis explicativas x_i é

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \dots + \beta_p x_p + \epsilon, \quad (4.10)$$

onde β_i , $i = 0, 1, \dots, p$, são os parâmetros a serem estimados e ϵ é uma variável aleatória desconhecida que interfere na verdadeira relação linear.

4.3.1 Equações normais

De modo similar à regressão linear simples, o método dos quadrados mínimos pode ser utilizado para estimar os $p + 1$ parâmetros β_i . Assim,

$$D(\beta_0, \beta_1, \beta_2, \dots, \beta_p) = \sum_{i=1}^n (y_i - u_i)^2 = \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} - \dots - \beta_p x_{ip})^2,$$

sendo x_{ij} a i -ésima observação da j -ésima variável explicativa. As derivadas parciais da função desvio D são

$$\frac{\partial D(\beta_0, \beta_1, \beta_2, \dots, \beta_p)}{\partial \beta_0} = -2 \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} - \dots - \beta_p x_{ip}),$$

$$\frac{\partial D(\beta_0, \beta_1, \beta_2, \dots, \beta_p)}{\partial \beta_1} = -2 \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} - \dots - \beta_p x_{ip}) x_{i1},$$

$$\frac{\partial D(\beta_0, \beta_1, \beta_2, \dots, \beta_p)}{\partial \beta_2} = -2 \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} - \dots - \beta_p x_{ip}) x_{i2},$$

$$\frac{\partial D(\beta_0, \beta_1, \beta_2, \dots, \beta_p)}{\partial \beta_p} = -2 \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} - \dots - \beta_p x_{ip}) x_{ip}.$$

Se $D(b_0, b_1, b_2, \dots, b_p)$ for o ponto de mínimo da função $D(\beta_0, \beta_1, \beta_2, \dots, \beta_p)$, então

$$\frac{\partial D(b_0, b_1, b_2, \dots, b_p)}{\partial \beta_i} = 0, \quad i = 0, 1, \dots, p$$

$$-2 \sum_{i=1}^n (y_i - b_0 - b_1 x_{i1} - b_2 x_{i2} - \dots - b_p x_{ip}) = 0 \rightsquigarrow$$

$$\sum_{i=1}^n b_0 + \sum_{i=1}^n b_1 x_{i1} + \sum_{i=1}^n b_2 x_{i2} + \dots + \sum_{i=1}^n b_p x_{ip} = \sum_{i=1}^n y_i,$$

$$-2 \sum_{i=1}^n (y_i - b_0 - b_1 x_{i1} - b_2 x_{i2} - \dots - b_p x_{ip}) x_{i1} = 0 \rightsquigarrow$$

$$\sum_{i=1}^n b_0 x_{i1} + \sum_{i=1}^n b_1 x_{i1} x_{i1} + \sum_{i=1}^n b_2 x_{i2} x_{i1} + \dots + \sum_{i=1}^n b_p x_{ip} x_{i1} = \sum_{i=1}^n x_{i1} y_i,$$

$$-2 \sum_{i=1}^n (y_i - b_0 - b_1 x_{i1} - b_2 x_{i2} - \dots - b_p x_{ip}) x_{i2} = 0 \rightsquigarrow$$

$$\sum_{i=1}^n b_0 x_{i2} + \sum_{i=1}^n b_1 x_{i1} x_{i2} + \sum_{i=1}^n b_2 x_{i2} x_{i2} + \dots + \sum_{i=1}^n b_p x_{ip} x_{i2} = \sum_{i=1}^n x_{i2} y_i,$$

⋮

$$-2 \sum_{i=1}^n (y_i - b_0 - b_1 x_{i1} - b_2 x_{i2} - \dots - b_p x_{ip}) x_{ip} = 0 \rightsquigarrow$$

$$\sum_{i=1}^n b_0 x_{ip} + \sum_{i=1}^n b_1 x_{i1} x_{ip} + \sum_{i=1}^n b_2 x_{i2} x_{ip} + \dots + \sum_{i=1}^n b_p x_{ip} x_{ip} = \sum_{i=1}^n x_{ip} y_i.$$

Escrevendo o sistema de equações lineares acima na forma matricial e simplificando a notação $\sum_{i=1}^n$ para \sum , têm-se as equações normais

$$\begin{bmatrix} n & \sum x_{i1} & \sum x_{i2} & \cdots & \sum x_{ip} \\ \sum x_{i1} & \sum x_{i1} x_{i1} & \sum x_{i2} x_{i1} & \cdots & \sum x_{ip} x_{i1} \\ \sum x_{i2} & \sum x_{i1} x_{i2} & \sum x_{i2} x_{i2} & \cdots & \sum x_{ip} x_{i2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum x_{ip} & \sum x_{i1} x_{ip} & \sum x_{i2} x_{ip} & \cdots & \sum x_{ip} x_{ip} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_p \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_{i1} y_i \\ \sum x_{i2} y_i \\ \vdots \\ \sum x_{ip} y_i \end{bmatrix}. \quad (4.11)$$

O vetor solução b $((p+1) \times 1)$ do sistema de equações lineares (4.11) fornece os parâmetros para a equação de quadrados mínimos

$$u = b_0 + b_1 x_1 + b_2 x_2 + \dots + b_p x_p.$$

O coeficiente de determinação

$$r^2 = 1 - \frac{D(b_0, b_1, \dots, b_p)}{\sum y_i^2 - \frac{1}{n} (\sum y_i)^2} \quad (4.12)$$

e a variância residual

$$\sigma^2 = \frac{D(b_0, b_1, \dots, b_p)}{n-p} \quad (4.13)$$

são utilizados para medir a qualidade do ajuste de um modelo de regressão linear múltipla.

4.3.2 Regressão polinomial

Um caso particular e importante de (4.10) é quando se relaciona a variável resposta y com uma variável explicativa x , segundo o modelo

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \dots + \beta_g x^g + \epsilon,$$

designado regressão polinomial. Para este caso particular, o sistema (4.11) torna-se

$$\begin{bmatrix} n & \sum x_i & \sum x_i^2 & \cdots & \sum x_i^g \\ \sum x_i & \sum x_i^2 & \sum x_i^3 & \cdots & \sum x_i^{g+1} \\ \sum x_i^2 & \sum x_i^3 & \sum x_i^4 & \cdots & \sum x_i^{g+2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \sum x_i^g & \sum x_i^{g+1} & \sum x_i^{g+2} & \cdots & \sum x_i^{2g} \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_g \end{bmatrix} = \begin{bmatrix} \sum y_i \\ \sum x_i y_i \\ \sum x_i^2 y_i \\ \vdots \\ \sum x_i^g y_i \end{bmatrix}. \quad (4.14)$$

4.3.3 Algoritmo e complexidade

A Figura 4.5 apresenta um algoritmo para calcular os parâmetros da equação de quadrados mínimos por meio das equações normais (4.11). Os parâmetros de entrada são o número n de pontos, o número v de variáveis, o número p de parâmetros da equação de regressão, a matriz x , de dimensão $n \times v$, contendo as variáveis explicativas e o vetor y , de tamanho n , com as variáveis respostas. Por exemplo, sejam os modelos abaixo e os correspondentes valores de v e p

$$u = b_0 + b_1 x \rightsquigarrow v = 1, p = 2,$$

$$u = b_0 + b_1 x_1 + b_2 x_2 \rightsquigarrow v = 2, p = 3 \text{ e}$$

$$u = b_0 + b_1 x + b_2 x^2 \rightsquigarrow v = 1, p = 3.$$

```

Algoritmo Regressão_linear_EN
{ Objetivo: Calcular parâmetros de quadrados mínimos de modelo linear múltiplo }
{ via equações normais }

parâmetros de entrada n, v, p, x, y
{ número de pontos, número de variáveis, número de parâmetros, }
{ variáveis explicativas e variáveis respostas }

parâmetros de saída b, r2, sigma2, CondErro
{ coef. de regressão, coef. de determinação, variância residual e condição de erro }

se v > 1 e v + 1 ≠ p então CondErro ← 1, abandone, fimse
CondErro ← 0; vpl ← v + 1; pm1 ← p - 1
para i ← 1 até n faça { inclusão de uma coluna de 1's relativa à b0 }
    para j ← vpl até 2 passo -1 faça
        x(i,j) ← x(i,j-1)
    fimpara; x(i,1) ← 1
fimpara

se v = 1 e p > 2 então { se regressão polinomial, então gera potências de x }
    para j ← 2 até pm1 faça
        jp1 ← j + 1
        para i ← 1 até n faça x(i,jp1) ← x(i,2)j, fimpara
    fimpara
fimse

{ equações normais }
para i ← 1 até p faça
    para j ← 1 até p faça
        Soma ← 0; para k ← 1 até n faça, Soma ← Soma + x(k,i) * x(k,j), fimpara
        Sx(i,j) ← Soma { matriz dos coeficientes }
    fimpara
    Soma ← 0; para k ← 1 até n faça, Soma ← Soma + x(k,i) * y(k), fimpara
    Sxy(i) ← Soma { vetor dos termos independentes }
fimpara

[L, Det, CondErro] ← Cholesky(p, Sxx) { decomposição de Cholesky } (ver Figura 2.7)
t ← Substituições_Sucessivas(p, L, Sxy) (ver Figura 2.3)
para i ← 1 até p faça
    para j ← 1 até n faça U(j,i) ← L(i,j), fimpara; { U = transposta de L }
fimpara

b ← Substituições_Retroativas(p, U, t) { coeficientes } (ver Figura 2.4)
D ← 0; Sy2 ← 0
para i ← 1 até n faça
    u ← 0; para j ← 1 até p faça, u ← u + b(j) * x(i,j), fimpara
    d ← y(i) - u; D ← D + d2; Sy2 ← Sy2 + y(i)2
fimpara
r2 ← 1 - D/(Sy2 - Sxy(1)2/n) { coeficiente de determinação }
sigma2 ← D/(n - p) { variância residual }

finalgoritmo

```

Figura 4.5 Regressão linear múltipla e polinomial via equações normais.

Todavia, não é permitido neste algoritmo um modelo no qual ($v > 1$ e $v + 1 \neq p$), como

$$u = b_0 + b_1x_1 + b_2x_2 + b_3x_2^2 \rightsquigarrow v = 2, p = 4.$$

Os parâmetros de saída são o vetor b contendo os coeficientes da equação de regressão, sendo $u = b_1 + b_2x_1 + b_3x_2 + b_4x_3 + \dots + b_px_v$, o coeficiente de determinação $r2$, a variância residual $sigma2$ e a condição de erro $CondErro$. A condição $CondErro = 1$ significa que o modelo não é permitido ($v > 1$ e $v + 1 \neq p$), situação em que o algoritmo é abandonado.

O algoritmo Cholesky, da Figura 2.7, pode ser utilizado para fazer a decomposição de Cholesky (Seção 2.5), pois a matriz das equações normais (4.11) é simétrica e definida positiva. Os algoritmos Substituições_Sucessivas, da Figura 2.3, e Substituições_Retroativas, da Figura 2.4, também são utilizados para obter o vetor solução b .

A Tabela 4.8 mostra a complexidade computacional do algoritmo da Figura 4.5. São apresentadas, separadamente, as complexidades da regressão linear múltipla e da polinomial, haja vista as diferenças entre os dois modelos. A potenciação é contada como multiplicações. As operações para solução do sistema linear não foram computadas, mas podem ser obtidas nas Tabelas 2.2, 2.3 e 2.5.

Tabela 4.8 Complexidade da regressão via equações normais.

(n: número de pontos e p: número de parâmetros. Não foram consideradas as operações para solução do sistema linear.)

| Operações | Complexidade | Operações | Complexidade |
|----------------|-----------------------|----------------|--|
| adições | $(p^2 + 3p + 2)n + 5$ | adições | $(p^2 + 2p + 4)n + p + 3$ |
| multiplicações | $(p^2 + 2p + 2)n + 1$ | multiplicações | $(\frac{3}{2}p^2 + \frac{1}{2}p + 3)n + 1$ |
| divisões | 3 | divisões | 3 |

a) Regressão linear múltipla

b) Regressão polinomial

Exemplo 4.4 Ajustar os dados da Tabela 4.9 ao modelo $u = b_0 + b_1x_1 + b_2x_2$ usando o algoritmo mostrado na Figura 4.5.

Tabela 4.9 Produto interno bruto dos Estados Unidos de 1947 a 1962.

(x_1 é o total de empregos (milhões), x_2 a população com 14 anos ou mais (milhões) e y o PIB americano (bilhões de dólares), conforme [36, Tabela 17.9.1].)

| i | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
|----------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| x_{i1} | 60,3 | 61,1 | 60,2 | 61,2 | 63,2 | 63,6 | 65,0 | 63,8 | 66,0 | 67,9 | 68,2 | 66,5 | 68,7 | 69,6 | 69,3 | 70,6 |
| x_{i2} | 108 | 109 | 110 | 112 | 112 | 113 | 115 | 116 | 117 | 119 | 120 | 122 | 123 | 125 | 128 | 130 |
| y_i | 234 | 259 | 258 | 285 | 329 | 347 | 365 | 363 | 396 | 419 | 443 | 445 | 483 | 503 | 518 | 555 |

```
% Os parametros de entrada
n = 16
v = 2
p = 3
x =
  60.3000 108.0000
  61.1000 109.0000
  60.2000 110.0000
  61.2000 112.0000
  63.2000 112.0000
  63.6000 113.0000
  65.0000 115.0000
  63.8000 116.0000
  66.0000 117.0000
  67.9000 119.0000
  68.2000 120.0000
  66.5000 122.0000
  68.7000 123.0000
  69.6000 125.0000
  69.3000 128.0000
  70.6000 130.0000
y =
  234
  259
  258
  285
  329
  347
  365
  363
  396
  419
  443
  445
  483
  503
  518
  555
% produzem os resultados
coeficientes de regressao
b(0) = -1.40740e+03
b(1) = 1.34511e+01
b(2) = 7.80271e+00
coeficiente de determinacao = 0.99267
variancia residual = 8.37581e+01
condicao de erro = 0
```

Portanto, a equação de quadrados mínimos é $u = -1,40740 \times 10^3 + 1,34511 \times 10^1 x_1 + 7,80271 x_2$, com coeficiente de determinação $r^2 = 0,99267$ e variância residual $\sigma^2 = 83,7581$.

Exemplo 4.5 A partir da Tabela 4.10, que compila valores de \sqrt{x} para $0,01 \leq x \leq 1$, determinar o polinômio de quadrados mínimos de grau $g = 3$ utilizando o algoritmo da Figura 4.5.

Tabela 4.10 Raiz quadrada de x para $0,01 \leq x \leq 1$.

| i | x_i | $\sqrt{x_i}$ |
|-----|-------|--------------|
| 1 | 0,01 | 0,1000 |
| 2 | 0,10 | 0,3162 |
| 3 | 0,20 | 0,4472 |
| 4 | 0,30 | 0,5477 |
| 5 | 0,40 | 0,6325 |
| 6 | 0,50 | 0,7071 |
| 7 | 0,60 | 0,7746 |
| 8 | 0,70 | 0,8367 |
| 9 | 0,80 | 0,8944 |
| 10 | 0,90 | 0,9487 |
| 11 | 1,00 | 1,0000 |

% Os parametros de entrada

```
n = 11
v = 1
p = 4
x =
  0.0100
  0.1000
  0.2000
  0.3000
  0.4000
  0.5000
  0.6000
  0.7000
  0.8000
  0.9000
  1.0000
y =
  0.1000
  0.3162
  0.4472
  0.5477
  0.6325
  0.7071
  0.7746
  0.8367
  0.8944
  0.9487
  1.0000
```

% fornecem os resultados

```
Coeficientes de regressao
b(0) = 1.01126e-01
b(1) = 2.06854e+00
b(2) = -2.17822e+00
b(3) = 1.01865e+00
Coeficiente de determinacao = 0.99738
Variancia residual = 2.96085e-04
Condicao de erro = 0
```

Conseqüentemente, o polinômio de quadrados mínimos que aproxima \sqrt{x} para $0,01 \leq x \leq 1$ é $u = 1,01865x^3 - 2,17822x^2 + 2,06854x + 0,10113$, com coeficiente de determinação $r^2 = 0,99738$ e variância residual $\sigma^2 = 2,96085 \times 10^{-4}$. ■

Exemplo 4.6 Sejam os dados históricos dos censos demográficos do Brasil, de acordo com o Instituto Brasileiro de Geografia e Estatística (IBGE)¹, apresentados na Tabela 4.11. Deseja-se determinar qual o melhor grau para uma regressão polinomial.

Tabela 4.11 População do Brasil.

| Ano | Urbana | Rural |
|------|-------------|------------|
| 1940 | 12.880.182 | 28.356.133 |
| 1950 | 18.782.891 | 33.161.506 |
| 1960 | 31.303.034 | 38.767.423 |
| 1970 | 52.084.984 | 41.054.053 |
| 1980 | 80.436.409 | 38.566.297 |
| 1991 | 110.990.990 | 35.834.485 |
| 1996 | 123.076.831 | 33.993.332 |
| 2000 | 137.953.959 | 31.845.211 |

O diagrama de dispersão da Figura 4.6(a) mostra que o ajuste não deve ser feito por um polinômio de grau 1 e sim por um de grau mais elevado. A Tabela 4.12 compila os valores do coeficiente de determinação r^2 e da variância residual σ^2 para o modelo polinomial $u = b_0 + b_1x + b_2x^2 + \dots + b_gx^g$ usando o algoritmo da Figura 4.5. Para reduzir os erros de arredondamento são feitas mudanças de variáveis. A variável explicativa é centrada de modo que $x = \text{Ano} - 1970$ e a variável resposta é dada em milhões de habitantes $y = \text{Urbana} \times 10^{-6}$.

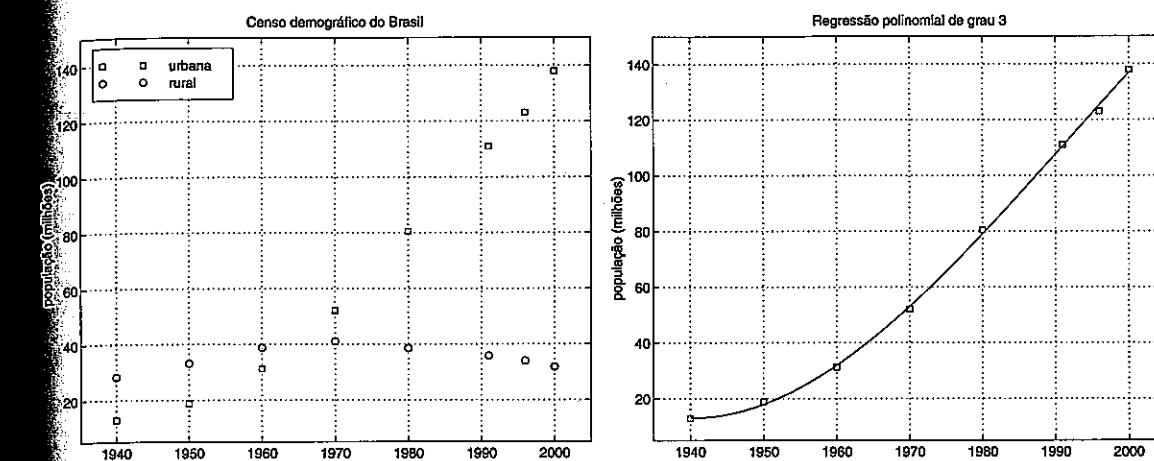
Tabela 4.12 Escolha do grau do polinômio de quadrados mínimos.

(g é o grau do polinômio, r^2 o coeficiente de determinação e σ^2 a variância residual.)

| g | r^2 | σ^2 |
|-----|---------|--------------------------|
| 1 | 0,96602 | $9,58219 \times 10^1$ |
| 2 | 0,99776 | $7,59261 \times 10^0$ |
| 3 | 0,99939 | $2,57228 \times 10^0$ |
| 4 | 0,99940 | $3,42930 \times 10^0$ |
| 5 | 0,99980 | $1,65615 \times 10^0$ |
| 6 | 0,99998 | $4,17203 \times 10^{-1}$ |

Como era de se esperar, r^2 aumenta quando o grau do polinômio de quadrados mínimos é aumentado. No entanto, σ^2 vai reduzindo até grau $g = 3$ e depois começa a oscilar. Este deve ser o grau escolhido para o ajuste polinomial porque não se deve usar grau elevado e o ideal seria que $n \gg p$. A Figura 4.6(b) mostra esse polinômio traçado no diagrama de dispersão.

¹www.ibge.gov.br



(a) Diagrama de dispersão.

(b) Polinômio de grau 3.

Figura 4.6 Grau do polinômio de quadrados mínimos.

4.3.4 Transformações não lineares

Modelos não lineares nos parâmetros podem ser transformados em modelos lineares pela simples substituição de variáveis por funções dessas variáveis, por exemplo,

$$y = ax^b \rightsquigarrow \log_e(y) = \log_e(a) + b \log_e(x);$$

$$y = ab^x \rightsquigarrow \log_e(y) = \log_e(a) + \log_e(b)x;$$

$$y = ae^{bx} \rightsquigarrow \log_e(y) = \log_e(a) + bx;$$

$$y = e^{a+bx_1+cx_2} \rightsquigarrow \log_e(y) = a + bx_1 + cx_2;$$

$$y = ax_1^bx_2^c \rightsquigarrow \log_e(y) = \log_e(a) + b \log_e(x_1) + c \log_e(x_2);$$

$$y = \frac{1}{a + bx_1 + cx_2} \rightsquigarrow \frac{1}{y} = a + bx_1 + cx_2;$$

$$y = \frac{1}{1 + e^{a+bx_1+cx_2}} \rightsquigarrow \log_e\left(\frac{1}{y} - 1\right) = a + bx_1 + cx_2.$$

Exemplo 4.7 Em uma reação química de primeira ordem, a constante k de velocidade se relaciona com a concentração c e o tempo t pela expressão

$$c = c_0 e^{-kt},$$

onde c_0 é a concentração inicial de um reagente. Usando os dados da Tabela 4.13 e o algoritmo da Figura 4.5, calcular a constante de velocidade.

Para tal, $c = c_0 e^{-kt} \rightsquigarrow \log_e(c) = \log_e(c_0) - kt$.

Tabela 4.13 Cinética de reação química de primeira ordem.

(t é o tempo (segundos) e c é a concentração (M).)

| i | 1 | 2 | 3 | 4 | 5 |
|---|------|------|------|------|------|
| t | 0,1 | 0,2 | 0,3 | 0,4 | 0,5 |
| c | 0,56 | 0,32 | 0,21 | 0,11 | 0,08 |

% Os parametros de entrada

n = 5

v = 1

p = 2

t =

0.1000

0.2000

0.3000

0.4000

0.5000

logc =

-0.5798

-1.1394

-1.5606

-2.2073

-2.5257

% fornecem os resultados

b(0) = -1.14650e-01

b(1) = -4.95970e+00

coeficiente de determinacao = 0.99179

variancia residual = 6.78379e-03

condicao de erro = 0

A equação de regressão é $\log_e(c) = -1,14650 \times 10^{-1} - 4,95970t \Rightarrow c_0 = e^{-1,14650 \times 10^{-1}} = 0,89168 M$ e $k = 4,95970$ segundos $^{-1}$. O coeficiente de determinação é $r^2 = 0,99179$ e a variância residual é $\sigma^2 = 6,78379 \times 10^{-3}$.

4.3.5 Malcondicionamento

Seja a equação de regressão polinomial

$$u = b_0 + b_1x + b_2x^2 + \dots + b_gx^g,$$

sendo os parâmetros b_i calculados pelas equações normais (4.14). A Tabela 4.14 mostra o coeficiente de determinação r^2 e o número de condição espectral $\kappa_2(X^T X)$ da matriz dos coeficientes $X^T X$ das equações normais, para o grau do polinômio $g = 1, 2, \dots, 7$. Foram usados os dados (Ano, Urbana) da Tabela 4.11 com número de pontos $n = 8$. Como era de se esperar, à medida que o grau g do polinômio aumenta, $r^2 \rightarrow 1$, contudo, $\kappa_2(X^T X) \rightarrow \infty$. Isso mostra que as equações normais possuem a matriz dos coeficientes malcondicionada.

Tabela 4.14 Malcondicionamento das equações normais.

(g é o grau do polinômio de regressão, r^2 o coeficiente de determinação e $\kappa_2(X^T X)$ o número de condição em norma-2 da matriz dos coeficientes $X^T X$ das equações normais.)

| g | r^2 | $\kappa_2(X^T X)$ |
|---|---------|------------------------|
| 1 | 0,96602 | $4,514 \times 10^2$ |
| 2 | 0,99776 | $8,298 \times 10^5$ |
| 3 | 0,99939 | $6,498 \times 10^8$ |
| 4 | 0,99940 | $8,591 \times 10^{11}$ |
| 5 | 0,99980 | $7,418 \times 10^{14}$ |
| 6 | 0,99998 | $9,923 \times 10^{17}$ |
| 7 | 1,00000 | $6,753 \times 10^{20}$ |

4.4 Ajuste via decomposição em valores singulares

O modelo de regressão linear múltipla pode ser descrito na forma matricial tal como

$$y = X\beta + \epsilon,$$

onde y é um vetor ($n \times 1$) contendo as n observações da variável resposta, X é uma matriz ($n \times (p+1)$), $n \geq p+1$, contendo os n valores das p variáveis explicativas, além da primeira coluna de 1's relativa a β_0 , β é um vetor $((p+1) \times 1)$ dos parâmetros a serem estimados e ϵ é um vetor ($n \times 1$) dos erros aleatórios

$$y = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ \vdots \\ y_n \end{bmatrix}, X = \begin{bmatrix} 1 & x_{11} & x_{12} & \cdots & x_{1p} \\ 1 & x_{21} & x_{22} & \cdots & x_{2p} \\ 1 & x_{31} & x_{32} & \cdots & x_{3p} \\ 1 & x_{41} & x_{42} & \cdots & x_{4p} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & x_{n2} & \cdots & x_{np} \end{bmatrix}, \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} \text{ e } \epsilon = \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \vdots \\ \epsilon_n \end{bmatrix}.$$

4.4.1 Cálculo dos parâmetros

Uma estimativa do vetor β , pelo método dos quadrados mínimos, consiste em minimizar a função

$$f(\beta) = \|y - X\beta\|_2^2 = (y - X\beta)^T(y - X\beta).$$

Pelas regras de diferenciação matricial

$$\frac{\partial f(\beta)}{\partial \beta^T} = \frac{\partial f(\beta)}{\partial (y - X\beta)^T} \frac{\partial (y - X\beta)}{\partial \beta^T} = 2(y - X\beta)^T(-X) = -2(y - X\beta)^T X.$$

Portanto,

$$\frac{\partial f(\beta)}{\partial \beta} = -2X^T(y - X\beta).$$

A função $f(\beta)$ apresenta um mínimo em $f(b)$, onde b é o ponto em que a derivada se anula

$$\frac{\partial f(b)}{\partial \beta} = -2X^T(y - Xb) = 0 \leadsto (X^T X)b = X^T y,$$

que são as equações normais (4.11) na forma matricial. Além disso,

$$\frac{\partial(\partial f(\beta)/\partial \beta)}{\partial \beta^T} = \frac{\partial(-2X^T y + 2X^T X\beta)}{\partial \beta^T} = 2X^T X.$$

Como em problemas de regressão a matriz $X^T X$ tem elementos reais e é não singular, então ela é definida positiva mostrando que $f(b)$ é, de fato, um mínimo de $f(\beta)$; mais ainda, o sistema acima apresenta uma única solução. No entanto, conforme visto anteriormente, as equações normais formam um sistema malcondicionado. Será apresentado, a seguir, um processo alternativo para a estimativa de β que evita a formação da matriz $X^T X$.

4.4.2 Decomposição em valores singulares

A decomposição em valores singulares [20, 34] consiste em fatorar uma matriz X ($n \times (p+1)$), tal que

$$X = USV^T, \quad (4.15)$$

onde U é uma matriz ortogonal ($n \times n$), V é uma matriz ortogonal ($(p+1) \times (p+1)$) e S é uma matriz diagonal ($n \times (p+1)$) da forma

$$S = \begin{bmatrix} S_1 \\ 0 \end{bmatrix},$$

sendo S_1 uma matriz quadrada diagonal de ordem $p+1$. A soma de quadrados residual é

$$\|y - Xb\|_2^2 = \|y - USV^T b\|_2^2.$$

A matriz ortogonal U^T não altera o valor da norma, portanto

$$\|y - Xb\|_2^2 = \|U^T y - U^T USV^T b\|_2^2 \leadsto \|y - Xb\|_2^2 = \|U^T y - SV^T b\|_2^2.$$

Sendo

$$U^T y = a = \begin{bmatrix} a_1 \\ a_2 \end{bmatrix}, \quad (4.16)$$

onde a_1 é um vetor de tamanho $p+1$ e a_2 é um vetor de tamanho $n-p-1$,

$$\tilde{b} = V^T b, \quad (4.17)$$

$$S\tilde{b} = \begin{bmatrix} S_1 \\ 0 \end{bmatrix} \tilde{b} = \begin{bmatrix} S_1 \tilde{b} \\ 0 \end{bmatrix},$$

então

$$\|y - Xb\|_2^2 = \left\| \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} - \begin{bmatrix} S_1 \tilde{b} \\ 0 \end{bmatrix} \right\|_2^2 \leadsto \|y - Xb\|_2^2 = \|a_1 - S_1 \tilde{b}\|_2^2 + \|a_2\|_2^2.$$

A soma de quadrados residual será mínima quando \tilde{b} for a solução do sistema diagonal $S_1 \tilde{b} = a_1$. Em vista de (4.17) e da ortogonalidade de V ,

$$b = V\tilde{b}. \quad (4.18)$$

No contexto da decomposição em valores singulares, a soma de quadrados residual é

$$D(b_0, b_1, \dots, b_p) = \|a_2\|_2^2 = a_2^T a_2,$$

os valores preditos são

$$u = Xb = USV^T b = US\tilde{b} = U \begin{bmatrix} S_1 \tilde{b} \\ 0 \end{bmatrix} \leadsto u = U \begin{bmatrix} a_1 \\ 0 \end{bmatrix}$$

e o vetor desvio d é

$$d = U \begin{bmatrix} 0 \\ a_2 \end{bmatrix}. \quad (4.19)$$

Exemplo 4.8 Calcular os parâmetros da reta de quadrados mínimos do Exemplo 4.1 utilizando a decomposição em valores singulares.

Sejam a matriz X e o vetor y

$$X = \begin{bmatrix} 1 & 0,3 \\ 1 & 2,7 \\ 1 & 4,5 \\ 1 & 5,9 \\ 1 & 7,8 \end{bmatrix} \text{ e } y = \begin{bmatrix} 1,8 \\ 1,9 \\ 3,1 \\ 3,9 \\ 3,3 \end{bmatrix}.$$

A decomposição em valores singulares de X fornece

$$U = \begin{bmatrix} 0,0414 & -0,8144 & -0,3904 & -0,3366 & -0,2635 \\ 0,2513 & -0,4559 & 0,1351 & 0,3940 & 0,7453 \\ 0,4087 & -0,1871 & 0,8174 & -0,2246 & -0,2816 \\ 0,5311 & 0,0219 & -0,2412 & 0,6611 & -0,4714 \\ 0,6972 & 0,3057 & -0,3209 & -0,4939 & 0,2712 \end{bmatrix},$$

$$S = \begin{bmatrix} 11,2679 & 0 \\ 0 & 1,1467 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \text{ e } V = \begin{bmatrix} 0,1713 & -0,9852 \\ 0,9852 & 0,1713 \end{bmatrix}.$$

Por (4.16)

$$a = U^T y = \begin{bmatrix} 6,1910 \\ -1,8179 \\ 0,0883 \\ 0,3949 \\ -0,8747 \end{bmatrix}.$$

O vetor \tilde{b} é a solução do sistema diagonal $S_1 \tilde{b} = a_1$

$$\begin{bmatrix} 11,2679 & 0 \\ 0 & 1,1467 \end{bmatrix} \begin{bmatrix} \tilde{b}_1 \\ \tilde{b}_2 \end{bmatrix} = \begin{bmatrix} 6,1910 \\ -1,8179 \end{bmatrix} \rightsquigarrow \tilde{b} = \begin{bmatrix} 0,5494 \\ -1,5853 \end{bmatrix}.$$

O vetor b dos coeficientes é obtido por (4.18)

$$b = V \tilde{b} \rightsquigarrow b = \begin{bmatrix} 1,6559 \\ 0,2697 \end{bmatrix}.$$

4.4.3 Algoritmo e complexidade

A Figura 4.7 apresenta um algoritmo para calcular os parâmetros da equação de quadrados mínimos por meio da decomposição em valores singulares usando (4.15)–(4.19). Os parâmetros de entrada são o número n de pontos, o número v de variáveis, o número p de parâmetros da equação de regressão, a matriz x , de dimensão $n \times v$, contendo as variáveis explicativas e o vetor y , de tamanho n , com as variáveis respostas.

Os parâmetros de saída são o vetor b contendo os coeficientes da equação de regressão, sendo $u = b_1 + b_2 x_1 + b_3 x_2 + b_4 x_3 + \dots + b_p x_v$, o coeficiente de determinação $r2$, a variância residual $sigma2$ e a condição de erro $CondErro$. O valor $CondErro = 1$ indica que o modelo não é permitido ($v > 1$ e $v + 1 \neq p$), caso em que o algoritmo não pode ser executado.

O algoritmo necessita de uma rotina para fazer a decomposição em valores singulares da matriz X , cuja descrição está além do escopo deste texto. É recomendado o uso das robustas e eficientes rotinas da biblioteca LAPACK².

```

Algoritmo Regressão_linear_DVS
{ Objetivo: Calcular parâmetros de quadrados mínimos de modelo linear múltiplo }
{ via decomposição em valores singulares }
parâmetros de entrada n, v, p, x, y
{ número de pontos, número de variáveis, número de parâmetros, }
{ variáveis explicativas e variáveis respostas }
parâmetros de saída b, r2, sigma2, CondErro
{ coef. de regressão, coef. de determinação, variância residual e condição de erro }
se v > 1 e v + 1 ≠ p então CondErro ← 1, abandone, fimse
CondErro ← 0; vp1 ← v + 1; pm1 ← p - 1
para i ← 1 até n faça { inclusão de uma coluna de 1's relativa à b0 }
    para j ← vp1 até 2 passo -1 faça
        x(i,j) ← x(i,j - 1)
    fimpara
    x(i,1) ← 1
fimpara
se v = 1 e p > 2 então { se regressão polinomial, então gera potências de x }
    para j ← 2 até pm1 faça
        jp1 ← j + 1
        para i ← 1 até n faça x(i,jp1) ← x(i,2)j, fimpara
    fimpara
fimse
[U, S, V] ← dvs(x) { chamada da rotina para decomposição em valores singulares }
para i ← 1 até n faça { Cálculo do vetor auxiliar a = UTy }
    a(i) ← 0
    para j ← 1 até n faça
        a(i) ← a(i) + U(j,i) * y(j)
    fimpara
fimpara
para i ← 1 até p faça { Cálculo do vetor dos coeficientes b = VS1-1a1 }
    b(i) ← 0
    para j ← 1 até p faça
        b(i) ← b(i) + V(i,j)/S(j) * a(j)
    fimpara
fimpara
D ← 0 { Cálculo do desvio }
para i ← p + 1 até n faça, D ← D + a(i)2, fimpara
Sy ← 0; Sy2 ← 0 { Cálculo dos somatórios auxiliares }
para i ← 1 até n faça
    Sy ← Sy + y(i); Sy2 ← Sy2 + y(i)2
fimpara
r2 ← 1 - D/(Sy2 - Sy2/n) { coeficiente de determinação }
sigma2 ← D/(n - p) { variância residual }
finalgoritmo

```

Figura 4.7 Regressão linear múltipla e polinomial via decomposição em valores singulares.

²www.netlib.org

A complexidade computacional do algoritmo da Figura 4.7 é apresentada na Tabela 4.15. São mostradas, separadamente, as complexidades da regressão linear múltipla e da polinomial devido às diferenças dos dois métodos. A potenciação é tratada como multiplicações. As operações para fazer a decomposição em valores singulares não foram computadas.

Tabela 4.15 Complexidade da regressão via decomposição em valores singulares.

(n : número de pontos e p : número de parâmetros. Não foram consideradas as operações para decomposição em valores singulares.)

| Operações | Complexidade |
|----------------|------------------------------|
| adições | $n^2 + (p+2)n + p^2 - p + 5$ |
| multiplicações | $n^2 + 2n + p^2 - p + 1$ |
| divisões | $p^2 + 3$ |

a) Regressão linear múltipla

| Operações | Complexidade |
|----------------|--|
| adições | $n^2 + 4n + p^2 + 3$ |
| multiplicações | $n^2 + (\frac{1}{2}p^2 - \frac{3}{2}p + 3)n + p^2 - p + 1$ |
| divisões | $p^2 + 3$ |

b) Regressão polinomial

Exemplo 4.9 Ajustar os dados do Exemplo 4.4 ao modelo $u = b_0 + b_1x_1 + b_2x_2$ usando o algoritmo mostrado na Figura 4.7.

% Os parametros de entrada

```
n = 16
v = 2
p = 3
x =
 60.3000 108.0000
 61.1000 109.0000
 60.2000 110.0000
 61.2000 112.0000
 63.2000 112.0000
 63.6000 113.0000
 65.0000 115.0000
 63.8000 116.0000
 66.0000 117.0000
 67.9000 119.0000
 68.2000 120.0000
 66.5000 122.0000
 68.7000 123.0000
 69.6000 125.0000
 69.3000 128.0000
 70.6000 130.0000
```

y =

```
234
259
258
285
329
347
365
363
```

396
419
443
445
483
503
518
555

% produzem os resultados
coeficientes de regressao

```
B(0) = -1.40740e+03
B(1) = 1.34511e+01
B(2) = 7.80271e+00
coeficiente de determinacao = 0.99267
variancia residual = 8.37581e+01
condicao de erro = 0
```

A equação de quadrados mínimos é $u = -1,40740 \times 10^3 + 1,34511 \times 10^1 x_1 + 7,80271 x_2$, com $r^2 = 0,99267$ e $\sigma^2 = 83,7581$ (ver resultados obtidos pelo algoritmo da Figura 4.5). ■

4.4.4 Comparação dos métodos computacionais para RLM

A questão de usar ou não as equações normais na solução de problemas de quadrados mínimos é discutível. Existem alguns aspectos no método das equações normais que devem ser destacados. Primeiro, o número de condição da matriz $X^T X$ é o quadrado daquele da matriz X . Por conseguinte, formar as equações normais pode conduzir a um problema numérico mais difícil do que um que, simplesmente, usa X . É também difícil computar as matrizes $X^T X$ e $X^T y$, exatamente, usando qualquer procedimento. Por isto, as perturbações feitas no problema básico, pela formação das equações normais, podem ter consequências desastrosas.

A decisão básica é entre as superiores propriedades numéricas dos métodos de ortogonalização (decomposição em valores singulares e QR) e a maior velocidade com que as equações normais podem ser formadas e resolvidas. Utilizando *precisão simples* em uma linguagem de programação, a solução via ortogonalização é, provavelmente, a preferível e, usando *precisão dupla*, a velocidade de formação das equações normais torna seu uso mais atrativo, geralmente.

Muitos autores recomendam que as equações normais nunca devam ser formadas ou usadas. Defensores do uso das equações normais, por outro lado, argumentam que como os programas estatísticos geralmente usam *precisão dupla*, a diferença de exatidão dos dois métodos, poucas vezes, valerá a pena ser considerada.

Os métodos de ortogonalização são criticados por alguns com base em que eles requerem uma quantidade de memória muito grande, mesmo para problemas de tamanho moderado, e que a complexidade computacional é maior que a da decomposição de Cholesky usada para resolver as equações normais. Defensores dos métodos de ortogonalização exaltam suas estabilidades numéricas e rebatem que há grande quantidade de memória disponível nos computadores atuais a um custo relativamente baixo. A escolha entre os dois métodos é um assunto polêmico e até especialistas podem discordar mesmo em uma aplicação específica.

4.5 Diferença entre regressão e interpolação

Foi visto no Capítulo 3 que, qualquer que seja o polinômio interpolador de grau $n - 1$, ele é sempre construído de modo a passar pelos n pontos dados. Um polinômio escrito na forma

$$P_{n-1}(x) = a_0 + a_1x + a_2x^2 + \dots + a_{n-1}x^{n-1}$$

tem sempre n coeficientes a_i , $i = 0, 1, \dots, n - 1$, isto é, o número de pontos utilizados para gerar o polinômio interpolador é igual ao número de coeficientes do polinômio.

Por outro lado, em uma regressão polinomial utilizando os mesmos n pontos, pode-se construir um polinômio de grau g

$$U_g(x) = b_0 + b_1x + b_2x^2 + \dots + b_gx^g,$$

tal que $g \leq n - 1$. Quando existir a igualdade $g = n - 1$, então o polinômio de regressão será idêntico ao polinômio interpolador. Para exemplificar, seja um polinômio interpolador de grau $g = 1$ que passa por $n = 2$ pontos (x_1, y_1) e (x_2, y_2) , conforme mostrado na Seção 3.1.1, cujos coeficientes são obtidos pela solução do sistema linear

$$\begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}.$$

Pré-multiplicando ambos os lados da igualdade acima pela transposta da matriz, tem-se

$$\begin{bmatrix} 1 & 1 \\ x_1 & x_2 \end{bmatrix} \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ x_1 & x_2 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \sim$$

$$\begin{bmatrix} 2 & x_1 + x_2 \\ x_1 + x_2 & x_1^2 + x_2^2 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \end{bmatrix} = \begin{bmatrix} y_1 + y_2 \\ x_1y_1 + x_2y_2 \end{bmatrix}.$$

O sistema linear acima é idêntico às equações normais (4.1), para $n = 2$, utilizadas para calcular os parâmetros de uma regressão linear simples. A Figura 4.8(a) mostra um polinômio de regressão de grau $g = 2$, obtido a partir dos dados da Tabela 4.1 com $n = 5$ pontos. Quando o polinômio de regressão possuir grau $g = n - 1 = 4$, então ele se torna idêntico a um polinômio interpolador de mesmo grau. A Figura 4.8(b) mostra que, neste caso, o polinômio passa por todos os pontos do diagrama de dispersão.

Em termos de complexidade computacional, a interpolação é um processo mais simples que a regressão polinomial, sobretudo se for evitada a solução de um sistema linear. A interpolação deve ser utilizada quando se necessita de um valor intermediário não constante de uma tabela. Por sua vez, a regressão tem que ser utilizada quando se deseja estimar um parâmetro de um modelo semideterminístico e/ou prever um valor dado por esse modelo.

É importante notar que a variância residual σ^2 , dada por (4.13), torna-se indefinida quando o número de parâmetros p do modelo for igual ao número de pontos n .

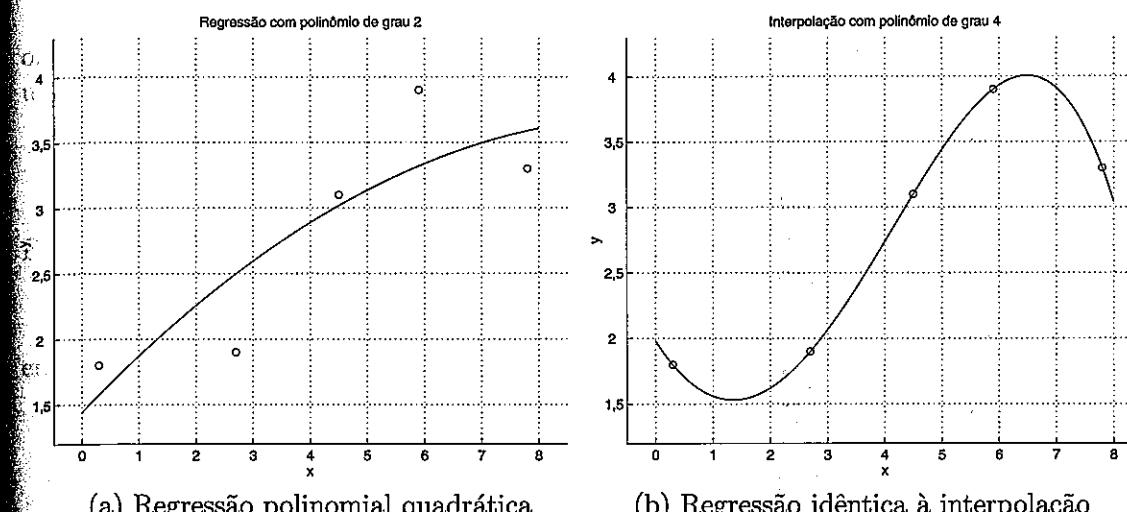


Figura 4.8 Diferença entre regressão e interpolação.

4.6 Exemplos de aplicação

Para exemplificar o uso de ajuste de curvas no cálculo de parâmetros de relações semideterminísticas, serão apresentadas aplicações no estudo de tensão-deformação de uma barra de aço e sobre o produto iônico da água.

4.6.1 Tensão-deformação de aço

Definição do problema

Calcular o módulo de Young de uma barra de aço a partir de valores de tensão t ($\text{ton} \times \text{cm}^{-2}$) e deformação d (adimensional) constantes da Figura 4.9(a).

Modelagem matemática

A Figura 4.9(b) mostra o diagrama de dispersão dos dados da Figura 4.9(a). Os dados apresentados se referem à fase linear do processo, e, portanto, podem-se calcular os parâmetros da relação linear entre a deformação d e a tensão t

$$t = E \times d + t_0,$$

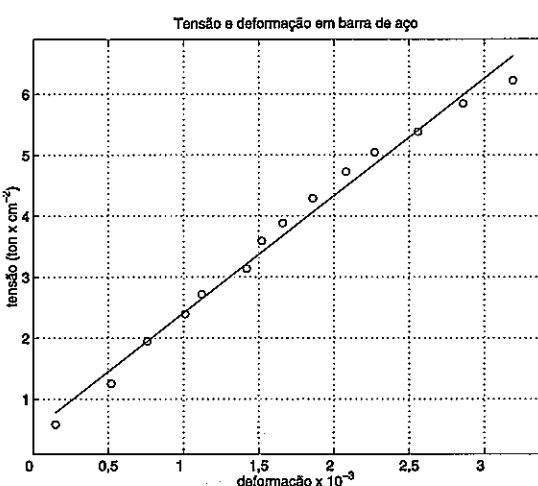
onde E é o módulo de Young e t_0 é a pré-tensão.

Solução numérica

Utilizando os dados da Figura 4.9(a) no algoritmo da Figura 4.5, obtém-se os resultados

coeficientes de regressão
 $b(0) = 4.89122\text{e-}01$
 $b(1) = 1.92225\text{e+}03$

| i | $d \times 10^3$ | t (ton \times cm $^{-2}$) |
|-----|-----------------|--------------------------------|
| 1 | 0,15 | 0,586 |
| 2 | 0,52 | 1,253 |
| 3 | 0,76 | 1,946 |
| 4 | 1,01 | 2,394 |
| 5 | 1,12 | 2,716 |
| 6 | 1,42 | 3,136 |
| 7 | 1,52 | 3,591 |
| 8 | 1,66 | 3,885 |
| 9 | 1,86 | 4,291 |
| 10 | 2,08 | 4,725 |
| 11 | 2,27 | 5,047 |
| 12 | 2,56 | 5,383 |
| 13 | 2,86 | 5,845 |
| 14 | 3,19 | 6,223 |



(a) Deformação e tensão.

(b) Fase linear da deformação \times tensão.

Figura 4.9 Tensão e deformação de uma barra de aço.

(d é a deformação e t a tensão, conforme [11, Tabela 7.4].)

coeficiente de determinacao = 0.98697
variancia residual = 4.20951e-02
condicao de erro = 0

Análise dos resultados

Pelos resultados acima, o módulo de Young é $E = 1,92225 \times 10^3$ e a pré-tensão é $t_0 = 4,89122 \times 10^{-1}$ ton \times cm $^{-2}$. O alto valor do coeficiente de determinação $r^2 = 0,98697$ confirma a adequação do modelo $t = E \times d + t_0$ aos dados. Deve ser mencionado que a relação é não linear a partir de certo valor da deformação, tornando a análise mais complexa [11].

4.6.2 Produto iônico da água

Definição do problema

Como as considerações teóricas não são capazes de fornecer uma relação exata entre o produto iônico da água K_w e a temperatura T , o que se deseja é encontrar uma relação semideterminística [15].

Modelagem matemática

Seja a equação de Gibbs-Helmholtz da Termodinâmica

$$d(\log_{10}(K_w)) = \frac{\Delta H^0}{\log_e(10)RT^2} dT, \quad (4.20)$$

onde K_w é o produto iônico da água, ΔH^0 a variação de entalpia-padrão, R a constante universal dos gases e T a temperatura. Integrando (4.20), tem-se

$$\int_{\log_{10}(K_w)_0}^{\log_{10}(K_w)} d(\log_{10}(K_w)) = \frac{1}{R \log_e(10)} \int_{T_0}^T \frac{\Delta H^0}{T^2} dT.$$

Expressando ΔH^0 como uma série de potências em T , obtém-se

$$\Delta H^0 = a_0 + a_1 T + a_2 T^2 + a_3 T^3 + \dots,$$

então

$$\log_{10}(K_w) = (\log_{10}(K_w))_0 + \frac{1}{R \log_e(10)} \int_{T_0}^T \left(\frac{a_0}{T^2} + \frac{a_1 T}{T^2} + \frac{a_2 T^2}{T^2} + \dots \right) dT.$$

Supondo que a série $\frac{a_0}{T^2} + \frac{a_1}{T} + a_2 + a_3 T + \dots$, converja uniformemente no intervalo $[T_0, T]$, tem-se

$$\begin{aligned} \log_{10}(K_w) &= (\log_{10}(K_w))_0 + \frac{1}{R \log_e(10)} \left(-a_0 \left(\frac{1}{T} - \frac{1}{T_0} \right) + a_1 \log_e \left(\frac{T}{T_0} \right) + \right. \\ &\quad \left. + a_2(T - T_0) + \frac{a_3}{2}(T^2 - T_0^2) + \dots \right) \end{aligned}$$

Esta equação pode ser reescrita na forma

$$-\log_{10}(K_w) = b_0 + b_1 \frac{1}{T} + b_2 \log_e(T) + b_3 T + b_4 T^2 + \dots \quad (4.21)$$

A questão agora é definir como truncar a expressão (4.21) de modo a obter um bom ajuste. Serão testados quatro modelos

$$\text{modelo 1: } -\log_{10}(K_w) = b_0 + b_1 \frac{1}{T},$$

$$\text{modelo 2: } -\log_{10}(K_w) = b_0 + b_1 \frac{1}{T} + b_2 \log_e(T),$$

$$\text{modelo 3: } -\log_{10}(K_w) = b_0 + b_1 \frac{1}{T} + b_2 \log_e(T) + b_3 T \text{ e}$$

$$\text{modelo 4: } -\log_{10}(K_w) = b_0 + b_1 \frac{1}{T} + b_2 \log_e(T) + b_3 T + b_4 T^2.$$

Solução numérica

Sejam os dados de $-\log_{10}(K_w) \times t$, $0 \leq t \leq 60^\circ\text{C}$, mostrados na Tabela 4.16. Utilizando o algoritmo da Figura 4.7, obtém-se os resultados compilados na Tabela 4.17.

Tabela 4.16 Produto iônico da água em função da temperatura.

(K_w é o produto iônico da água e t a temperatura ($^{\circ}\text{C}$), conforme [21, Tabela II].)

| i | t_i | $-\log_{10}(K_w)_i$ |
|-----|-------|---------------------|
| 1 | 0 | 14,9435 |
| 2 | 5 | 14,7338 |
| 3 | 10 | 14,5346 |
| 4 | 15 | 14,3463 |
| 5 | 20 | 14,1669 |
| 6 | 25 | 13,9965 |
| 7 | 30 | 13,8330 |
| 8 | 35 | 13,6801 |
| 9 | 40 | 13,5348 |
| 10 | 45 | 13,3960 |
| 11 | 50 | 13,2617 |
| 12 | 55 | 13,1369 |
| 13 | 60 | 13,0171 |

Tabela 4.17 Escolha do modelo de $-\log_{10}(K_w)$ em função da temperatura.

(m é o modelo e T a temperatura em Kelvin ($T = 273,15 + t$)).

| m | b_0 | b_1 | b_2 | b_3 | b_4 | $1 - r^2$ | σ^2 |
|-----|-----------------------|-----------------------|-----------------------|--------------------------|--------------------------|-----------------------|-----------------------|
| 1 | $4,2216 \times 10^0$ | $2,9201 \times 10^3$ | | | | $8,58 \times 10^{-4}$ | $3,65 \times 10^{-4}$ |
| 2 | $-6,5127 \times 10^1$ | $6,0342 \times 10^3$ | $1,0335 \times 10^1$ | | | $9,86 \times 10^{-7}$ | $4,61 \times 10^{-7}$ |
| 3 | $-7,1817 \times 10^1$ | $6,2107 \times 10^3$ | $1,1507 \times 10^1$ | $-1,9400 \times 10^{-3}$ | | $9,83 \times 10^{-7}$ | $5,11 \times 10^{-7}$ |
| 4 | $3,5736 \times 10^2$ | $-2,0656 \times 10^3$ | $-7,0883 \times 10^1$ | $2,7101 \times 10^{-1}$ | $-1,5047 \times 10^{-4}$ | $9,72 \times 10^{-7}$ | $5,68 \times 10^{-7}$ |

Análise dos resultados

Pelos resultados de $1 - r^2$ e σ^2 da Tabela 4.17, pode-se notar que a melhoria do modelo 2 em relação ao modelo 1 é significativa, pois tanto $1 - r^2$ quanto a variância residual σ^2 reduziram. No entanto, essa melhoria não é percebida quando se usam os modelos 3 e 4, porque apesar de $1 - r^2$ reduzir um pouco, o valor de σ^2 aumenta a partir do modelo 2. Portanto, o modelo 2

$$-\log_{10}(K_w) = -6,5127 \times 10^1 + 6,0342 \times 10^3 \frac{1}{T} + 1,0335 \times 10^1 \log_e(T)$$

é o melhor, visto que consegue o mesmo nível de qualidade no ajuste usando um menor número de parâmetros.

4.7 Exercícios

Seção 4.1

Seja a tabela

| i | x_i | y_i |
|-----|-------|-------|
| 1 | 0,5 | 5,1 |
| 2 | 1,2 | 3,2 |
| 3 | 2,1 | 2,8 |
| 4 | 3,5 | 1,0 |
| 5 | 5,4 | 0,4 |

4.1. Fazer o diagrama de dispersão.

4.2. Determinar o polinômio de grau 1 que passa pelo primeiro e segundo pontos e calcular o desvio D .

4.3. Achar o polinômio de grau 1 que passa pelo terceiro e quinto pontos e calcular D .

4.4. Encontrar a reta de quadrados mínimos usando os cinco pontos da tabela e calcular D .

4.5. Verificar qual dos três modelos acima (4.2, 4.3 e 4.4) é o melhor.

Seção 4.2

4.6. Dada a tabela, calcular o coeficiente de determinação r^2 do modelo $u = b_0 + b_1x$

| i | x_i | y_i |
|-----|-------|-------|
| 1 | 1,4 | 4,2 |
| 2 | 2,1 | 2,3 |
| 3 | 3,0 | 1,9 |
| 4 | 4,4 | 1,1 |

4.7. Determinar a variância residual σ^2 do modelo acima.

4.8. Calcular o coeficiente de determinação e a variância residual do modelo linear $u = b_0 + b_1x$ a partir dos pontos da Tabela 4.5.

4.9. Usando os dados (Ano, Urbana) da Tabela 4.11, calcular o coeficiente de determinação e a variância residual do modelo $u = b_0 + b_1x$.

4.10. O que mede o coeficiente de determinação?

Seção 4.3

4.11. Achar a equação de quadrados mínimos $u = b_0 + b_1x_1 + b_2x_2$ a partir dos pontos da tabela

| i | x_{i1} | x_{i2} | y_i |
|-----|----------|----------|-------|
| 1 | -1 | -2 | 13 |
| 2 | 0 | -1 | 11 |
| 3 | 1 | 0 | 9 |
| 4 | 2 | 1 | 4 |
| 5 | 4 | 1 | 11 |
| 6 | 5 | 2 | 9 |
| 7 | 5 | 3 | 1 |
| 8 | 6 | 4 | -1 |

4.12. Calcular os coeficientes do modelo $u = b_0 + b_1x + b_2x^2$ usando os pontos

| i | x_i | y_i |
|-----|-------|-------|
| 1 | -2,0 | -30,5 |
| 2 | -1,5 | -20,2 |
| 3 | 0,0 | -3,3 |
| 4 | 1,0 | 8,9 |
| 5 | 2,2 | 16,8 |
| 6 | 3,1 | 21,4 |

4.13. Implementar, em qualquer linguagem de programação, o algoritmo de regressão linear múltipla e polinomial da Figura 4.5.

4.14. Determinar os parâmetros do modelo do Exercício 4.11 usando o programa implementado no Exercício 4.13.

4.15. Achar os coeficientes do modelo do Exercício 4.12 com o programa do Exercício 4.13.

Seção 4.4

4.16. Implementar a decomposição em valores singulares em uma linguagem de programação.

4.17. Usando a mesma linguagem do exercício anterior, implementar o algoritmo de regressão linear múltipla e polinomial via decomposição em valores singulares mostrado na Figura 4.7.

4.18. Resolver o Exercício 4.11 usando o programa do Exercício 4.17.

4.19. Resolver o Exercício 4.12 utilizando o programa do Exercício 4.17.

4.20. Refazer o Exemplo 4.6 usando o programa do Exercício 4.17 e comparar com os valores da Tabela 4.12.

Seção 4.5

Seja a tabela

| i | x_i | y_i |
|---|-------|-------|
| 1 | 1 | 5 |
| 2 | 3 | 11 |

4.21. Determinar o polinômio interpolador de grau 1 que passa pelos pontos da tabela acima.

4.22. Calcular a reta de quadrados mínimos usando os pontos da tabela acima.

4.23. Os resultados dos Exercícios 4.21 e 4.22 são iguais? Por quê?

4.24. Mostrar a igualdade da interpolação e regressão para um polinômio de grau 2 utilizando 3 pontos.

4.25. Em termos de utilização, qual a diferença entre interpolação e regressão?

Gerais

4.26. Transformar os modelos abaixo em relações lineares

a) $y = \frac{a}{b+cx}$,

b) $y = ab^x$,

c) $y = \frac{a}{b+x}$,

d) $y = \frac{1}{1+e^{bx}}$.

4.27. Deduzir as equações normais para o modelo $u = b_1x_1 + b_2x_2 + b_3x_3$, que passa pela origem ($b_0 = 0$).

4.28. Resolver o Exemplo 4.5 usando $u = b_1x + b_2x^2 + b_3x^3$ e comparar os coeficientes com o do modelo com $b_0 \neq 0$. Cumpre salientar que

as expressões do coeficiente de determinação r^2 e da variância residual σ^2 são diferentes para modelos com $b_0 = 0$ (ver [13]).

4.29. Dada a tabela, qual dos modelos abaixo é o melhor? Por quê?

| i | x_i | y_i |
|----|-------|-------|
| 1 | 5 | 0,01 |
| 2 | 6 | 0,05 |
| 3 | 7 | 0,08 |
| 4 | 8 | 0,14 |
| 5 | 9 | 0,18 |
| 6 | 10 | 0,26 |
| 7 | 11 | 0,44 |
| 8 | 12 | 0,51 |
| 9 | 13 | 0,79 |
| 10 | 14 | 1,02 |

a) $u = b_0 + b_1x$,

b) $u = b_0 + b_1x + b_2x^2$,

c) $u = b_0 + b_1x + b_2x^2 + b_3x^3$,

d) $u = ax^b$,

e) $u = ab^x$.

4.30. Seja o tempo de germinação de sementes (dias) em função da temperatura média do solo ($^{\circ}\text{C}$) para doze locais de plantio

| Temperatura ($^{\circ}\text{C}$) | Germinação (dias) |
|------------------------------------|-------------------|
| 14 | 10 |
| 6 | 26 |
| 3 | 41 |
| 6 | 29 |
| 7 | 27 |
| 6 | 27 |
| 7 | 19 |
| 4 | 28 |
| 8 | 19 |
| 7 | 31 |
| 6 | 29 |
| 4 | 33 |

Determinar uma relação entre a temperatura e o tempo de germinação das sementes.

Capítulo 5

Integração numérica

Seja uma função $f(x)$ integrável no intervalo $[a, b]$, então

$$\int_a^b f(x) dx = F(b) - F(a), \text{ onde } F'(x) = f(x).$$

Quando a forma analítica de $F(x)$ for de difícil obtenção ou se forem conhecidos somente valores discretos de $f(x)$, se faz necessário o uso de métodos numéricos para avaliar a integral de $f(x)$. Esses métodos consistem em aproximar a função $f(x)$ por um polinômio interpolador e determinar analiticamente a integral desse polinômio no intervalo $[a, b]$.

Neste capítulo serão abordadas duas classes de métodos para integração numérica: as fórmulas de Newton-Cotes e a quadratura de Gauss-Legendre. Elas serão utilizadas tanto para o cálculo de integrais simples quanto para integrais duplas.

5.1 Fórmulas de Newton-Cotes

Seja uma função $f(x)$ aproximada por um polinômio interpolador, por exemplo, um polinômio de Gregory-Newton (3.12)

$$f(x) \approx P_n(x) = y_0 + \sum_{i=1}^n \frac{\Delta^i y_0}{i!} \prod_{j=0}^{i-1} (u_x - j), \text{ onde } u_x = \frac{x - x_0}{h}. \quad (5.1)$$

Para (5.1) com $n = 1$, isto é, utilizando um polinômio interpolador de grau 1, tem-se

$$\int_a^b f(x) dx \approx \int_{a=x_0}^{b=x_1} P_1(x) dx.$$

Fazendo mudança de variável de $x \rightarrow u_x$ e simplificando a notação de $u_x \rightarrow u$, obtém-se

$$x = a = x_0 \longrightarrow u = \frac{x_0 - x_0}{h} \sim u = 0 \text{ e}$$

$$x = b = x_1 \rightarrow u = \frac{x_1 - x_0}{h} = \frac{h}{h} \rightsquigarrow u = 1$$

e usando a notação $y_i = f(x_i)$, a equação de integração resulta em

$$I_1 = \int_{a=x_0}^{b=x_1} P_1(x) dx = \int_0^1 (y_0 + u\Delta y_0) h du.$$

Integrando, analiticamente, este polinômio de grau 1 em relação a u , tem-se que

$$I_1 = h \left[y_0 u + \frac{u^2}{2} \Delta y_0 \right]_0^1 = h \left(y_0 + \frac{y_1 - y_0}{2} \right) = \frac{h}{2} (2y_0 + y_1 - y_0),$$

$$I_1 = \frac{h}{2} (y_0 + y_1),$$

(5.2)

que é conhecida como a regra do trapézio.

Exemplo 5.1 Calcular $\int_1^7 \frac{1}{x} dx$ pela regra do trapézio (5.2).

O polinômio de grau 1 passa pelos pontos com abscissas $a = x_0 = 1$ e $b = x_1 = 7$, assim

$$h = 7 - 1 = 6 \rightarrow I_1 = \frac{6}{2} \left(\frac{1}{1} + \frac{1}{7} \right) \rightsquigarrow I_1 = 3,4286.$$

A Figura 5.1 mostra a aproximação da função $f(x) = 1/x$ por um polinômio interpolador $P(x)$ de grau 1.

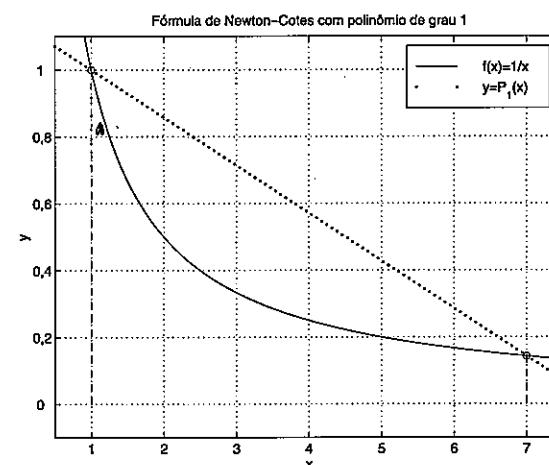


Figura 5.1 Integração numérica pela regra do trapézio.

Aproximando agora $f(x)$ por um polinômio interpolador $P_2(x)$ de grau 2, ou seja, para (5.1) com $n = 2$, tem-se

$$\int_a^b f(x) dx \approx \int_{a=x_0}^{b=x_2} P_2(x) dx.$$

Fazendo mudança de variável, obtém-se

$$x = a = x_0 \rightarrow u = \frac{x_0 - x_0}{h} \rightsquigarrow u = 0 \text{ e}$$

$$x = b = x_2 \rightarrow u = \frac{x_2 - x_0}{h} = \frac{2h}{h} = 2 \rightsquigarrow u = 2$$

e com a notação $y_i = f(x_i)$, a equação de integração resulta em

$$I_2 = \int_{a=x_0}^{b=x_2} P_2(x) dx = \int_0^2 \left(y_0 + u\Delta y_0 + \frac{u^2 - u}{2} \Delta^2 y_0 \right) h du.$$

Integrando, analiticamente, este polinômio de grau 2 em relação a u , tem-se que

$$I_2 = h \left[y_0 u + \frac{u^2}{2} \Delta y_0 + \left(\frac{u^3}{6} - \frac{u^2}{4} \right) \Delta^2 y_0 \right]_0^2,$$

$$I_2 = h \left[2y_0 + 2(y_1 - y_0) + \frac{1}{3}(y_2 - 2y_1 + y_0) \right],$$

$$I_2 = \frac{h}{3} (y_0 + 4y_1 + y_2),$$

(5.3)

que é a regra do 1/3 de Simpson ou primeira regra de Simpson.

Exemplo 5.2 Calcular $\int_1^7 \frac{1}{x} dx$, usando a regra do 1/3 de Simpson (5.3).

Para construir um polinômio de grau 2 são necessários 3 pontos; portanto, as abscissas por onde o polinômio irá passar são $a = x_0 = 1$, $x_1 = 4$ e $b = x_2 = 7$. Assim,

$$h = \frac{7 - 1}{2} = 3 \rightarrow I_2 = \frac{3}{3} \left(\frac{1}{1} + 4 \frac{1}{4} + \frac{1}{7} \right) \rightsquigarrow I_2 = 2,1429.$$

A aproximação da função $f(x) = 1/x$ por um polinômio interpolador $P(x)$ de grau 2 é exibida na Figura 5.2.

Se $f(x)$ for aproximada por um polinômio interpolador de grau 3, então

$$\int_a^b f(x) dx \approx \int_{a=x_0}^{b=x_3} P_3(x) dx.$$

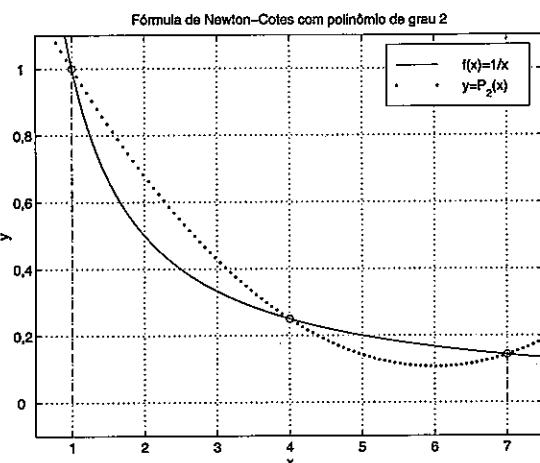


Figura 5.2 Integração numérica pela regra do 1/3 de Simpson.

Fazendo mudança de variável, tem-se que

$$x = a = x_0 \rightarrow u = \frac{x_0 - x_0}{h} \rightsquigarrow u = 0 \text{ e}$$

$$x = b = x_3 \rightarrow u = \frac{x_3 - x_0}{h} = \frac{3h}{h} \rightsquigarrow u = 3$$

e com $y_i = f(x_i)$, a equação de integração resulta em

$$I_3 = \int_{a=x_0}^{b=x_3} P_3(x) dx,$$

$$I_3 = \int_0^3 \left(y_0 + u\Delta y_0 + \frac{u^2 - u}{2} \Delta^2 y_0 + \frac{u^3 - 3u^2 + 2u}{6} \Delta^3 y_0 \right) h du.$$

Integrando, analiticamente, este polinômio de grau 3 em relação a u , tem-se que

$$I_3 = h \left[y_0 u + \frac{u^2}{2} \Delta y_0 + \left(\frac{u^3}{6} - \frac{u^2}{4} \right) \Delta^2 y_0 + \left(\frac{u^4}{24} - \frac{u^3}{6} + \frac{u^2}{6} \right) \Delta^3 y_0 \right]_0^3,$$

$$I_3 = h \left[3y_0 + \frac{9}{2}(y_1 - y_0) + \frac{9}{4}(y_2 - 2y_1 + y_0) + \frac{3}{8}(y_3 - 3y_2 + 3y_1 - y_0) \right],$$

$$I_3 = \frac{3h}{8} (y_0 + 3y_1 + 3y_2 + y_3), \quad (5.4)$$

que é a regra dos 3/8 de Simpson ou segunda regra de Simpson.

Exemplo 5.3 Calcular $\int_1^7 \frac{1}{x} dx$ pela regra dos 3/8 de Simpson (5.4).

São necessários 4 pontos para construir um polinômio de grau 3; consequentemente, $a = x_0 = 1$, $x_1 = 3$, $x_2 = 5$ e $b = x_3 = 7$. Por isso,

$$h = \frac{7 - 1}{3} = 2 \rightarrow I_3 = \frac{3 \times 2}{8} \left(\frac{1}{1} + 3 \frac{1}{3} + 3 \frac{1}{5} + \frac{1}{7} \right) \rightsquigarrow I_3 = 2,0571.$$

A Figura 5.3 ilustra a aproximação da função $f(x) = 1/x$ por um polinômio interpolador $P_3(x)$ de grau 3. ■

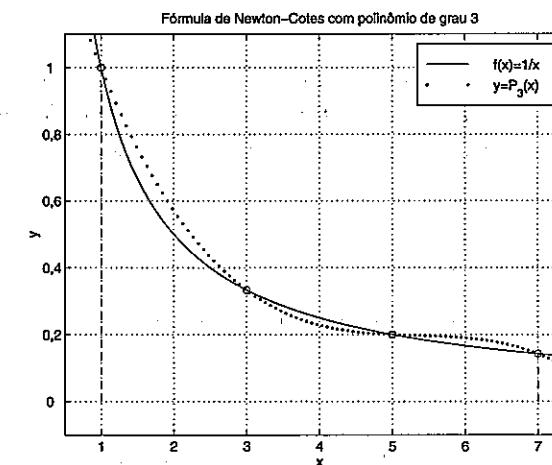


Figura 5.3 Integração numérica pela regra dos 3/8 de Simpson.

Considerando que $\int_1^7 \frac{1}{x} dx = \log_e(x) \Big|_1^7 = \log_e(7) \approx 1,9459$, deve ser observado que o resultado da integração melhora à medida que o grau do polinômio interpolador aumenta.

| n | I_n | $ I_n - \log_e(7) $ |
|-----|--------|---------------------|
| 1 | 3,4286 | 1,4827 |
| 2 | 2,1429 | 0,1970 |
| 3 | 2,0571 | 0,1112 |

Comparando (5.2), (5.3) e (5.4), nota-se que as fórmulas de Newton-Cotes apresentam a forma geral

$$I_n = \frac{nh}{d_n} \sum_{i=0}^n c_i y_i, \quad (5.5)$$

Onde c_i são chamados de coeficientes de Cotes. A Tabela 5.1 apresenta d_n e os coeficientes de Cotes para $n = 1, 2, \dots, 8$, conforme Demidovich e Maron [10, Tabela 65].

Tabela 5.1 Coeficientes de Cotes.

| n | d_n | c_0 | c_1 | c_2 | c_3 | c_4 | c_5 | c_6 | c_7 | c_8 |
|-----|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| 1 | 2 | 1 | 1 | | | | | | | |
| 2 | 6 | 1 | 4 | 1 | | | | | | |
| 3 | 8 | 1 | 3 | 3 | 1 | | | | | |
| 4 | 90 | 7 | 32 | 12 | 32 | 7 | | | | |
| 5 | 288 | 19 | 75 | 50 | 50 | 75 | 19 | | | |
| 6 | 840 | 41 | 216 | 27 | 272 | 27 | 216 | 41 | | |
| 7 | 17280 | 751 | 3577 | 1323 | 2989 | 2989 | 1323 | 3577 | 751 | |
| 8 | 28350 | 989 | 5888 | -928 | 10496 | -4540 | 10496 | -928 | 5888 | 989 |

Na prática, dificilmente é usado um polinômio de grau superior a 3 para a integração numérica. O resultado é melhorado pela subdivisão do intervalo de integração e subseqüente aplicação de uma fórmula de Newton-Cotes em cada subintervalo, resultando nos três métodos compostos que serão vistos a seguir.

5.1.1 Regra do trapézio

Seja (5.2) a fórmula de integração que utiliza um polinômio interpolador de grau 1

$$I_1 = \frac{h}{2}(y_0 + y_1).$$

Subdividindo o intervalo $[a, b]$ em m subintervalos iguais e aplicando a equação acima a cada 2 pontos, tem-se

$$\int_{a=x_0}^{b=x_m} f(x) dx \approx I_1, \text{ com}$$

$$I_1 = \frac{h}{2}(y_0 + y_1) + \frac{h}{2}(y_1 + y_2) + \frac{h}{2}(y_2 + y_3) + \cdots + \frac{h}{2}(y_{m-1} + y_m),$$

$$I_1 = \frac{h}{2}(y_0 + 2y_1 + 2y_2 + \cdots + 2y_{m-1} + y_m),$$

$$I_1 = \frac{h}{2} \sum_{i=0}^m c_i y_i, \quad (5.6)$$

com $c_0 = c_m = 1$ e $c_i = 2$, $i = 1, 2, \dots, m-1$. Para a regra do trapézio composta, pode ser utilizado qualquer valor de número de subintervalos m .

A Figura 5.4 mostra a integração da função $f(x) = e^x \sin(10x) + 8$ utilizando 6 polinômios interpoladores $P(x)$ de grau 1.

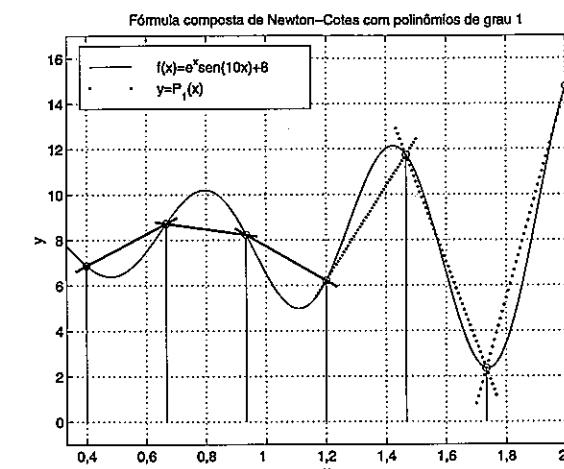


Figura 5.4 Integração numérica pela regra do trapézio composta.

Exemplo 5.4 Calcular $\int_1^3 x^3 \log_e(x) dx$ pela regra do trapézio composta (5.6) com $m = 4$ subintervalos.

$$h = \frac{b-a}{m} = \frac{3-1}{4} \rightarrow h = 0,5.$$

Pode ser construído um dispositivo prático formado por quatro colunas, com $i = 0, 1, \dots, m$, $x_i = a, a+h, a+2h, \dots, b$, $y_i = f(x_i)$ e c_i sendo os coeficientes de Cotes

| i | x_i | y_i | c_i |
|-----|-------|---------|-------|
| 0 | 1,0 | 0,0000 | 1 |
| 1 | 1,5 | 1,3684 | 2 |
| 2 | 2,0 | 5,5452 | 2 |
| 3 | 2,5 | 14,3170 | 2 |
| 4 | 3,0 | 29,6625 | 1 |

$$I_1 = \frac{0,5}{2}(0,0000 + 2(1,3684 + 5,5452 + 14,3170) + 29,6625) \rightarrow I_1 = 18,0309. \blacksquare$$

Exemplo 5.5 Calcular $\int_0^2 \frac{e^{-\cos(x)}}{\sqrt{2x+4}} dx$ pela regra do trapézio composta (5.6) com $m = 5$ subintervalos.

$$h = \frac{b-a}{m} = \frac{2-0}{5} \rightarrow h = 0,4$$

| i | x_i | y_i | c_i |
|-----|-------|--------|-------|
| 0 | 0,0 | 0,1839 | 1 |
| 1 | 0,4 | 0,1817 | 2 |
| 2 | 0,8 | 0,2105 | 2 |
| 3 | 1,2 | 0,2751 | 2 |
| 4 | 1,6 | 0,3837 | 2 |
| 5 | 2,0 | 0,5360 | 1 |

$$I_1 = \frac{0,4}{2}(0,1839 + 2(0,1817+0,2105+0,2751+0,3837) + 0,5360) \sim I_1 = 0,5644.$$

5.1.2 Regra do 1/3 de Simpson

Considere (5.3) baseada em um polinômio interpolador de grau 2,

$$I_2 = \frac{h}{3}(y_0 + 4y_1 + y_2).$$

Subdividindo o intervalo $[a, b]$ em m (múltiplo de 2) subintervalos iguais e aplicando a equação acima a cada 3 pontos, tem-se

$$\int_{a=x_0}^{b=x_m} f(x) dx \approx I_2, \text{ com}$$

$$I_2 = \frac{h}{3}(y_0 + 4y_1 + y_2) + \frac{h}{3}(y_2 + 4y_3 + y_4) + \cdots + \frac{h}{3}(y_{m-2} + 4y_{m-1} + y_m),$$

$$I_2 = \frac{h}{3}(y_0 + 4y_1 + 2y_2 + 4y_3 + 2y_4 + \cdots + 2y_{m-2} + 4y_{m-1} + y_m),$$

$$I_2 = \frac{h}{3} \sum_{i=0}^m c_i y_i, \quad (5.7)$$

com $c_0 = c_m = 1$, $c_i = 4$ se i for ímpar e $c_i = 2$ se i for par. Para a regra do 1/3 de Simpson composta, o número de subintervalos m deve ser múltiplo de 2, que é o grau do polinômio interpolador usado pela regra. A Figura 5.5 ilustra a integração da função $f(x) = e^x \sin(10x) + 8$ utilizando 3 polinômios interpoladores $P(x)$ de grau 2.

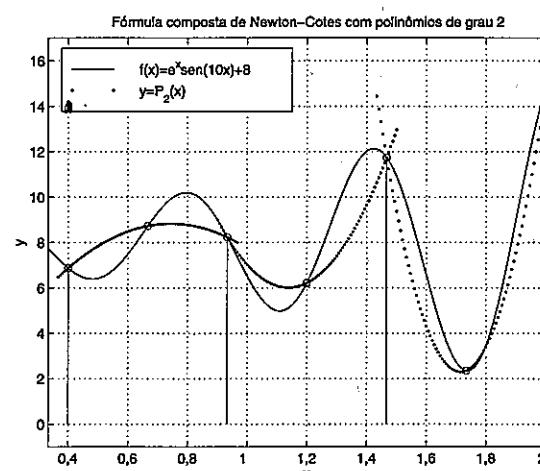


Figura 5.5 Integração numérica pela regra do 1/3 de Simpson composta.

Exemplo 5.6 Verificar que $\pi = 4 \int_0^1 \frac{1}{1+x^2} dx$ usando a regra do 1/3 de Simpson composta (5.7) com passo de integração $h = 0,25$.

$$m = \frac{b-a}{h} = \frac{1-0}{0,25} \rightarrow m = 4 \text{ (múltiplo de 2)}.$$

No dispositivo prático, tem-se que $i = 0, 1, \dots, m$, $x_i = a, a+h, a+2h, \dots, b$, $y_i = f(x_i)$ e c_i são os coeficientes de Cotes para a regra do 1/3 de Simpson composta

| i | x_i | y_i | c_i |
|-----|-------|--------|-------|
| 0 | 0,00 | 1,0000 | 1 |
| 1 | 0,25 | 0,9412 | 4 |
| 2 | 0,50 | 0,8000 | 2 |
| 3 | 0,75 | 0,6400 | 4 |
| 4 | 1,00 | 0,5000 | 1 |

$$I_2 = \frac{0,25}{3}(1,0000 + 4(0,9412 + 0,6400) + 2(0,8000) + 0,5000) \sim I_2 = 0,7854 \text{ e}$$

$$4 \times I_2 = 3,1416 \approx \pi.$$

Exemplo 5.7 Calcular $\int_0^3 \frac{xe^{2x}}{(1+2x)^2} dx$ usando a regra do 1/3 de Simpson composta (5.7) com $m = 6$ (múltiplo de 2) subintervalos.

$$h = \frac{b-a}{m} = \frac{3-0}{6} \rightarrow h = 0,5$$

| i | x_i | y_i | c_i |
|-----|-------|---------|-------|
| 0 | 0,0 | 0,0000 | 1 |
| 1 | 0,5 | 0,3398 | 4 |
| 2 | 1,0 | 0,8210 | 2 |
| 3 | 1,5 | 1,8830 | 4 |
| 4 | 2,0 | 4,3679 | 2 |
| 5 | 2,5 | 10,3065 | 4 |
| 6 | 3,0 | 24,6997 | 1 |

$$I_2 = \frac{0,5}{3}(0,0000 + 4(0,3398 + 1,8830 + 10,3065) + 2(0,8210 + 4,3679) + 24,6997) \sim$$

$$I_2 = 14,1991.$$

5.1.3 Regra dos 3/8 de Simpson

Seja (5.4) baseada em um polinômio interpolador de grau 3,

$$I_3 = \frac{3h}{8}(y_0 + 3y_1 + 3y_2 + y_3).$$

Subdividindo o intervalo $[a, b]$ em m (múltiplo de 3) subintervalos iguais e aplicando a equação acima a cada 4 pontos, tem-se

$$\begin{aligned} \int_{a=x_0}^{b=x_m} f(x) dx &\approx I_3 = \frac{3h}{8}(y_0 + 3y_1 + 3y_2 + y_3) + \\ &\quad \frac{3h}{8}(y_3 + 3y_4 + 3y_5 + y_6) + \cdots + \frac{3h}{8}(y_{m-3} + 3y_{m-2} + 3y_{m-1} + y_m), \\ I_3 &= \frac{3h}{8}(y_0 + 3y_1 + 3y_2 + 2y_3 + 3y_4 + 3y_5 + 2y_6 + \cdots + 2y_{m-3} + 3y_{m-2} + 3y_{m-1} + y_m), \\ I_3 &= \frac{3h}{8} \sum_{i=0}^m c_i y_i, \end{aligned} \quad (5.8)$$

com $c_0 = c_m = 1$ e se i for múltiplo de 3, então $c_i = 2$ senão $c_i = 3$. Para a regra dos 3/8 de Simpson composta, o número de subintervalos m deve ser múltiplo de 3, que é o grau do polinômio interpolador no qual a regra se baseia. Deste modo, para qualquer regra de integração composta, o número de subintervalos m tem que ser múltiplo do grau n do polinômio interpolador utilizado pela regra. A integração da função $f(x) = e^x \sin(10x) + 8$, utilizando 2 polinômios interpoladores $P(x)$ de grau 3, pode ser vista na Figura 5.6.

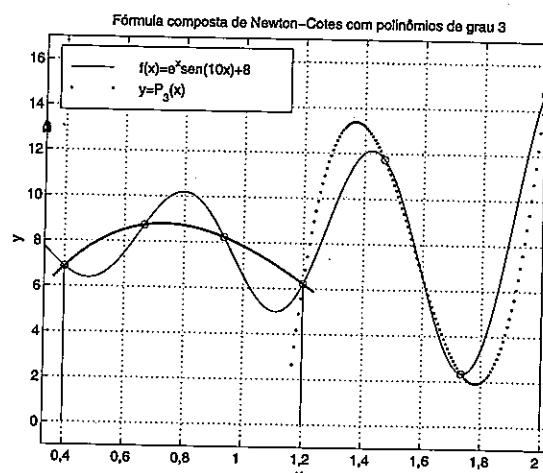


Figura 5.6 Integração numérica pela regra dos 3/8 de Simpson composta.

Exemplo 5.8 Calcular $\int_1^4 \log_e(x^3 + \sqrt{e^x + 1}) dx$ pela regra dos 3/8 de Simpson composta (5.8) com $m = 6$ (múltiplo de 3) subintervalos.

$$h = \frac{b-a}{m} = \frac{4-1}{6} \rightarrow h = 0,5.$$

No dispositivo prático, tem-se que $i = 0, 1, \dots, m$, $x_i = a, a+h, a+2h, \dots, b$, $y_i = f(x_i)$ e c_i são os coeficientes de Cotes para a regra dos 3/8 de Simpson composta,

| i | x_i | y_i | c_i |
|-----|-------|--------|-------|
| 0 | 1,0 | 1,0744 | 1 |
| 1 | 1,5 | 1,7433 | 3 |
| 2 | 2,0 | 2,3884 | 3 |
| 3 | 2,5 | 2,9578 | 2 |
| 4 | 3,0 | 3,4529 | 3 |
| 5 | 3,5 | 3,8860 | 3 |
| 6 | 4,0 | 4,2691 | 1 |

$$\begin{aligned} I_3 &= \frac{3 \times 0,5}{8} (1,0744 + 3(1,7433 + 2,3884 + 3,4529 + 3,8860) + 2 \times 2,9578 + 4,2691) \sim \\ I_3 &= 8,5633. \end{aligned}$$

Exemplo 5.9 Calcular $\int_0^{2,7} \frac{x + \sin(x)}{1 + \cos(x)} dx$, usando a regra dos 3/8 de Simpson composta (5.8) com passo de integração $h = 0,3$.

$$m = \frac{b-a}{h} = \frac{2,7-0}{0,3} \rightarrow m = 9 \text{ (múltiplo de 3)},$$

| i | x_i | y_i | c_i |
|-----|-------|---------|-------|
| 0 | 0,0 | 0,0000 | 1 |
| 1 | 0,3 | 0,3046 | 3 |
| 2 | 0,6 | 0,6380 | 3 |
| 3 | 0,9 | 1,0381 | 2 |
| 4 | 1,2 | 1,5650 | 3 |
| 5 | 1,5 | 2,3325 | 3 |
| 6 | 1,8 | 3,5894 | 2 |
| 7 | 2,1 | 5,9844 | 3 |
| 8 | 2,4 | 11,7113 | 3 |
| 9 | 2,7 | 32,6014 | 1 |

$$\begin{aligned} I_3 &= \frac{3 \times 0,3}{8} (0,0000 + 3(0,3046 + 0,6380 + 1,5650 + 2,3325 + 5,9844 + 11,7113) + \\ &\quad 2(1,0381 + 3,5894) + 32,6014) \sim I_3 = 12,3147. \end{aligned}$$

5.1.4 Erro de integração dos métodos de Newton-Cotes

Seja o erro de truncamento do polinômio de Gregory-Newton de grau n dado por (3.13)

$$T_n(x) = \prod_{i=0}^n (x - x_i) \frac{f^{n+1}(\theta)}{(n+1)!}, \quad x_0 < \theta < x_n.$$

Como a regra do trapézio é baseada em um polinômio de grau $n = 1$, tem-se que

$$T_1(x) = (x - x_0)(x - x_1) \frac{f''(\theta_1)}{2!}, \quad x_0 < \theta_1 < x_1.$$

O erro de integração $E_{1,1}$ cometido ao utilizar a regra do trapézio será a integral da fórmula acima no intervalo $[x_0, x_1]$

$$E_{1,1} = \int_{x_0}^{x_1} (x - x_0)(x - x_1) \frac{f''(\theta_1)}{2} dx.$$

Fazendo uma mudança de variável de x para $u = u_x = \frac{x - x_0}{h}$, tem-se

$$\begin{aligned} E_{1,1} &= \int_0^1 (hu)(h(u-1)) \frac{f''(\theta_1)}{2} h du = \frac{h^3 f''(\theta_1)}{2} \left(\frac{u^3}{3} - \frac{u^2}{2} \right) \Big|_0^1 \sim \\ E_{1,1} &= -\frac{h^3 f''(\theta_1)}{12}. \end{aligned}$$

O erro de integração global considerando os m subintervalos é

$$E_1 = \sum_{i=1}^m E_{1,i} = -\frac{h^3}{12} (f''(\theta_1) + f''(\theta_2) + \dots + f''(\theta_m)),$$

onde θ_i é determinado em cada um dos m subintervalos. Se $f''(x)$ for contínua no intervalo $[a, b]$, então existe algum valor de $x = \theta \in [a, b]$ para o qual o somatório acima é igual a $mf''(\theta)$. Deste modo, considerando que $h = (b - a)/m$, o erro global de integração da regra do trapézio torna-se

$$\begin{aligned} E_1 &= -\frac{h^3 m f''(\theta)}{12} = -\frac{(b-a)^3}{m^3} \frac{m f''(\theta)}{12}, \quad a < \theta < b, \\ E_1 &= -\frac{(b-a)^3}{12m^2} f''(\theta), \quad a < \theta < b. \end{aligned} \tag{5.9}$$

Similarmente, pode ser mostrado que para a regra do 1/3 de Simpson,

$$E_2 = -\frac{(b-a)^5}{180m^4} f'''(\theta), \quad a < \theta < b \tag{5.10}$$

e, para a regra dos 3/8 de Simpson,

$$E_3 = -\frac{(b-a)^5}{80m^4} f'''(\theta), \quad a < \theta < b. \tag{5.11}$$

Devido à dificuldade de determinar o valor de θ , ele é tomado como sendo o ponto no intervalo $[a, b]$, no qual a derivada de $f(x)$ apresenta o maior valor em módulo. Conseqüentemente, as três equações acima fornecem a cota máxima do erro de integração.

Apesar de a regra do 1/3 de Simpson aproximar a função $f(x)$ a ser integrada por um polinômio de grau 2, ela fornecerá resultado exato quando $f(x)$ for um polinômio de grau até 3, visto que, neste caso, $f'''(x) = 0 \rightarrow E_2 = 0$.

Exemplo 5.10 Calcular $\int_1^3 (4x^3 + 3x^2 + x + 1) dx$ utilizando a regra do 1/3 de Simpson (5.7) com $m = 2$ subintervalos.

$$h = \frac{b-a}{m} = \frac{3-1}{2} \rightarrow h = 1,$$

| i | x_i | y_i | c_i |
|-----|-------|-------|-------|
| 0 | 1 | 9 | 1 |
| 1 | 2 | 47 | 4 |
| 2 | 3 | 139 | 1 |

$$I_2 = \frac{1}{3}(9 + 4 \times 47 + 139) \sim I_2 = 112.$$

O erro de integração por (5.10) é zero, pois

$$f(x) = 4x^3 + 3x^2 + x + 1, \quad f'(x) = 12x^2 + 6x + 1, \quad f''(x) = 24x + 6, \quad f'''(x) = 24,$$

$$f'''(x) = 0 \rightarrow E_2 = -\frac{(b-a)^5}{180m^4} f'''(\theta) = -\frac{(3-1)^5}{180 \times 2^4} \times 0 \sim E_2 = 0.$$

De fato, o resultado é exato porque

$$\int_1^3 (4x^3 + 3x^2 + x + 1) dx = \left(x^4 + x^3 + \frac{x^2}{2} + x \right) \Big|_1^3 = 115,5 - 3,5 = 112.$$

O próximo exemplo ilustra que, comparando (5.9), (5.10) e (5.11), verifica-se que a regra do 1/3 de Simpson apresenta o menor erro de integração para um mesmo número de subintervalos m .

Exemplo 5.11 Calcular a integral $\int_0^\pi (e^x + \sin(x) + 2) dx$ usando as três primeiras fórmulas de Newton-Cotes com $m = 6$ subintervalos.

$$h = \frac{b-a}{m} = \frac{\pi-0}{6} \rightarrow h = \frac{\pi}{6},$$

| i | x_i | y_i | $c_i(t)$ | $c_i(1S)$ | $c_i(2S)$ |
|-----|----------|---------|----------|-----------|-----------|
| 0 | 0 | 3,0000 | 1 | 1 | 1 |
| 1 | $\pi/6$ | 4,1881 | 2 | 4 | 3 |
| 2 | $\pi/3$ | 5,7157 | 2 | 2 | 3 |
| 3 | $\pi/2$ | 7,8105 | 2 | 4 | 2 |
| 4 | $2\pi/3$ | 10,9866 | 2 | 2 | 3 |
| 5 | $5\pi/6$ | 16,2082 | 2 | 4 | 3 |
| 6 | π | 25,1407 | 1 | 1 | 1 |

A regra do trapézio fornece, por (5.6),

$$I_1 = \frac{\pi}{6 \times 2} (3,0000 + 2(4,1881 + 5,7157 + 7,8105 + 10,9866 + 16,2082) + 25,1407),$$

$$I_1 = 30,8816.$$

A regra do 1/3 de Simpson dará, por (5.7),

$$I_2 = \frac{\pi}{6 \times 3} (3,0000 + 4(4,1881 + 7,8105 + 16,2082) + 2(5,7157 + 10,9866) + 25,1407),$$

$$I_2 = 30,4337.$$

E a regra dos 3/8 de Simpson produzirá, por (5.8),

$$I_3 = \frac{3\pi}{6 \times 8} (3,0000 + 3(4,1881 + 5,7157 + 10,9866 + 16,2082) + 2 \times 7,8105 + 25,1407),$$

$$I_3 = 30,4455.$$

Para calcular o erro de integração de cada fórmula,

$$\begin{aligned} f(x) &= e^x + \operatorname{sen}(x) + 2, \quad f'(x) = e^x + \cos(x), \quad f''(x) = e^x - \operatorname{sen}(x) \sim \theta = \pi, \\ f'''(x) &= e^x - \cos(x) \text{ e } f^{iv}(x) = e^x + \operatorname{sen}(x) \sim \theta = \pi, \end{aligned}$$

sendo θ a abscissa do ponto onde a derivada apresenta o maior valor em módulo. Por (5.9), (5.10) e (5.11), tem-se

$$E_1 = -\frac{(b-a)^3}{12m^2} f''(\theta) = -\frac{(\pi-0)^3}{12 \times 6^2} (e^\pi - \operatorname{sen}(\pi)) \sim E_1 = -1,6609,$$

$$E_2 = -\frac{(b-a)^5}{180m^4} f^{iv}(\theta) = -\frac{(\pi-0)^5}{180 \times 6^4} (e^\pi + \operatorname{sen}(\pi)) \sim E_2 = -0,0304 \text{ e}$$

$$E_3 = -\frac{(b-a)^5}{80m^4} f^{iv}(\theta) = -\frac{(\pi-0)^5}{80 \times 6^4} (e^\pi + \operatorname{sen}(\pi)) \sim E_3 = -0,0683.$$

Considerando que $\int_0^\pi (e^x + \operatorname{sen}(x) + 2) dx = (e^x - \cos(x) + 2x) \Big|_0^\pi \approx 30,4239$, então o erro de integração máximo e real cometidos foram

| n | I_n | E_n | $30,4239 - I_n$ |
|-----|---------|---------|-----------------|
| 1 | 30,8816 | -1,6609 | -0,4577 |
| 2 | 30,4337 | -0,0304 | -0,0098 |
| 3 | 30,4455 | -0,0683 | -0,0216 |

A regra do 1/3 de Simpson produziu os menores erro máximo e erro real. O sinal negativo de E_n indica que a integração numérica foi por excesso, ou seja, $I_n > I_{\text{exata}}$. O exemplo a seguir mostra como calcular uma integral com um determinado erro máximo de integração.

Exemplo 5.12 Calcular $\int_0^\pi \left(\frac{x^4}{4} + x^2 + \operatorname{sen}(x) \right) dx$ com $E < 10^{-2}$ usando uma das três primeiras fórmulas de Newton-Cotes.

Inicialmente, determina-se o valor de m para cada fórmula usando (5.9), (5.10) e (5.11)

$$\begin{aligned} f(x) &= \frac{x^4}{4} + x^2 + \operatorname{sen}(x), \quad f'(x) = x^3 + 2x + \cos(x), \quad f''(x) = 3x^2 + 2 - \operatorname{sen}(x) \sim \\ \theta &= \pi, \end{aligned}$$

$$f'''(x) = 6x - \cos(x) \text{ e } f^{iv}(x) = 6 + \operatorname{sen}(x) \sim \theta = \frac{\pi}{2}.$$

Assim,

$$\left| \frac{(b-a)^3}{12m^2} f''(\theta) \right| < 10^{-2} \rightarrow m_1 > \left(\frac{(\pi-0)^3}{12 \times 10^{-2}} (3\pi^2 + 2 - \operatorname{sen}(\pi)) \right)^{\frac{1}{2}} \approx 90,37 \sim \\ m_1 = 91.$$

$$\left| \frac{(b-a)^5}{180m^4} f^{iv}(\theta) \right| < 10^{-2} \rightarrow m_2 > \left(\frac{(\pi-0)^5}{180 \times 10^{-2}} (6 + \operatorname{sen}(\pi/2)) \right)^{\frac{1}{4}} \approx 5,87 \sim \\ m_2 = 6.$$

$$\left| \frac{(b-a)^5}{80m^4} f^{iv}(\theta) \right| < 10^{-2} \rightarrow m_3 > \left(\frac{(\pi-0)^5}{80 \times 10^{-2}} (6 + \operatorname{sen}(\pi/2)) \right)^{\frac{1}{4}} \approx 7,19 \sim \\ m_3 = 9.$$

Os resultados de m_2 e m_3 foram arredondados para valores inteiros maiores e múltiplos de 2 e 3, respectivamente, para atender às restrições de cada uma das regras de integração. Como o menor m é m_2 , então a regra do 1/3 de Simpson deve ser usada, pois requer o menor esforço computacional para alcançar a exatidão requerida.

$$h = \frac{b-a}{m} = \frac{\pi-0}{6} \rightarrow h = \frac{\pi}{6},$$

| i | x_i | y_i | c_i |
|-----|----------|---------|-------|
| 0 | 0 | 0,0000 | 1 |
| 1 | $\pi/6$ | 0,7929 | 4 |
| 2 | $\pi/3$ | 2,2633 | 2 |
| 3 | $\pi/2$ | 4,9894 | 4 |
| 4 | $2\pi/3$ | 10,0628 | 2 |
| 5 | $5\pi/6$ | 19,0979 | 4 |
| 6 | π | 34,2219 | 1 |

Pela regra do 1/3 de Simpson (5.7)

$$I_2 = \frac{\pi}{6 \times 3} (0,0000 + 4(0,7929 + 4,9894 + 19,0979) + 2(2,2633 + 10,0628) + 34,2219),$$

$$I_2 = 27,6451.$$

Com o intuito de verificar se a exatidão requerida foi mesmo satisfeita,

$$\int_0^\pi \left(\frac{x^4}{4} + x^2 + \sin(x) \right) dx = \left(\frac{x^5}{20} + \frac{x^3}{3} - \cos(x) \right) \Big|_0^\pi \approx 27,6364,$$

$$|27,6364 - 27,6451| = 0,0087 < 10^{-2}.$$

5.1.5 Algoritmo e complexidade

A Figura 5.7 apresenta um algoritmo para calcular $\int_a^b f(x) dx$ por uma fórmula de Newton-Cotes gerada a partir de um polinômio de grau n , para $1 \leq n \leq 8$, utilizando m subintervalos iguais. Os parâmetros de entrada são o limite de integração inferior a e o superior b , o grau n do polinômio interpolador usado pela regra de integração e o número m de subintervalos. A função $f(x)$ deve ser especificada de acordo com a linguagem de programação escolhida. Os parâmetros de saída são o resultado da integração $Integral$ e a condição de erro $CondErro$, em que $CondErro = 0$ significa que não houve erro de consistência dos parâmetros de entrada, $CondErro = 1$, o grau do polinômio é menor que 1 ou maior que 8, e $CondErro = 2$ indica que o número de subintervalos m não é múltiplo do grau do polinômio n . Se ambas as condições ocorrerem, então $CondErro = 3$.

Os coeficientes de Cotes são aqueles compilados na Tabela 5.1, os quais são simétricos em relação à $\text{trunca}(n/2)$, isto é, $c_i = c_{n-i}$, $i = 0, 1, \dots, \text{trunca}(n/2)$, (trunca significa desprezar a parte fracionária). Deste modo, não é necessário armazenar todos os coeficientes c_i . Dado o grau n do polinômio, o apontador p indica a posição a partir da qual estão os coeficientes de Cotes no vetor c .

A Tabela 5.2 mostra a complexidade computacional da fórmula de Newton-Cotes implementada pelo algoritmo da Figura 5.7. Os polinômios são dados em termos do número de subintervalos m , pois independem do grau n do polinômio interpolador utilizado.

Algoritmo Newton-Cotes

```
{ Objetivo: Integrar uma função pelo método de Newton-Cotes }
parâmetros de entrada a, b, n, m
    { limite inferior, limite superior, grau do polinômio, número de subintervalos }
parâmetros de saída Integral, CondErro
    { valor da integral e condição de erro, sendo }
    { CondErro = 0 se não houve erro de consistência dos parâmetros dados, }
    { CondErro = 1 se ( $n < 1$  ou  $n > 8$ ), }
    { CondErro = 2 se resto( $m, n$ ) ≠ 0 e }
    { CondErro = 3 se ambas as condições ocorrerem }
d(1) ← 2; d(2) ← 6; d(3) ← 8; d(4) ← 90; d(5) ← 288; d(6) ← 840
d(7) ← 17280; d(8) ← 28350
c(1) ← 1; c(2) ← 1; c(3) ← 4; c(4) ← 1; c(5) ← 3; c(6) ← 7; c(7) ← 32
c(8) ← 12; c(9) ← 19; c(10) ← 75; c(11) ← 50; c(12) ← 41; c(13) ← 216
c(14) ← 27; c(15) ← 272; c(16) ← 751; c(17) ← 3577; c(18) ← 1323
c(19) ← 2989; c(20) ← 989; c(21) ← 5388; c(22) ← -928; c(23) ← 10496
c(24) ← -4540
CondErro ← 0; Integral ← 0
{ consistência dos parâmetros }
se  $n < 1$  ou  $n > 8$  então CondErro ← CondErro + 1, fimse
se resto( $m, n$ ) ≠ 0 então CondErro ← CondErro + 2, fimse
se CondErro ≠ 0 então abandone, fimse
{ cálculo da integral }
p ← trunca(0,25 * ((n * (n + 2)) + resto(n, 2))); h ← ((b - a))/m
para i ← 0 até m faça
    x ← a + i * h
    y ← f(x) { avaliar a função integrando em x }
    j ← p + trunca(0,5 * n - abs(resto(i, n) - 0,5 * n)))
    k ← 1 + trunca(((n - resto(i, n))/n)) - trunca(((m - resto(i, m))/m))
    Integral ← Integral + y * c(j) * k
    escreva i, x, y, c(j) * k
fimpara
Integral ← n * h/d(n) * Integral
finalgoritmo
```

Figura 5.7 Integração numérica pelo método de Newton-Cotes.

(Ver significado das funções abs, resto e trunca na Tabela 1.1, na página 6.)

Exemplo 5.13 Calcular $\int_0^\pi \sin(x) dx$ pelo algoritmo da Figura 5.7 com polinômios de grau $n = 2$ e $n = 3$, utilizando $m = 6$ subintervalos.

Para $n = 2$

Tabela 5.2 Complexidade da integração pelo método de Newton-Cotes.

| Operações | Complexidade |
|----------------|--------------|
| adições | $9m + 12$ |
| multiplicações | $5m + 9$ |
| divisões | $2m + 4$ |

% Os parametros de entrada

a = 0

b = 3.14159

n = 2

m = 6

% fornecem os resultados

Integracao por Newton-Cotes com polinomio de grau 2

| i | x(i) | y(i) | c(i) |
|---|---------|---------|------|
| 0 | 0.00000 | 0.00000 | 1 |
| 1 | 0.52360 | 0.50000 | 4 |
| 2 | 1.04720 | 0.86602 | 2 |
| 3 | 1.57080 | 1.00000 | 4 |
| 4 | 2.09439 | 0.86603 | 2 |
| 5 | 2.61799 | 0.50000 | 4 |
| 6 | 3.14159 | 0.00000 | 1 |

Integral = 2.00086

CondErro = 0

Para n = 3

% Os parametros de entrada

a = 0

b = 3.14159

n = 3

m = 6

% fornecem os resultados

Integracao por Newton-Cotes com polinomio de grau 3

| i | x(i) | y(i) | c(i) |
|---|---------|---------|------|
| 0 | 0.00000 | 0.00000 | 1 |
| 1 | 0.52360 | 0.50000 | 3 |
| 2 | 1.04720 | 0.86602 | 3 |
| 3 | 1.57080 | 1.00000 | 2 |
| 4 | 2.09439 | 0.86603 | 3 |
| 5 | 2.61799 | 0.50000 | 3 |
| 6 | 3.14159 | 0.00000 | 1 |

Integral = 2.00201

CondErro = 0

Não houve erro de consistência dos parâmetros de entrada e, como era de esperar, a regra do 1/3 de Simpson forneceu um resultado mais exato que a regra dos 3/8, para um mesmo número de subintervalos. ■

O próximo exemplo mostra que o uso de fórmulas baseadas em polinômios interpoladores de grau par é mais vantajoso.

Exemplo 5.14 Verificar o erro real cometido no cálculo de $\int_0^5 x \sin(3x)dx$, a ser apresentada na Seção 5.3, usando as sete primeiras fórmulas de Newton-Cotes, com $m = 420$

| n | I_exata - I_n |
|---|--------------------------|
| 1 | $1,2690 \times 10^{-4}$ |
| 2 | $9,4861 \times 10^{-9}$ |
| 3 | $2,1346 \times 10^{-8}$ |
| 4 | $3,9775 \times 10^{-12}$ |
| 5 | $8,5454 \times 10^{-12}$ |
| 6 | $6,6613 \times 10^{-16}$ |
| 7 | $2,6645 \times 10^{-15}$ |

De forma global, à medida que o grau n do polinômio interpolador aumenta, o erro diminui. No entanto, uma fórmula utilizando grau par é melhor do que a de grau ímpar seguinte. ■

5.2 Quadratura de Gauss-Legendre

O fato de escolher pontos igualmente espaçados nas fórmulas de Newton-Cotes certamente simplifica os cálculos. Contudo, se as abscissas não tiverem esta imposição de espaçamento constante, então podem ser obtidas fórmulas que forneçam uma maior exatidão, usando o mesmo número de pontos.

5.2.1 Fórmula para dois pontos

A Figura 5.8(a) ilustra a integração de uma função $f(x)$ pela regra do trapézio, a qual é baseada em um polinômio interpolador de grau 1 passando pelos pontos A e B . Pela Figura 5.8(b), os pontos C e D da curva podem ser escolhidos de tal maneira que a área do trapézio seja a mais próxima possível da área sob a curva.Com esse objetivo, é feita, inicialmente, uma mudança de variável de x para t , definida no intervalo $[-1, 1]$, por intermédio da expressão

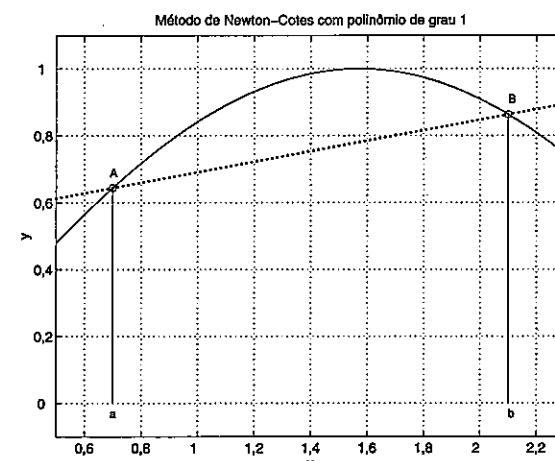
$$x = x(t) = \frac{b-a}{2}t + \frac{a+b}{2} \quad (5.12)$$

de modo que

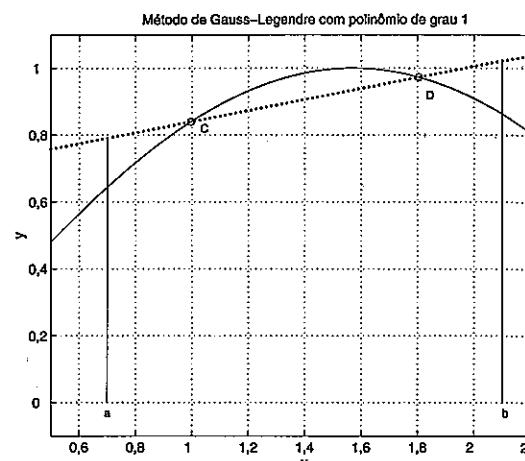
$$dx = \frac{b-a}{2}dt.$$

Definindo

$$F(t) = \frac{b-a}{2}f(x(t)), \quad (5.13)$$



(a) Newton-Cotes



(b) Gauss-Legendre

Figura 5.8 Aproximação da função por polinômio de grau 1.

então

$$\int_a^b f(x) dx = \int_{-1}^1 \frac{2}{b-a} F(t) \frac{b-a}{2} dt \sim \int_a^b f(x) dx = \int_{-1}^1 F(t) dt.$$

Considere agora a Figura 5.9. Como o ponto C tem coordenadas $C[t_1, F(t_1)]$ e o $D[t_2, F(t_2)]$, então deseja-se que,

$$\int_{-1}^1 F(t) dt \approx I_2,$$

sendo

$$I_2 = A_1 F(t_1) + A_2 F(t_2). \quad (5.14)$$

Em vista de (5.13) e com $x_i = x(t_i)$, tem-se

$$I_2 = \frac{b-a}{2} (A_1 f(x_1) + A_2 f(x_2)). \quad (5.15)$$

Deve-se notar que a expressão acima é análoga à regra do trapézio, dada por (5.2), na qual

$$\int_a^b f(x) dx \approx \frac{h}{2} f(a) + \frac{h}{2} f(b).$$

O problema consiste em encontrar valores de t_1 , t_2 , A_1 e A_2 que tornem a exatidão a maior possível. Para isso, o método é construído de modo a ser exato para polinômios de grau até 3, pois neste modo ter-se-á quatro incógnitas (t_1 , t_2 , A_1 e A_2) e quatro equações, uma para cada

$$F(t) = t^k, \quad k = 0, 1, 2, 3.$$

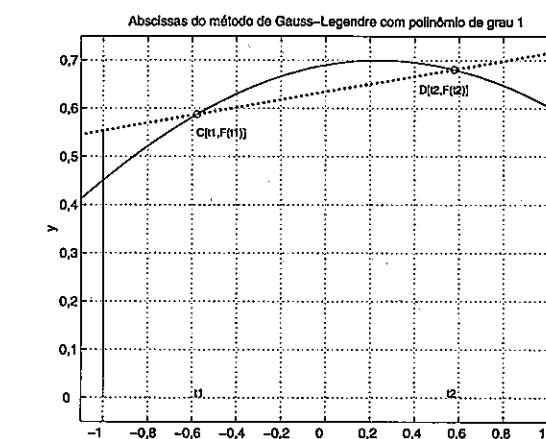


Figura 5.9 Escolha das abscissas no método de Gauss-Legendre.

Impondo (5.14) ser igual à integral analítica de $F(t)$, então

para $k = 0$

$$F(t) = 1 \rightarrow \int_{-1}^1 1 dt = 1 - (-1) = 2 = A_1 1 + A_2 1,$$

para $k = 1$

$$F(t) = t \rightarrow \int_{-1}^1 t dt = \frac{t^2}{2} \Big|_{-1}^1 = \frac{1}{2} - \left(-\frac{1}{2}\right) = 0 = A_1 t_1 + A_2 t_2,$$

para $k = 2$

$$F(t) = t^2 \rightarrow \int_{-1}^1 t^2 dt = \frac{t^3}{3} \Big|_{-1}^1 = \frac{1}{3} - \left(-\frac{1}{3}\right) = \frac{2}{3} = A_1 t_1^2 + A_2 t_2^2 \text{ e}$$

para $k = 3$

$$F(t) = t^3 \rightarrow \int_{-1}^1 t^3 dt = \frac{t^4}{4} \Big|_{-1}^1 = \frac{1}{4} - \left(-\frac{1}{4}\right) = 0 = A_1 t_1^3 + A_2 t_2^3.$$

As expressões acima constituem um sistema de equações não lineares de ordem 4

$$A_1 + A_2 = 2,$$

$$A_1 t_1 + A_2 t_2 = 0,$$

$$A_1 t_1^2 + A_2 t_2^2 = \frac{2}{3} \text{ e}$$

$$A_1 t_1^3 + A_2 t_2^3 = 0,$$

cuja solução é

$$t_1 = -\frac{1}{\sqrt{3}} \approx -0,5774, \quad t_2 = \frac{1}{\sqrt{3}} \approx 0,5774, \quad A_1 = 1 \text{ e } A_2 = 1.$$

Exemplo 5.15 Calcular $\int_1^5 (2x^3 + 3x^2 + 6x + 1)dx$, usando (5.15).

Por (5.12),

$$x_i = \frac{b-a}{2}t_i + \frac{a+b}{2} = \frac{5-1}{2}t_i + \frac{1+5}{2} \sim x_i = 2t_i + 3.$$

Os cálculos podem ser sistematizados em um dispositivo prático composto por cinco colunas, de modo que $i = 1, 2$; $t_1 = -\frac{1}{\sqrt{3}}$ e $t_2 = \frac{1}{\sqrt{3}}$ (para $n = 2$), x_i , $f(x_i)$ e $A_1 = A_2 = 1$ (para $n = 2$),

| i | t_i | x_i | $f(x_i)$ | A_i |
|-----|-----------------------|--------|----------|-------|
| 1 | $-\frac{1}{\sqrt{3}}$ | 1,8453 | 34,8542 | 1 |
| 2 | $\frac{1}{\sqrt{3}}$ | 4,1547 | 221,1458 | 1 |

$$I_2 = \frac{b-a}{2}(A_1f(x_1) + A_2f(x_2)) = \frac{5-1}{2}(1 \times 34,8542 + 1 \times 221,1458) \sim$$

$$I_2 = 512,0000.$$

Este resultado, de fato, é exato porque

$$\int_1^5 (2x^3 + 3x^2 + 6x + 1)dx = \left(\frac{x^4}{2} + x^3 + 3x^2 + x \right) \Big|_1^5 = 517,5 - 5,5 = 512.$$

Exemplo 5.16 Calcular $\int_0^\pi (e^x + \operatorname{sen}(x) + 2)dx$, usando (5.15).

Por (5.12),

$$x_i = \frac{b-a}{2}t_i + \frac{a+b}{2} = \frac{\pi-0}{2}t_i + \frac{0+\pi}{2} \sim x_i = \frac{\pi}{2}(t_i + 1).$$

| i | t_i | x_i | $f(x_i)$ | A_i |
|-----|-----------------------|--------|----------|-------|
| 1 | $-\frac{1}{\sqrt{3}}$ | 0,6639 | 4,5585 | 1 |
| 2 | $\frac{1}{\sqrt{3}}$ | 2,4777 | 14,5300 | 1 |

Assim,

$$I_2 = \frac{b-a}{2}(A_1f(x_1) + A_2f(x_2)) = \frac{\pi-0}{2}(1 \times 4,5585 + 1 \times 14,5300) \sim$$

$$I_2 = 29,9841.$$

No Exemplo 5.11, foi visto que o valor exato dessa integral é aproximadamente 30,4239, portanto o erro cometido pela quadratura de Gauss-Legendre com 2 pontos foi de $|30,4239 - 29,9841| = 0,4398$, que é mais exato que aquele obtido pela regra do trapézio com $m = 6$ subintervalos, equivalente a 7 pontos ($|30,4239 - 30,8816| = 0,4577$).

Exemplo 5.17 Calcular a soma das áreas S_1 , S_2 e S_3 entre o polinômio de Legendre de grau 1 e a função $f(x) = x^3 - 6x^2 + 11x - 5$ obtidas no intervalo $[1, 4]$ e mostradas na Figura 5.10.

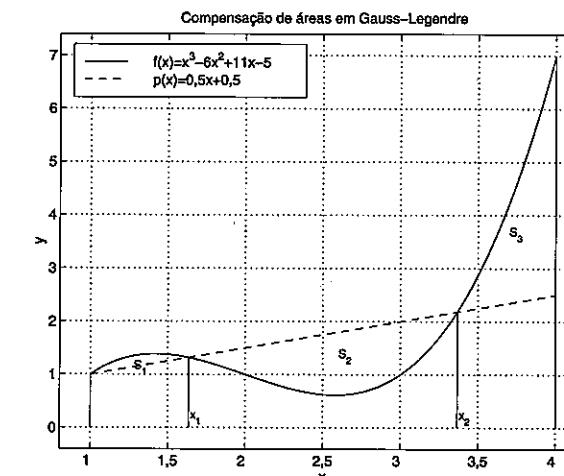


Figura 5.10 Áreas entre a função e o polinômio no método de Gauss-Legendre.

Inicialmente, calculam-se as abscissas x_1 e x_2 a partir de $t_1 = -\frac{1}{\sqrt{3}}$ e $t_2 = \frac{1}{\sqrt{3}}$. Por (5.12),

$$x_1 = \frac{4-1}{2}t_1 + \frac{1+4}{2} = \frac{5-\sqrt{3}}{2} \approx 1,63397,$$

$$x_2 = \frac{4-1}{2}t_2 + \frac{1+4}{2} = \frac{5+\sqrt{3}}{2} \approx 3,36603.$$

A forma analítica do polinômio de Legendre de grau 1 que passa pelos pontos de coordenadas $(x_1, f(x_1))$ e $(x_2, f(x_2))$ é

$$p(x) = f(x_1) + \frac{f(x_2) - f(x_1)}{x_2 - x_1}(x - x_1),$$

$$p(x) = 0,5x + 0,5.$$

Sendo $g(x) = f(x) - p(x) = x^3 - 6x^2 + 10,5x - 5,5$, tem-se que

$$S_1 = \int_1^{x_1} g(x)dx \approx 0,08702,$$

$$S_2 = \int_{x_1}^{x_2} g(x)dx \approx -1,29904,$$

$$S_3 = \int_{x_2}^4 g(x)dx \approx 1,21202.$$

Portanto, a soma das três áreas é igual a 0, mostrando que para $n = 2$ pontos, existe uma compensação exata das áreas entre o polinômio de Legendre de grau 1 e a função polinomial de grau 3.

5.2.2 Fórmula geral

Agora, o problema é determinar os valores dos pesos A_i , e das abscissas t_i , $i = 1, 2, \dots, n$, as quais se encontram no intervalo $[-1, 1]$, para utilizá-los na fórmula

$$\int_a^b f(x)dx = \int_{-1}^1 F(t)dt \approx I_n,$$

sendo

$$I_n = A_1F(t_1) + A_2F(t_2) + \dots + A_nF(t_n), \quad (5.16)$$

de modo que a mesma seja exata para polinômios de grau menor ou igual a $2n - 1$. Fazendo

$$F(t) = t^k, \quad k = 0, 1, \dots, 2n - 1,$$

e sabendo que

$$\int_{-1}^1 t^k dt = \begin{cases} 0, & \text{se } k \text{ for ímpar,} \\ \frac{2}{k+1}, & \text{se } k \text{ for par,} \end{cases}$$

então impondo que (5.16) seja exata para a integração de $F(t)$ acima, é obtido o seguinte sistema de equações não lineares de ordem $2n$

$$A_1 + A_2 + A_3 + \dots + A_n = 2,$$

$$A_1t_1 + A_2t_2 + A_3t_3 + \dots + A_nt_n = 0,$$

$$A_1t_1^2 + A_2t_2^2 + A_3t_3^2 + \dots + A_nt_n^2 = \frac{2}{3},$$

...

$$A_1t_1^{2n-1} + A_2t_2^{2n-1} + A_3t_3^{2n-1} + \dots + A_nt_n^{2n-1} = 0,$$

cuja solução fornece os n pesos A_i e as n abscissas t_i desejados. Em vista de (5.13) e com $x_i = x(t_i)$, (5.16) é equivalente a

$$I_n = \frac{b-a}{2} \sum_{i=1}^n A_i f(x_i). \quad (5.17)$$

A necessidade de resolver este sistema não linear pode ser evitada usando um processo alternativo. Inicialmente, sejam os polinômios de Legendre [8] definidos pela fórmula de recorrência

$$L_n(x) = \frac{(2n-1)xL_{n-1}(x) - (n-1)L_{n-2}(x)}{n}, \quad (5.18)$$

com $L_0(x) = 1$ e $L_1(x) = x$. Por exemplo,

$$L_2(x) = \frac{3x^2 - 1}{2},$$

$$L_3(x) = \frac{5x^3 - 3x}{2},$$

$$L_4(x) = \frac{35x^4 - 30x^2 + 3}{8} \quad \text{e}$$

$$L_5(x) = \frac{63x^5 - 70x^3 + 15x}{8}.$$

Os polinômios de Legendre possuem as seguintes propriedades básicas

$$L_n(1) = 1 \quad \text{e} \quad L_n(-1) = (-1)^n, \quad n = 0, 1, 2, \dots \quad \text{e}$$

$$\int_{-1}^1 L_n(x)Q_k(x) dx = 0, \quad n > k, \quad (5.19)$$

sendo $Q_k(x)$ um polinômio qualquer de grau $k < n$. A integral de (5.19) é chamada de produto escalar das funções $L_n(x)$ e $Q_k(x)$. Duas funções são ditas ortogonais se seu produto escalar for nulo, portanto, $L_n(x)$ e $Q_k(x)$ são ortogonais. Além disso,

$$\int_{-1}^1 L_n(x)L_k(x) dx \begin{cases} = 0, & \text{se } n \neq k, \\ > 0, & \text{se } n = k. \end{cases}$$

Uma outra propriedade é que as equações algébricas $L_n(x) = 0$ possuem n raízes reais distintas pertencentes ao intervalo $(-1, 1)$ e simétricas em relação à origem, conforme pode ser observado na Figura 5.11.

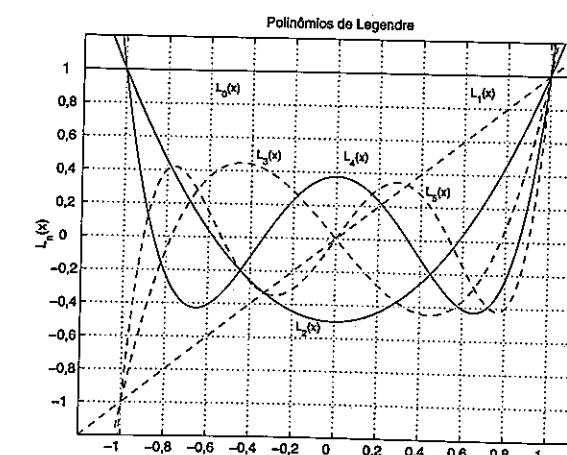


Figura 5.11 Polinômios de Legendre de grau até 5.

Considere agora os polinômios

$$F_k(t) = t^k L_n(t), \quad k = 0, 1, 2, \dots, n-1,$$

onde $L_n(t)$ é um polinômio de Legendre de grau n . Desde que $F_k(t)$ é de grau menor ou igual a $2n-1$, então (5.16) é exata

$$\int_{-1}^1 F_k(t) dt = \sum_{i=1}^n A_i F_k(t_i), \quad k = 0, 1, 2, \dots, n-1,$$

$$\int_{-1}^1 t^k L_n(t) dt = \sum_{i=1}^n A_i t_i^k L_n(t_i), \quad k = 0, 1, 2, \dots, n-1.$$

Devido à ortogonalidade dos polinômios de Legendre com qualquer polinômio de grau menor que o de Legendre, mostrado em (5.19), tem-se que

$$\int_{-1}^1 t^k L_n(t) dt = 0, \quad n > k.$$

Portanto,

$$\sum_{i=1}^n A_i t_i^k L_n(t_i) = 0, \quad k = 0, 1, 2, \dots, n-1.$$

Essa expressão será verdadeira para qualquer valor de A_i se $L_n(t_i) = 0$ para todo i . Assim, para obter uma maior exatidão na fórmula de quadratura (5.16) é suficiente que t_i , $i = 1, 2, \dots, n$ sejam os zeros do polinômio de Legendre de grau n .

Sendo conhecidas as abscissas t_i , então o sistema não linear se reduz a um sistema linear de ordem n , cuja solução fornece os pesos A_i , $i = 1, 2, \dots, n$

$$\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ t_1 & t_2 & t_3 & \cdots & t_n \\ t_1^2 & t_2^2 & t_3^2 & \cdots & t_n^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ t_1^{n-1} & t_2^{n-1} & t_3^{n-1} & \cdots & t_n^{n-1} \end{bmatrix} \begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ \vdots \\ A_n \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ \frac{2}{3} \\ \vdots \\ \end{bmatrix}.$$

No entanto, em vez de resolver este sistema via decomposição LU , os pesos A_i podem ser obtidos mais facilmente pela expressão abaixo, conforme Rybicki [34],

$$A_i = \frac{2}{(1 - t_i^2)(L'_n(t_i))^2}, \quad i = 1, 2, \dots, n, \quad (5.20)$$

onde $L'_n(t_i)$ é a derivada de $L_n(x)$ na abscissa t_i . Os valores de t_i e A_i para $n = 1, 2, \dots, 10$, extraídos de Abramowitz e Stegun [1, página 916], estão compilados na Tabela 5.3.

Tabela 5.3 Abscissas e pesos para quadratura de Gauss-Legendre.

| n | i | t_i | A_i |
|-----|-------|-----------------------------|---------------------|
| 1 | 1 | 0 | 2 |
| 2 | 2; 1 | $\pm 0,57735\ 02691\ 89626$ | 1 |
| 3 | 2 | 0 | 0,88888 88888 88889 |
| | 3; 1 | $\pm 0,77459\ 66692\ 41483$ | 0,55555 55555 55556 |
| 4 | 3; 2 | $\pm 0,33998\ 10435\ 84856$ | 0,65214 51548 62546 |
| | 4; 1 | $\pm 0,86113\ 63115\ 94053$ | 0,34785 48451 37454 |
| 5 | 3 | 0 | 0,56888 88888 88889 |
| | 4; 2 | $\pm 0,53846\ 93101\ 05683$ | 0,47862 86704 99366 |
| | 5; 1 | $\pm 0,90617\ 98459\ 38664$ | 0,23692 68850 56189 |
| 6 | 4; 3 | $\pm 0,23861\ 91860\ 83197$ | 0,46791 39345 72691 |
| | 5; 2 | $\pm 0,66120\ 93864\ 66265$ | 0,36076 15730 48139 |
| | 6; 1 | $\pm 0,93246\ 95142\ 03152$ | 0,17132 44923 79170 |
| 7 | 4 | 0 | 0,41795 91836 73469 |
| | 5; 3 | $\pm 0,40584\ 51513\ 77397$ | 0,38183 00505 05119 |
| | 6; 2 | $\pm 0,74153\ 11855\ 99394$ | 0,27970 53914 89277 |
| | 7; 1 | $\pm 0,94910\ 79123\ 42759$ | 0,12948 49661 68870 |
| 8 | 5; 4 | $\pm 0,18343\ 46424\ 95650$ | 0,36268 37833 78362 |
| | 6; 3 | $\pm 0,52553\ 24099\ 16329$ | 0,31370 66458 77887 |
| | 7; 2 | $\pm 0,79666\ 64774\ 13627$ | 0,22238 10344 53374 |
| | 8; 1 | $\pm 0,96028\ 98564\ 97536$ | 0,10122 85362 90376 |
| 9 | 5 | 0 | 0,33023 93550 01260 |
| | 6; 4 | $\pm 0,32425\ 34234\ 03809$ | 0,31234 70770 40003 |
| | 7; 3 | $\pm 0,61337\ 14327\ 00590$ | 0,26061 06964 02935 |
| | 8; 2 | $\pm 0,83603\ 11073\ 26636$ | 0,18064 81606 94857 |
| | 9; 1 | $\pm 0,96816\ 02395\ 07626$ | 0,08127 43883 61574 |
| 10 | 6; 5 | $\pm 0,14887\ 43389\ 81631$ | 0,29552 42247 14753 |
| | 7; 4 | $\pm 0,43339\ 53941\ 29247$ | 0,26926 67193 09996 |
| | 8; 3 | $\pm 0,67940\ 95682\ 99024$ | 0,21908 63625 15982 |
| | 9; 2 | $\pm 0,86506\ 33666\ 88985$ | 0,14945 13491 50581 |
| | 10; 1 | $\pm 0,97390\ 65285\ 17172$ | 0,06667 13443 08688 |

Exemplo 5.18 Verificar a ortogonalidade dos polinômios $L_2(x)$ e $L_3(x)$ de Legendre.

Os polinômios serão ortogonais se

$$\int_{-1}^1 L_2(x)L_3(x)dx = 0.$$

Assim,

$$\int_{-1}^1 L_2(x)L_3(x)dx = \int_{-1}^1 \left(\frac{3x^2 - 1}{2}\right) \left(\frac{5x^3 - 3x}{2}\right) dx,$$

$$\int_{-1}^1 L_2(x)L_3(x)dx = \int_{-1}^1 \frac{1}{4} (15x^5 - 14x^3 + 3x) dx,$$

$$\int_{-1}^1 L_2(x)L_3(x)dx = \frac{1}{4} \left(\frac{15}{6}x^6 - \frac{7}{2}x^4 + \frac{3}{2}x^2 \right) \Big|_{-1}^1,$$

$$\int_{-1}^1 L_2(x)L_3(x)dx = 0.$$

Exemplo 5.19 Verificar que $\pi = 4 \int_0^1 \frac{1}{1+x^2} dx$ por intermédio de (5.17) com $n = 3$ e $n = 4$.

Por (5.12),

$$x_i = \frac{b-a}{2}t_i + \frac{a+b}{2} = \frac{1-0}{2}t_i + \frac{0+1}{2} \sim x_i = \frac{1}{2}(t_i + 1).$$

No dispositivo prático, $i = 1, 2, \dots, n$; t_i são os n zeros do polinômio de Legendre de grau n e A_i são os n pesos obtidos da Tabela 5.3 ou pela expressão (5.20).

Para $n = 3$

| i | t_i | x_i | $f(x_i)$ | A_i |
|-----|----------|---------|----------|---------|
| 1 | -0,77460 | 0,11270 | 0,98746 | 0,55556 |
| 2 | 0 | 0,50000 | 0,80000 | 0,88889 |
| 3 | 0,77460 | 0,88730 | 0,55950 | 0,55556 |

Usando (5.17) com $n = 3$, tem-se que

$$I_3 = \frac{b-a}{2} \sum_{i=1}^3 A_i f(x_i) \rightarrow I_3 = 0,78527 \sim 4 \times I_3 = 3,14108 \approx \pi.$$

Para $n = 4$

| i | t_i | x_i | $f(x_i)$ | A_i |
|-----|----------|---------|----------|---------|
| 1 | -0,86114 | 0,06943 | 0,99520 | 0,34785 |
| 2 | -0,33998 | 0,33001 | 0,90179 | 0,65215 |
| 3 | 0,33998 | 0,66999 | 0,69019 | 0,65215 |
| 4 | 0,86114 | 0,93057 | 0,53592 | 0,34785 |

Utilizando (5.17) com $n = 4$, obtém-se

$$I_4 = \frac{b-a}{2} \sum_{i=1}^4 A_i f(x_i) \rightarrow I_4 = 0,78540 \sim 4 \times I_4 = 3,14160 \approx \pi.$$

Exemplo 5.20 Calcular $\int_0^\pi (e^x + \sin(x) + 2)dx$ do Exemplo 5.11, pela quadratura de Gauss-Legendre com $n = 5$.

Por (5.12),

$$x_i = \frac{b-a}{2}t_i + \frac{a+b}{2} = \frac{\pi-0}{2}t_i + \frac{0+\pi}{2} \sim x_i = \frac{\pi}{2}(t_i + 1).$$

| i | t_i | x_i | $f(x_i)$ | A_i |
|-----|----------|---------|----------|---------|
| 1 | -0,90618 | 0,14737 | 3,30562 | 0,23693 |
| 2 | -0,53847 | 0,72497 | 4,72778 | 0,47863 |
| 3 | 0 | 1,57080 | 7,81050 | 0,56889 |
| 4 | 0,53847 | 2,41662 | 13,87103 | 0,47863 |
| 5 | 0,90618 | 2,99422 | 22,11662 | 0,23693 |

Usando (5.17) com $n = 5$, tem-se

$$I_5 = \frac{b-a}{2} \sum_{i=1}^5 A_i f(x_i) \rightarrow I_5 = 30,42406.$$

Comparando com o resultado exato que é aproximadamente 30,42388, a quadratura de Gauss-Legendre com $n = 5$ produz um resultado mais exato do que a primeira regra de Simpson com $m = 6$ (30,4337).

5.2.3 Erro de integração da fórmula de Gauss-Legendre

O erro de integração da fórmula de Gauss-Legendre é, conforme Demidovich e Maron [10], dado por

$$E_n = \frac{(b-a)^{2n+1}(n!)^4}{((2n)!)^3(2n+1)} f^{2n}(\theta), \quad a < \theta < b, \quad (5.21)$$

onde θ é a abscissa, na qual a derivada $f^{2n}(x)$ apresenta o maior valor em módulo no intervalo $[a, b]$. Consequentemente, (5.21) fornece a cota máxima do erro de integração da fórmula de Gauss-Legendre.

Exemplo 5.21 Calcular $\int_0^\pi \left(\frac{x^4}{4} + x^2 + \sin(x)\right) dx$ usando o método de Gauss-Legendre com $n = 2$ e o respectivo erro de integração.

Por (5.12),

$$x_i = \frac{b-a}{2}t_i + \frac{a+b}{2} = \frac{\pi-0}{2}t_i + \frac{0+\pi}{2} \sim x_i = \frac{\pi}{2}(t_i + 1).$$

Para $n = 2$

| i | t_i | x_i | $f(x_i)$ | A_i |
|-----|----------|---------|----------|-------|
| 1 | -0,57735 | 0,66390 | 1,10552 | 1 |
| 2 | 0,57735 | 2,47770 | 16,17701 | 1 |

Usando (5.17) com $n = 2$, tem-se que

$$I_2 = \frac{b-a}{2} \sum_{i=1}^2 A_i f(x_i) \rightarrow I_2 = 27,14733.$$

Para o cálculo do erro máximo,

$$f(x) = \frac{x^4}{4} + x^2 + \operatorname{sen}(x), \quad f'(x) = x^3 + 2x + \cos(x),$$

$$f''(x) = 3x^2 + 2 - \operatorname{sen}(x), \quad f'''(x) = 6x - \cos(x) \text{ e}$$

$$f^{iv}(x) = 6 + \operatorname{sen}(x) \rightarrow \theta = \frac{\pi}{2}.$$

Por (5.21),

$$E_n = \frac{(b-a)^{2n+1}(n!)^4}{((2n)!)^3(2n+1)} f^{2n}(\theta) \rightarrow E_2 = \frac{(\pi-0)^5(2!)^4}{(4!)^3(5)} \left(6 + \operatorname{sen}\left(\frac{\pi}{2}\right)\right) \sim$$

$$E_2 = 0,49587.$$

Considerando que o valor exato da integral é aproximadamente 27,63641, então o erro real é $|27,63641 - 27,14733| = 0,48908 < E_2$.

5.2.4 Algoritmos e complexidade

Inicialmente, as abscissas t_i , isto é, os zeros do polinômio de Legendre (5.18), são calculadas pelo método de Newton (ver Seção 6.5.1), e os pesos A_i pela expressão (5.20), segundo um esquema proposto por Rybicki [34, página 145]. Esse esquema foi implementado no algoritmo **PesAbsGL** da Figura 5.12.

O parâmetro de entrada é o número de pontos n da fórmula desejada. Os parâmetros de saída são o vetor A com os pesos, o vetor T com as abscissas e CondErro , a condição de erro no parâmetro de entrada. Se $\text{CondErro} = 0$ não houve erro de consistência ($n \geq 1$), e $\text{CondErro} = 1$ significa que $n < 1$. Como os zeros dos polinômios de Legendre são simétricos em relação à origem, somente as abscissas não negativas são calculadas. Este algoritmo pode ser utilizado também para gerar os valores de t_i e A_i como aqueles descritos na Tabela 5.3.

Exemplo 5.22 Calcular os pesos e as abscissas para a fórmula de Gauss-Legendre com $n = 5$, utilizando o algoritmo da Figura 5.12.

```

Algoritmo PesAbsGL
{ Objetivo: Calcular pesos e abscissas para a fórmula de Gauss-Legendre }
parâmetros de entrada n { número de pontos }
parâmetros de saída A, T, CondErro
{ Pesos, abscissas e condição de erro, sendo }
{ CondErro = 0 se não houve erro ( $n \geq 1$ ) e CondErro = 1 se  $n < 1$  }
se  $n < 1$  então CondErro ← 1, abandone, fimse
CondErro ← 0; p1 ← 3,14159265358979323846; m ← trunc(0,5 * (n + 1))
para i ← 1 até m faça
    z ← cos(p1 * ((i - 0,25) / (n + 0,5)))
    repita
        p1 ← 1; p2 ← 0
        para j ← 1 até n faça
            p3 ← p2; p2 ← p1
            { polinômio de Legendre no ponto z }
            p1 ← (((2 * j - 1) * z * p2 - (j - 1) * p3) / j)
        fimpara
        { derivada do polinômio de Legendre no ponto z }
        pp ← n * (z * p1 - p2) / (z^2 - 1); z1 ← z
        { método de Newton para calcular os zeros do polinômio }
        z ← z1 - p1 / pp
        se abs(z - z1) < 10^-15 então interrompa, fimse
    fimrepita
    T(m + 1 - i) ← z { abscissa }
    A(m + 1 - i) ← 2 / (((1 - z^2) * pp^2)) { peso }
    { somente as raízes não negativas são calculadas devido à simetria }
fimpara
finalgoritmo

```

Figura 5.12 Cálculo das abscissas e pesos para as fórmulas de Gauss-Legendre.

(Ver significado das funções abs, cos e trunc na Tabela 1.1, na página 6.)

```

% 0 parâmetro de entrada
n = 5
% produz os resultados
A = 0.56888888888889 0.47862867049937 0.23692688505619
T =
CondErro = 0

```

A Figura 5.13 apresenta o algoritmo **Gauss-Legendre** para integrar uma função $f(x)$ de a até b pelo método de Gauss-Legendre com qualquer número de pontos $n \geq 1$. Os parâmetros de entrada são os limites de integração inferior a e superior b e o número de pontos n da fórmula desejada. A função $f(x)$ deve ser especificada de acordo com a linguagem de programação adotada. Os parâmetros de saída são o valor da integração *Integral* e *CondErro*,

a condição de erro do número de pontos, sendo que $CondErro = 0$ significa que não houve erro de consistência ($n \geq 1$) e $CondErro = 1$ indica que $n < 1$. Os pesos e as abscissas são calculados pelo algoritmo **PesAbsGL** da Figura 5.12. No entanto, o algoritmo **Gauss-Legendre** pode ser adaptado de forma que os pesos e as abscissas possam ser fornecidos como parâmetros de entrada.

```

Algoritmo Gauss-Legendre
{ Objetivo: Integrar uma função pelo método de Gauss-Legendre }
parâmetros de entrada a, b, n
    { limite inferior, limite superior, número de pontos }
parâmetros de saída Integral, CondErro
    { valor da integral e condição de erro, sendo }
    { CondErro = 0 se não houve erro ( $n \geq 1$ ) e CondErro = 1 se  $n < 1$  }
Integral ← 0
{ cálculo dos pesos e abscissas }
[Avet, Tvet, CondErro] ← PesAbsGL(n)  (ver Figura 5.12)
se CondErro ≠ 0 então abandone, fimse
{ cálculo da integral }
e1 ← (b - a)/2
e2 ← (a + b)/2
se resto(n, 2) = 0 então
    c1 ← 1; c2 ← 0,5
senão
    c1 ← 0; c2 ← 1
fimse
para i ← 1 até n faça
    k ← trunc((i - 0,5 * (n + 1)) + sinal(i - 0,5 * ((n + c1)) * c2))
    t ← sinal(k) * Tvet(abs(k))
    x ← e1 * t + e2
    y ← f(x)  { avaliar a função integrando em x }
    c ← Avet(abs(k))
    Integral ← Integral + y * c
    escreva i, t, x, y, c
fimpara
Integral ← e1 * Integral
finalgoritmo

```

Figura 5.13 Integração numérica pelo método de Gauss-Legendre.

(Ver significado das funções abs, resto, trunc e sinal na Tabela 1.1, na página 6.)

Durante o cálculo da integral, a variável k , avaliada de modo não trivial, fornece a posição dos pesos e das abscissas nos vetores **Avet** e **Tvet**, respectivamente, além de controlar o sinal das abscissas, visto que somente aquelas positivas e nulas são calculadas e armazenadas.

A Tabela 5.4 apresenta a complexidade computacional da quadratura de Gauss-Legendre quando implementada pelo algoritmo da Figura 5.13.

Tabela 5.4 Complexidade da integração pela quadratura de Gauss-Legendre.

| Operações | Complexidade |
|----------------|--------------|
| adições | $7n + 2$ |
| multiplicações | $6n + 1$ |
| divisões | 2 |

Exemplo 5.23 Calcular $\int_0^{\pi} \sin(x) dx$ pelo algoritmo da Figura 5.13 com $n = 5$ e $n = 6$.

Para $n = 5$

```

% Os parametros de entrada
a = 0
b = 3.14159
n = 5
% produzem os resultados
    Integracao numerica pelo metodo de Gauss-Legendre
    i      t(i)          x(i)        f(x(i))      A(i)
    1     -0.90618       0.14737     0.14684     0.23693
    2     -0.53847       0.72497     0.66311     0.47863
    3      0.00000       1.57080     1.00000     0.56889
    4      0.53847       2.41662     0.66312     0.47863
    5      0.90618       2.99422     0.14684     0.23693
Integral = 2.0000001103
CondErro = 0

```

Para $n = 6$

```

% Os parametros de entrada
a = 0
b = 3.14159
n = 6
% produzem os resultados
    Integracao numerica pelo metodo de Gauss-Legendre
    i      t(i)          x(i)        f(x(i))      A(i)
    1     -0.93247       0.10608     0.10588     0.17132
    2     -0.66121       0.53217     0.50740     0.36076
    3     -0.23862       1.19597     0.93057     0.46791
    4      0.23862       1.94562     0.93057     0.46791
    5      0.66121       2.60942     0.50741     0.36076
    6      0.93247       3.03551     0.10588     0.17132
Integral = 1.9999999995
CondErro = 0

```

Exemplo 5.24 Verificar que $\pi = \int_0^1 \frac{4}{1+x^2} dx$ com $n = 10$ utilizando o algoritmo descrito na Figura 5.13.

```
% Os parametros de entrada
a = 0
b = 1
n = 10
% produzem os resultados
Integracao numerica pelo metodo de Gauss-Legendre
i      t(i)      x(i)      f(x(i))      A(i)
1     -0.97391   0.01305   3.99932   0.06667
2     -0.86506   0.06747   3.98187   0.14945
3     -0.67941   0.16030   3.89980   0.21909
4     -0.43340   0.28330   3.70281   0.26927
5     -0.14887   0.42556   3.38666   0.29552
6     0.14887   0.57444   3.00757   0.29552
7     0.43340   0.71670   2.64261   0.26927
8     0.67941   0.83970   2.34590   0.21909
9     0.86506   0.93253   2.13948   0.14945
10    0.97391   0.98695   2.02626   0.06667
```

Integral = 3.1415926536
CondErro = 0

5.3 Comparação dos métodos de integração simples

Pelos exemplos apresentados neste capítulo, parece óbvio que a quadratura de Gauss-Legendre é bem superior aos métodos de Newton-Cotes. A seguir, serão feitas comparações entre essas duas classes de métodos por intermédio de três exemplos.

Primeiramente, seja a integral da função apresentada na Figura 5.14(a),

$$\int_0^\pi \sin(x) dx,$$

cuja solução exata é

$$I = -\cos(x) \Big|_0^\pi = 2.$$

A Tabela 5.5 mostra a diferença entre o valor exato e os resultados obtidos pelas diversas fórmulas de integração.

Foram utilizadas regras simples de Newton-Cotes, isto é, o grau do polinômio é igual ao número de subintervalos. Além disso, o número de pontos de Gauss-Legendre é igual a $m + 1$, sendo m o número de subintervalos de Newton-Cotes, de forma a ter um mesmo número de pontos avaliados da função. É inquestionável a vantagem da quadratura de Gauss-Legendre sobre a de Newton-Cotes.

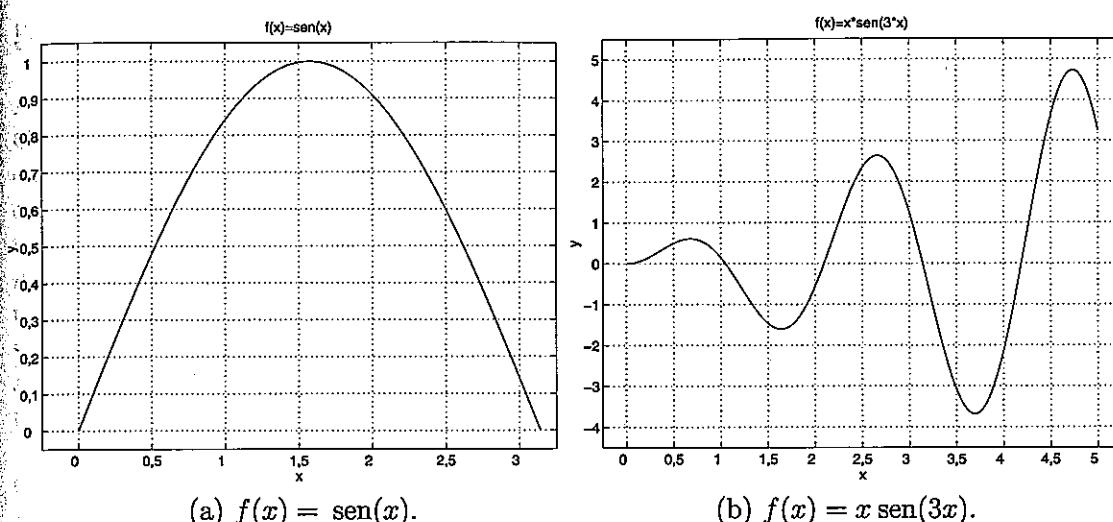


Figura 5.14 Funções usadas na comparação das regras de integração.

Tabela 5.5 Comparação entre Newton-Cotes e Gauss-Legendre.

| Grau do polinômio | Número de subintervalos | Newton-Cotes | Número de pontos | Gauss-Legendre |
|-------------------|-------------------------|------------------------|------------------|-------------------------|
| 1 | 1 | $2,000 \times 10^0$ | 2 | $6,418 \times 10^{-2}$ |
| 2 | 2 | $9,440 \times 10^{-2}$ | 3 | $1,389 \times 10^{-3}$ |
| 3 | 3 | $4,052 \times 10^{-2}$ | 4 | $1,577 \times 10^{-5}$ |
| 4 | 4 | $1,429 \times 10^{-3}$ | 5 | $1,103 \times 10^{-7}$ |
| 5 | 5 | $7,969 \times 10^{-4}$ | 6 | $5,227 \times 10^{-10}$ |
| 6 | 6 | $1,781 \times 10^{-5}$ | 7 | $1,791 \times 10^{-12}$ |
| 7 | 7 | $1,087 \times 10^{-5}$ | 8 | $4,441 \times 10^{-15}$ |
| 8 | 8 | $1,647 \times 10^{-7}$ | 9 | $4,441 \times 10^{-16}$ |

Seja agora a integral da função apresentada na Figura 5.14(b)

$$\int_0^5 x \sin(3x) dx = \left. \frac{\sin(3x)}{9} - \frac{x \cos(3x)}{3} \right|_0^5 \approx 1,3384.$$

A Figura 5.15(a) mostra uma tabela com a diferença entre o valor exato e o obtido pela regra do 1/3 de Simpson com vários números de subintervalos m e Gauss-Legendre com $m + 1$ pontos.

Por sua vez, a Figura 5.15(b) exibe um gráfico de \log_{10} da diferença entre o valor exato e o valor calculado pelos dois métodos de integração para cada valor de m . A partir de $m = 15$, os resultados obtidos por Gauss-Legendre estabilizam em torno da precisão do computador ($\approx 10^{-15}$), enquanto os valores da primeira regra de Simpson tendem lentamente para uma maior exatidão. Também por este exemplo é evidente a supremacia da quadratura de Gauss-Legendre sobre a primeira regra de Simpson.

| m | 1/3 de Simpson | Gauss-Legendre |
|-----|------------------------|-------------------------|
| 2 | $9,188 \times 10^0$ | $8,819 \times 10^0$ |
| 4 | $5,269 \times 10^0$ | $4,990 \times 10^{-1}$ |
| 6 | $5,221 \times 10^{-1}$ | $3,629 \times 10^{-3}$ |
| 8 | $1,093 \times 10^{-1}$ | $5,452 \times 10^{-6}$ |
| 10 | $3,799 \times 10^{-2}$ | $9,417 \times 10^{-10}$ |
| 12 | $1,688 \times 10^{-2}$ | $5,250 \times 10^{-12}$ |
| 14 | $8,689 \times 10^{-3}$ | $8,660 \times 10^{-15}$ |
| 16 | $4,944 \times 10^{-3}$ | $1,110 \times 10^{-15}$ |

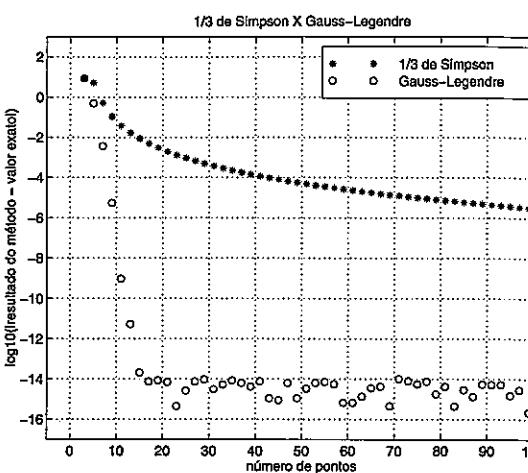
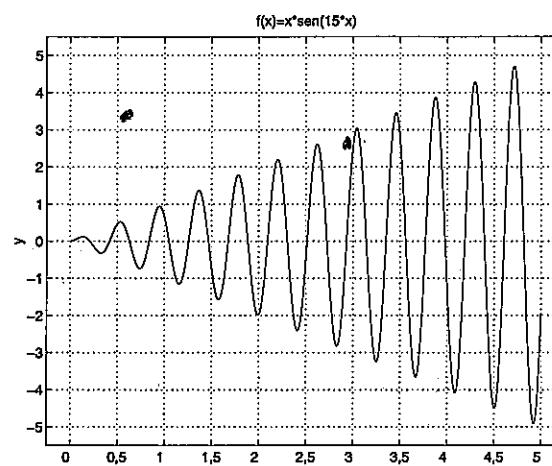
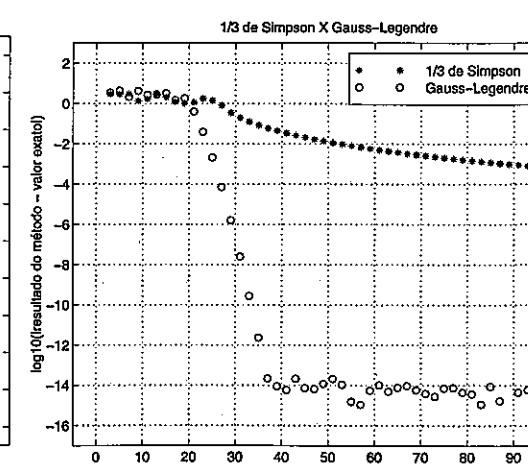
(a) $| \text{método} - \text{exato} |$ (b) $\log_{10}(|\text{método} - \text{exato}|) \times m$

Figura 5.15 Comparação entre a regra do 1/3 de Simpson e Gauss-Legendre.

Finalmente, considere a integral da função mostrada na Figura 5.16(a),

$$\int_0^5 x \sin(15x) dx = \left[\frac{\sin(15x)}{225} - \frac{x \cos(15x)}{15} \right]_0^5 \approx -0,3090.$$

Como esta é uma função menos suave que as outras duas, as fórmulas de integração necessitam de mais pontos para fornecerem uma exatidão aceitável, conforme a Figura 5.16(b). Somente para $m > 37$, Gauss-Legendre produz resultados que estabilizam em torno da precisão do computador. Todavia, para este exemplo, a primeira regra de Simpson está longe de gerar um resultado aceitável com $m < 100$.

(a) $f(x) = x \sin(15x)$.

(b) 1/3 de Simpson e Gauss-Legendre.

Figura 5.16 Integração de uma função não suave.

5.4 Integração numérica iterativa

As Figuras 5.15(b) e 5.16(b) evidenciam que as fórmulas de integração, principalmente de Gauss-Legendre, calculam a integral com grau crescente de exatidão à medida que aumenta o número de pontos. O algoritmo **Gauss-Legendre_iterativo**, apresentado na Figura 5.17, calcula a integral de uma função $f(x)$ de a até b , pelo método de Gauss-Legendre, utilizando um processo iterativo. Inicialmente, a integral é calculada com $n = 8$ pontos e depois com $n = 13$ pontos. Se a diferença relativa entre os dois valores da integral for menor ou igual a uma dada tolerância então o processo termina. Caso contrário, o valor de n é incrementado, seguindo uma série de Fibonacci, e a integral é calculada novamente. O processo repete até que a diferença relativa entre os dois últimos valores da integral seja menor ou igual à tolerância predefinida.

Os parâmetros de entrada do algoritmo da Figura 5.17 são os limites de integração inferior a e superior b , a tolerância $Toler$ e o número máximo de iterações $IterMax$. A função $f(x)$ deve ser especificada de acordo com a linguagem de programação escolhida. Os parâmetros de saída são o resultado da integração *Integral*, a menor diferença relativa obtida *Delta* e a condição de erro *CondErro*, em que *CondErro* = 0 significa que *Delta* \leq *Toler* e *CondErro* = 1 avisa que *Delta* $>$ *Toler*. O algoritmo **Gauss-Legendre** da Figura 5.13 é evocado para calcular a integral.

Exemplo 5.25 Calcular $\int_0^{20} x \sin(15x) dx$ utilizando o algoritmo mostrado na Figura 5.17, com uma tolerância de 10^{-10} e com, no máximo, 10 iterações. Um esboço desta função é exibido na Figura 5.16(a), para $0 \leq x \leq 5$.

```
% Os parametros de entrada
a = 0
b = 20
Toler = 1e-10
IterMax = 10
% produzem os resultados
    Integracao iterativa pelo metodo de Gauss-Legendre
    Iter      n          Integral        Dif. relativa
        1       8        32.7305341124
        2      13        24.9187432521  3.135e-01
        3      21       -16.8767733573  2.477e+00
        4      34        49.5529883366  1.341e+00
        5      55       -31.2365609799  2.586e+00
        6      89        0.0247820806  1.261e+03
        7     144        0.0250187998  9.462e-03
        8     233        0.0250187997  1.006e-11
```

Integral = 2.5018799750e-02

Delta = 1.00649e-11

CondErro = 0

Como CondErro = 0, isto significa que a integral foi calculada com sucesso, com uma diferença relativa menor ou igual à tolerância desejada.

```

Algoritmo Gauss-Legendre iterativo
{ Objetivo: Integrar uma função iterativamente pelo método de Gauss-Legendre }
parâmetros de entrada  $a, b, Toler, IterMax$ 
{ limite inferior, limite superior, tolerância e número máximo de iterações }
parâmetros de saída  $Integral, Delta, CondErro$ 
{ valor da integral, menor diferença relativa obtida e condição de erro, sendo }
{  $CondErro = 0$  se  $\Delta \leq Toler$  e  $CondErro = 1$  se  $\Delta > Toler$  }
Iter ← 1; n1 ← 5; n2 ← 8
[Int, CondErro] ← Gauss-Legendre(a, b, n2) (ver Figura 5.13)
escreva Iter, n2, Int
{ sucessivos cálculos das integrais }
repita
  Iter ← Iter + 1; n ← n1 + n2
  [Integral, CondErro] ← Gauss-Legendre(a, b, n)
  se Integral ≠ 0 então
    Delta ← abs((Integral - Int)/Integral)
  senão
    Delta ← abs(Integral - Int)
  fimse
  escreva Iter, n, Integral, Delta
  se Delta ≤ Toler ou Iter = IterMax então interrompa, fimse
  Int ← Integral; n1 ← n2; n2 ← n
fimrepita
{ teste de convergência }
se Delta ≤ Toler então
  CondErro ← 0
senão
  CondErro ← 1
fimse
fimalgoritmo

```

Figura 5.17 Integração iterativa pelo método de Gauss-Legendre.

(Ver significado da função abs na Tabela 1.1, na página 6.)

5.5 Integração dupla pelas fórmulas de Newton-Cotes

O cálculo de uma integral dupla definida

$$I = \int_a^b \int_c^d f(x, y) dy dx \quad (5.22)$$

é de grande valia na solução de problemas de diversas áreas. De modo similar à integração simples, a função integrando $f(x, y)$ pode ser aproximada por um polinômio interpolador, e a integral deste polinômio é, então, obtida analiticamente. O texto sobre integração dupla, a seguir, é baseado em um trabalho de Bocanegra, Medeiros e Campos [5].

Fazendo $G(x) = \int_c^d f(x, y) dy$ em (5.22), tem-se que

$$I = \int_a^b G(x) dx. \quad (5.23)$$

Desse modo, o cálculo de uma integral dupla consiste na solução de duas integrais simples.

5.5.1 Fórmulas simples

Para resolver uma integral simples, pode-se aplicar qualquer uma das fórmulas de Newton-Cotes. Se for utilizada a regra do 1/3 de Simpson em (5.23), tem-se

$$I = \int_a^b G(x) dx = \frac{1}{3} h_x (G(x_0) + 4G(x_1) + G(x_2)), \quad (5.24)$$

$$I = \frac{1}{3} h_x \sum_{i=0}^2 c_{x_i} G(x_i),$$

onde $h_x = (b - a)/2$, $c_{x_0} = c_{x_2} = 1$, $c_{x_1} = 4$ e $G(x_i) = \int_c^d f(x_i, y) dy$, $i = 0, 1, 2$.

Para o cálculo de $G(x_i)$ pode ser utilizada também qualquer uma das fórmulas de Newton-Cotes, como, por exemplo, a regra dos 3/8 de Simpson

$$\begin{aligned} G(x_i) &= \int_c^d f(x_i, y) dy \\ &= \frac{3}{8} h_y (f(x_i, y_0) + 3f(x_i, y_1) + 3f(x_i, y_2) + f(x_i, y_3)), \end{aligned}$$

$$G(x_i) = \frac{3}{8} h_y \sum_{j=0}^3 c_{y_j} f(x_i, y_j), \quad (5.25)$$

onde $h_y = (d - c)/3$, $c_{y_0} = c_{y_3} = 1$, $c_{y_1} = c_{y_2} = 3$ e $f(x_i, y_j)$ é o valor da função integrando no ponto (x_i, y_j) . Levando os valores de $G(x_i)$ dados por (5.25) em (5.24), obtém-se o valor da integral dupla

$$I = \frac{1}{3} h_x \frac{3}{8} h_y \sum_{i=0}^2 \sum_{j=0}^3 c_{x_i} c_{y_j} f(x_i, y_j). \quad (5.26)$$

Exemplo 5.26 Calcular $I = \int_0^{\frac{\pi}{2}} \int_0^{\frac{\pi}{4}} \sin(x+y) dy dx$.

Fazendo $G(x) = \int_0^{\frac{\pi}{4}} \sin(x+y) dy \rightarrow I = \int_0^{\frac{\pi}{2}} G(x) dx$. Utilizando a regra do 1/3 de Simpson em x , tem-se

$$I = \frac{1}{3} h_x (G(x_0) + 4G(x_1) + G(x_2)), \text{ com } h_x = \frac{b-a}{2} = \frac{\pi}{4}.$$

Para o cálculo de $G(x_i) = \int_0^{\frac{\pi}{4}} \sin(x_i + y) dy$, $i = 0, 1, 2$ e $x_i = a + ih_x = i\frac{\pi}{4}$, pode ser utilizada a regra dos 3/8 de Simpson

$$G(x_i) = \frac{3}{8}h_y(\sin(x_i + y_0) + 3\sin(x_i + y_1) + 3\sin(x_i + y_2) + \sin(x_i + y_3)),$$

com

$$h_y = \frac{d - c}{3} = \frac{\pi}{12} \text{ e } y_j = c + jh_y = j\frac{\pi}{12}.$$

$$\text{Para } x_0 = 0 \times \frac{\pi}{4} = 0$$

$$G(x_0) = \frac{3}{8}\frac{\pi}{12}(\sin(0 + y_0) + 3\sin(0 + y_1) + 3\sin(0 + y_2) + \sin(0 + y_3)),$$

$$G(x_0) = \frac{\pi}{32} \left(\sin(0) + 3\sin\left(\frac{\pi}{12}\right) + 3\sin\left(\frac{\pi}{6}\right) + \sin\left(\frac{\pi}{4}\right) \right) \sim$$

$$G(x_0) = 0,2929.$$

$$\text{Para } x_1 = 1 \times \frac{\pi}{4} = \frac{\pi}{4}$$

$$G(x_1) = \frac{3}{8}\frac{\pi}{12} \left(\sin\left(\frac{\pi}{4} + y_0\right) + 3\sin\left(\frac{\pi}{4} + y_1\right) + 3\sin\left(\frac{\pi}{4} + y_2\right) + \sin\left(\frac{\pi}{4} + y_3\right) \right),$$

$$G(x_1) = \frac{\pi}{32} \left(\sin\left(\frac{\pi}{4}\right) + 3\sin\left(\frac{\pi}{4} + \frac{\pi}{12}\right) + 3\sin\left(\frac{\pi}{4} + \frac{\pi}{6}\right) + \sin\left(\frac{\pi}{4} + \frac{\pi}{4}\right) \right) \sim$$

$$G(x_1) = 0,7071.$$

$$\text{Para } x_2 = 2 \times \frac{\pi}{4} = \frac{\pi}{2}$$

$$G(x_2) = \frac{3}{8}\frac{\pi}{12} \left(\sin\left(\frac{\pi}{2} + y_0\right) + 3\sin\left(\frac{\pi}{2} + y_1\right) + 3\sin\left(\frac{\pi}{2} + y_2\right) + \sin\left(\frac{\pi}{2} + y_3\right) \right),$$

$$G(x_2) = \frac{\pi}{32} \left(\sin\left(\frac{\pi}{2}\right) + 3\sin\left(\frac{\pi}{2} + \frac{\pi}{12}\right) + 3\sin\left(\frac{\pi}{2} + \frac{\pi}{6}\right) + \sin\left(\frac{\pi}{2} + \frac{\pi}{4}\right) \right) \sim$$

$$G(x_2) = 0,7071.$$

Considerando que

$$I = \frac{1}{3}h_x(G(x_0) + 4G(x_1) + G(x_2)),$$

$$I = \frac{1}{3}\frac{\pi}{4}(0,2929 + 4 \times 0,7071 + 0,7071) \sim I = 1,0023.$$

Resolvendo esta integral analiticamente, tem-se que

$$I = \int_0^{\frac{\pi}{2}} \int_0^{\frac{\pi}{4}} \sin(x + y) dy dx = - \int_0^{\frac{\pi}{2}} \cos(x + y) \Big|_0^{\frac{\pi}{4}} dx,$$

$$I = - \int_0^{\frac{\pi}{2}} \left(\cos\left(x + \frac{\pi}{4}\right) - \cos(x + 0) \right) dx = - \left[\sin\left(x + \frac{\pi}{4}\right) - \sin(x) \right] \Big|_0^{\frac{\pi}{2}},$$

$$I = - \left(\sin\left(\frac{\pi}{2} + \frac{\pi}{4}\right) - \sin\left(\frac{\pi}{2}\right) \right) + \left(\sin\left(0 + \frac{\pi}{4}\right) - \sin(0) \right) \sim I = 1.$$

Dispositivo prático

Os dados necessários para calcular uma integral dupla pelas fórmulas de Newton-Cotes podem ser sumarizados em um dispositivo prático. Se a regra do 1/3 de Simpson for utilizada para a integração em x e a regra dos 3/8 em y , ter-se-á um dispositivo como o mostrado na Tabela 5.6.

Tabela 5.6 Dispositivo prático para integração dupla por Newton-Cotes.

| | | j | 0 | 1 | 2 | 3 |
|-----|-----------|-----------------------------|---|---|---|---|
| | | y_j | c | $c + h_y$ | $c + 2h_y$ | $c + 3h_y$ |
| i | x_i | $c_{x_i} \setminus c_{y_j}$ | 1 | 3 | 3 | 1 |
| 0 | a | 1 | $c_{x_0} \times c_{y_0}$ $f(x_0, y_0)$ | $c_{x_0} \times c_{y_1}$ $f(x_0, y_1)$ | $c_{x_0} \times c_{y_2}$ $f(x_0, y_2)$ | $c_{x_0} \times c_{y_3}$ $f(x_0, y_3)$ |
| | | 4 | $c_{x_1} \times c_{y_0}$ $f(x_1, y_0)$ | $c_{x_1} \times c_{y_1}$ $f(x_1, y_1)$ | $c_{x_1} \times c_{y_2}$ $f(x_1, y_2)$ | $c_{x_1} \times c_{y_3}$ $f(x_1, y_3)$ |
| 1 | $a + h_x$ | 1 | $c_{x_2} \times c_{y_0}$ $f(x_2, y_0)$ | $c_{x_2} \times c_{y_1}$ $f(x_2, y_1)$ | $c_{x_2} \times c_{y_2}$ $f(x_2, y_2)$ | $c_{x_2} \times c_{y_3}$ $f(x_2, y_3)$ |

Fazendo

$$S = \sum_{i=0}^2 \sum_{j=0}^3 c_{x_i} c_{y_j} f(x_i, y_j), \text{ então por (5.26), tem-se } I = \frac{1}{3}h_x \frac{3}{8}h_y S,$$

onde S é a soma obtida, tomando-se todas as células da tabela, do produto $c_{x_i} \times c_{y_j}$ dos coeficientes de Cotes pelo valor da função $f(x_i, y_j)$.

Exemplo 5.27 Calcular $I = \int_0^{\frac{\pi}{2}} \int_0^{\frac{\pi}{4}} \sin(x + y) dy dx$ usando o dispositivo prático mostrado na Tabela 5.6.

Para tal,

$$h_x = \frac{b - a}{2} = \frac{\pi/2 - 0}{2} \rightarrow h_x = \frac{\pi}{4} \text{ e } x_i = a + ih_x = 0 + i\frac{\pi}{4} \rightarrow x_i = i\frac{\pi}{4},$$

$$h_y = \frac{d - c}{3} = \frac{\pi/4 - 0}{3} \rightarrow h_y = \frac{\pi}{12} \text{ e } y_j = c + jh_y = 0 + j\frac{\pi}{12} \rightarrow y_j = j\frac{\pi}{12}.$$

| | j | 0 | 1 | 2 | 3 | |
|-----|---------|-----------------------------|----------|---------|---------|--------|
| | y_j | 0 | $\pi/12$ | $\pi/6$ | $\pi/4$ | |
| i | x_i | $c_{x_i} \setminus c_{y_j}$ | 1 | 3 | 3 | 1 |
| 0 | 0 | 1 | 1 | 3 | 3 | 1 |
| | | | 0,0000 | 0,2588 | 0,5000 | 0,7071 |
| 1 | $\pi/4$ | 4 | 4 | 12 | 12 | 4 |
| | | | 0,7071 | 0,8660 | 0,9659 | 1,0000 |
| 2 | $\pi/2$ | 1 | 1 | 3 | 3 | 1 |
| | | | 1,0000 | 0,9659 | 0,8660 | 0,7071 |

Portanto, o valor da integral é

$$I = \frac{1}{3} h_x \frac{3}{8} h_y S = \frac{1}{3} \frac{\pi}{4} \frac{3}{8} \frac{\pi}{12} 38,9975 \leadsto I = 1,0023.$$

5.5.2 Fórmulas compostas

Para melhorar a exatidão de uma integral, deve-se subdividir o intervalo $[a, b]$ em m_x subintervalos iguais, sendo m_x múltiplo do grau n_x do polinômio interpolador utilizado para obter a regra de integração em x . No caso da regra do 1/3 de Simpson, m_x deve ser múltiplo de 2 ($= n_x$). Aplicando (5.24) a cada 3 ($= n_x + 1$) pontos, tem-se

$$\begin{aligned} I &= \int_a^b G(x) dx \\ &= \frac{1}{3} h_x (G(x_0) + 4G(x_1) + G(x_2)) + \frac{1}{3} h_x (G(x_2) + 4G(x_3) + G(x_4)) + \\ &\quad \dots + \frac{1}{3} h_x (G(x_{m_x-2}) + 4G(x_{m_x-1}) + G(x_{m_x})), \end{aligned}$$

$$\begin{aligned} I &= \frac{1}{3} h_x (G(x_0) + 4G(x_1) + 2G(x_2) + 4G(x_3) + 2G(x_4) + \dots \\ &\quad + 2G(x_{m_x-2}) + 4G(x_{m_x-1}) + G(x_{m_x})) \leadsto \end{aligned}$$

$$I = \frac{1}{3} h_x \sum_{i=0}^{m_x} c_{x_i} G(x_i), \quad (5.27)$$

onde $c_{x_0} = c_{x_{m_x}} = 1$, $c_{x_i} = 4$ para todo i ímpar, $c_{x_i} = 2$ para todo i par e $h_x = (b - a)/m_x$. Para o cálculo de $G(x_i)$, $i = 0, 1, \dots, m_x$ também pode ser utilizada qualquer uma das fórmulas de Newton-Cotes, como, por exemplo, a regra dos 3/8 de Simpson

$$G(x_i) = \int_c^d f(x_i, y) dy = \frac{3}{8} h_y (f(x_i, y_0) + 3f(x_i, y_1) + 3f(x_i, y_2) + f(x_i, y_3)). \quad (5.28)$$

Novamente, para uma melhor exatidão, também subdivide-se o intervalo $[c, d]$ em m_y subintervalos iguais, sendo m_y múltiplo do grau n_y do polinômio interpolador usado para construir a regra de integração em y no caso em questão múltiplo de 3 ($= n_y$). Aplicando-se (5.28) a cada 4 ($= n_y + 1$) pontos, tem-se

$$\begin{aligned} G(x_i) &= \int_c^d f(x_i, y) dy \\ &= \frac{3}{8} h_y (f(x_i, y_0) + 3f(x_i, y_1) + 3f(x_i, y_2) + f(x_i, y_3)) \\ &\quad + \frac{3}{8} h_y (f(x_i, y_3) + 3f(x_i, y_4) + 3f(x_i, y_5) + f(x_i, y_6)) + \dots \\ &\quad + \frac{3}{8} h_y (f(x_i, y_{m_y-3}) + 3f(x_i, y_{m_y-2}) + 3f(x_i, y_{m_y-1}) + f(x_i, y_{m_y})), \\ G(x_i) &= \frac{3}{8} h_y (f(x_i, y_0) + 3f(x_i, y_1) + 3f(x_i, y_2) + 2f(x_i, y_3) \\ &\quad + 3f(x_i, y_4) + 3f(x_i, y_5) + 2f(x_i, y_6) + \dots \\ &\quad + 2f(x_i, y_{m_y-3}) + 3f(x_i, y_{m_y-2}) + 3f(x_i, y_{m_y-1}) + f(x_i, y_{m_y})), \\ G(x_i) &= \frac{3}{8} h_y \sum_{j=0}^{m_y} c_{y_j} f(x_i, y_j), \quad i = 0, 1, 2, \dots, m_x, \end{aligned}$$

onde $c_{y_0} = c_{y_{m_y}} = 1$, e para os j 's restantes, $c_{y_j} = 2$ se j for múltiplo de 3 e $c_{y_j} = 3$ se j não for múltiplo de 3 e $h_y = (d - c)/m_y$. Levando os valores de $G(x_i)$ em (5.27), obtém-se o valor da integral dupla

$$I = \frac{1}{3} h_x \sum_{i=0}^{m_x} c_{x_i} \left(\frac{3}{8} h_y \sum_{j=0}^{m_y} c_{y_j} f(x_i, y_j) \right) \leadsto I = \frac{1}{3} h_x \frac{3}{8} h_y \sum_{i=0}^{m_x} \sum_{j=0}^{m_y} c_{x_i} c_{y_j} f(x_i, y_j).$$

Esta fórmula pode ser generalizada para qualquer grau do polinômio interpolador utilizado

$$I = \frac{n_x}{d_{n_x}} h_x \frac{n_y}{d_{n_y}} h_y \sum_{i=0}^{m_x} \sum_{j=0}^{m_y} c_{x_i} c_{y_j} f(x_i, y_j), \quad (5.29)$$

sendo os parâmetros $h_x = (b - a)/m_x$ e $h_y = (d - c)/m_y$. Os valores de d_{n_x} , d_{n_y} , c_{x_i} e c_{y_j} , para $n = 1, 2, \dots, 8$, são dados na Tabela 5.1.

5.5.3 Algoritmo

O algoritmo da Figura 5.18 calcula uma integral dupla pelas fórmulas de Newton-Cotes. Os parâmetros de entrada são ax (limite inferior de integração no eixo x), bx (limite superior de

integração no eixo x), nx (grau do polinômio utilizado para integração em x), mx (número de subintervalos para integração em x), ay (limite inferior de integração no eixo y), by (limite superior de integração no eixo y), ny (grau do polinômio utilizado para integração em y) e my (número de subintervalos para integração em y). A função $f(x, y)$ deve ser especificada de acordo com a linguagem de programação adotada. Os parâmetros de saída são *Integral* (valor da integral) e *CondErro* (condição de erro), em que *CondErro* = 0 significa que não houve erro de consistência nos parâmetros de entrada. Se *CondErro* = 1, então o grau do polinômio $nx < 1$ ou $nx > 8$ ou $ny < 1$ ou $ny > 8$, e se *CondErro* = 2, mx não é múltiplo de nx ou que my não é múltiplo de ny . O parâmetro *CondErro* = 3 se ambas condições ocorrerem.

Exemplo 5.28 Calcular $\int_2^5 \int_0^1 \sin(x^2 + y^2) dy dx$, utilizando o algoritmo da Figura 5.18, com $n_x = 3$ (regra dos 3/8), $m_x = 3$ subintervalos em x , $n_y = 2$ (regra do 1/3) e $m_y = 4$ subintervalos em y .

% Os parâmetros de entrada

```
ax = 2
bx = 5
nx = 3
mx = 3
ay = 0
by = 1
ny = 2
my = 4
```

% fornecem os resultados

| Integracao dupla por Newton-Cotes | | | | | | |
|-----------------------------------|-------------|------|---|-------------|------|--------------|
| i | x(i) | c(i) | j | y(j) | c(j) | f(x(i),y(j)) |
| 0 | 2.00000e+00 | 1 | 0 | 0.00000e+00 | 1 | -7.56802e-01 |
| | | | 1 | 2.50000e-01 | 4 | -7.96151e-01 |
| | | | 2 | 5.00000e-01 | 2 | -8.94989e-01 |
| | | | 3 | 7.50000e-01 | 4 | -9.88788e-01 |
| | | | 4 | 1.00000e+00 | 1 | -9.58924e-01 |
| 1 | 3.00000e+00 | 3 | 0 | 0.00000e+00 | 1 | 4.12118e-01 |
| | | | 1 | 2.50000e-01 | 4 | 3.54405e-01 |
| | | | 2 | 5.00000e-01 | 2 | 1.73889e-01 |
| | | | 3 | 7.50000e-01 | 4 | -1.37287e-01 |
| | | | 4 | 1.00000e+00 | 1 | -5.44021e-01 |
| 2 | 4.00000e+00 | 3 | 0 | 0.00000e+00 | 1 | -2.87903e-01 |
| | | | 1 | 2.50000e-01 | 4 | -3.47156e-01 |
| | | | 2 | 5.00000e-01 | 2 | -5.15882e-01 |
| | | | 3 | 7.50000e-01 | 4 | -7.54267e-01 |
| | | | 4 | 1.00000e+00 | 1 | -9.61397e-01 |
| 3 | 5.00000e+00 | 1 | 0 | 0.00000e+00 | 1 | -1.32352e-01 |
| | | | 1 | 2.50000e-01 | 4 | -7.01835e-02 |
| | | | 2 | 5.00000e-01 | 2 | 1.16990e-01 |
| | | | 3 | 7.50000e-01 | 4 | 4.16652e-01 |
| | | | 4 | 1.00000e+00 | 1 | 7.62558e-01 |

Integral = -0.78758

CondErro = 0

O resultado da integral é $I = -0,78758$ e o valor de *CondErro* indica que os parâmetros de entrada estão consistentes.

Algoritmo Newton-Cotes-Dupla

{ Objetivo: Cálculo de integral dupla pelas fórmulas de Newton-Cotes }

parâmetros de entrada *ax*, *bx*, *nx*, *mx*, *ay*, *by*, *ny*, *my*

{ limite inferior em x , limite superior em x , }

{ grau do polinômio em x , número de subintervalos em x , }

{ limite inferior em y , limite superior em y , }

{ grau do polinômio em y , número de subintervalos em y }

parâmetros de saída *Integral*, *CondErro*

{ valor da integral e condição de erro, sendo }

{ *CondErro* = 0 se não houve erro de consistência dos parâmetros dados, }

{ *CondErro* = 1 se ($n < 1$ ou $n > 8$), }

{ *CondErro* = 2 se resto(m, n) ≠ 0 e }

{ *CondErro* = 3 se ambas as condições ocorrerem. }

d(1) ← 2; d(2) ← 6; d(3) ← 8; d(4) ← 90; d(5) ← 238; d(6) ← 840

d(7) ← 17280; d(8) ← 28350

c(1) ← 1; c(2) ← 1; c(3) ← 4; c(4) ← 1; c(5) ← 3; c(6) ← 7; c(7) ← 32

c(8) ← 12; c(9) ← 19; c(10) ← 75; c(11) ← 50; c(12) ← 41; c(13) ← 216

c(14) ← 27; c(15) ← 272; c(16) ← 751; c(17) ← 3577; c(18) ← 1323

c(19) ← 2989; c(20) ← 989; c(21) ← 5888; c(22) ← -928; c(23) ← 10496

c(24) ← -4540

{ consistência dos parâmetros }

CondErro ← 0; Integral ← 0

se $nx < 1$ ou $nx > 8$ ou $ny < 1$ ou $ny > 8$ então *CondErro ← CondErro + 1*, fimse

se resto(mx, nx) ≠ 0 ou resto(my, ny) ≠ 0 então *CondErro ← CondErro + 2*, fimse

se *CondErro* ≠ 0 então abandone, fimse

{ cálculo da integral }

*px ← trunc((0,25 * (nx * (nx + 2) + resto(nx, 2))))*

*py ← trunc((0,25 * (ny * (ny + 2) + resto(ny, 2))))*

hx ← ((bx - ax)/mx; hy ← ((by - ay)/my)

para *i* ← 0 até *mx* faça

*x ← ax + i * hx; jx ← px + trunc((0,5 * nx - abs(resto(i, nx) - 0,5 * nx)))*

kx ← 1 + trunc(((nx - resto(i, nx))/nx) - trunc(((mx - resto(i, mx))/mx)))

para *j* ← 0 até *ny* faça

*y ← ay + j * hy; jy ← py + trunc((0,5 * ny - abs(resto(j, ny) - 0,5 * ny)))*

ky ← 1 + trunc(((ny - resto(j, ny))/ny) - trunc(((my - resto(j, my))/my)))

fxy ← f(x, y) { avaliar a função integrando em (x, y) }

*Integral ← Integral + fxy * c(jx) * kx * c(jy) * ky*

se *j* = 0 então escreva *i, x, c(jx) * kx, j, y, c(jy) * ky, fxy*

senão escreva *j, y, c(jy) * ky, fxy*

fimse

fimpara

fimpara

*Integral ← nx * ny * hx * hy / (d(nx) * d(ny)) + Integral*

finalgoritmo

Figura 5.18 Integração dupla pelas fórmulas de Newton-Cotes.

(Ver significado das funções abs, resto e trunc na Tabela 1.1, na página 6.)

5.6 Integração dupla via fórmulas de Gauss-Legendre

De modo similar à integração simples, as fórmulas de Gauss-Legendre também podem ser utilizadas para o cálculo aproximado da integral dupla definida (5.22). Fazendo $G(x) = \int_c^d f(x, y) dy$, tem-se que $I = \int_a^b G(x) dx$. Será mostrado, a seguir, que o cálculo de uma integral dupla por Gauss-Legendre também consiste na determinação de duas integrais simples.

5.6.1 Fórmula para dois pontos

Fazendo uma mudança de variável de x para t , sendo $-1 \leq t \leq 1$, tem-se

$$x = x(t) = \frac{b-a}{2}t + \frac{a+b}{2} \rightarrow dx = \frac{b-a}{2}dt.$$

Assim, pode-se tomar

$$x_i = \frac{b-a}{2}t_i + \frac{a+b}{2}.$$

Definindo

$$H(t) = \frac{b-a}{2}G(x(t)),$$

tem-se que

$$I = \int_a^b G(x) dx = \int_{-1}^1 \frac{2}{b-a} H(t) \frac{b-a}{2} dt = \int_{-1}^1 H(t) dt.$$

Resolvendo esta integral simples por Gauss-Legendre, com $n_x = 2$ pontos, obtém-se

$$I = \int_{-1}^1 H(t) dt = A_1 H(t_1) + A_2 H(t_2), \quad (5.30)$$

onde A_i , $i = 1, 2$ são os pesos e t_i são as abscissas ou os zeros do polinômio de Legendre de grau $n_x = 2$. Os valores de A_i e t_i podem ser obtidos na Tabela 5.3 ou gerados pelo algoritmo da Figura 5.12. Particularmente, para $n_x = 2$, tem-se que $A_1 = A_2 = 1$ e $t_1 = -1/\sqrt{3}$ e $t_2 = 1/\sqrt{3}$.

Para o cálculo de $G(x_i) = \int_c^d f(x_i, y) dy$ é feita uma mudança de variável de y para u tal que $-1 \leq u \leq 1$

$$y = y(u) = \frac{d-c}{2}u + \frac{c+d}{2} \rightarrow dy = \frac{d-c}{2}du$$

e, então, pode-se tomar

$$y_j = \frac{d-c}{2}u_j + \frac{c+d}{2}.$$

Definindo

$$F_i(u) = \frac{d-c}{2}f(x_i, y(u)),$$

tem-se que

$$G(x_i) = \int_c^d f(x_i, y) dy = \int_{-1}^1 \frac{2}{d-c} F_i(u) \frac{d-c}{2} du = \int_{-1}^1 F_i(u) du.$$

Usando, novamente, a fórmula para $n_y = 2$ pontos, obtém-se

$$G(x_i) = \int_{-1}^1 F_i(u) du = B_1 F_i(u_1) + B_2 F_i(u_2), \quad i = 1, 2,$$

onde B_j , $j = 1, 2$, são os pesos e $F_i(u_j) = \frac{d-c}{2}f(x_i, y_j)$, para $j = 1, 2$. Logo,

$$H(t_i) = \frac{b-a}{2}G(x_i) = \frac{b-a}{2}(B_1 F_i(u_1) + B_2 F_i(u_2)), \quad i = 1, 2.$$

Levando-se esses valores de $H(t_i)$ em (5.30), obtém-se

$$I = A_1 \left(\frac{b-a}{2} (B_1 F_1(u_1) + B_2 F_1(u_2)) \right) + A_2 \left(\frac{b-a}{2} (B_1 F_2(u_1) + B_2 F_2(u_2)) \right).$$

Substituindo $F_i(u_j)$, $j = 1, 2$, tem-se

$$\begin{aligned} I &= A_1 \left(\frac{b-a}{2} \left(B_1 \frac{d-c}{2} f(x_1, y_1) + B_2 \frac{d-c}{2} f(x_1, y_2) \right) \right) \\ &\quad + A_2 \left(\frac{b-a}{2} \left(B_1 \frac{d-c}{2} f(x_2, y_1) + B_2 \frac{d-c}{2} f(x_2, y_2) \right) \right). \end{aligned}$$

Rearranjando,

$$\begin{aligned} I &= \frac{(b-a)(d-c)}{2} (A_1 B_1 f(x_1, y_1) + A_1 B_2 f(x_1, y_2) + A_2 B_1 f(x_2, y_1) + \\ &\quad + A_2 B_2 f(x_2, y_2)), \end{aligned}$$

$$I = \frac{1}{4}(b-a)(d-c) \sum_{i=1}^2 A_i \sum_{j=1}^2 B_j f(x_i, y_j). \quad (5.31)$$

Dispositivo prático

A Tabela 5.7 mostra um dispositivo prático para sistematizar os dados necessários para calcular uma integral dupla pela fórmula de Gauss-Legendre com $n_x = 2$ pontos em x e $n_y = 2$ pontos em y . O valor da integral é

$$I = \frac{1}{4}(b-a)(d-c)S, \quad \text{sendo } S = \sum_{i=1}^2 A_i \sum_{j=1}^2 B_j f(x_i, y_j).$$

Tabela 5.7 Dispositivo prático para integração dupla por Gauss-Legendre.

| j | 1 | 2 |
|-------|---------------|---------------------|
| u_j | $-1/\sqrt{3}$ | $1/\sqrt{3}$ |
| y_j | y_1 | y_2 |
| i | t_i | x_i |
| | | $A_i \setminus B_j$ |
| 1 | $-1/\sqrt{3}$ | x_1 |
| 2 | $1/\sqrt{3}$ | x_2 |
| | | $f(x_1, y_1)$ |
| | | $f(x_2, y_1)$ |
| | | $f(x_1, y_2)$ |
| | | $f(x_2, y_2)$ |

Exemplo 5.29 Calcular $\int_0^{\frac{\pi}{2}} \int_0^{\frac{\pi}{4}} \sin(x+y) dy dx$ usando a fórmula de Gauss-Legendre para $n_x = n_y = 2$ pontos.

$$x_i = \frac{b-a}{2}t_i + \frac{a+b}{2} = \frac{\pi/2-0}{2}t_i + \frac{0+\pi/2}{2} \rightarrow x_i = \frac{\pi}{4}(t_i + 1) \text{ e}$$

$$y_j = \frac{d-c}{2}u_j + \frac{c+d}{2} = \frac{\pi/4-0}{2}u_j + \frac{0+\pi/4}{2} \rightarrow y_j = \frac{\pi}{8}(u_j + 1).$$

| j | 1 | 2 |
|-------|---------------|---------------------|
| u_j | $-1/\sqrt{3}$ | $1/\sqrt{3}$ |
| y_j | 0,1660 | 0,6194 |
| i | t_i | x_i |
| | | $A_i \setminus B_j$ |
| 1 | $-1/\sqrt{3}$ | x_1 |
| 2 | $1/\sqrt{3}$ | x_2 |
| | | 1 |
| | | 1 |
| | | 0,4776 |
| | | 0,8142 |
| | | 1,2388 |
| | | 1 |
| | | 0,9863 |
| | | 0,9590 |

A integral é

$$I = \frac{1}{4}(b-a)(d-c)S = \frac{1}{4}(\pi/2-0)(\pi/4-0) \times 3,2371 \sim I = 0,9984.$$

5.6.2 Fórmula geral

A Fórmula (5.31) para $n_x = n_y = 2$ pontos pode ser modificada para um número qualquer de pontos em x e em y , resultando na fórmula geral para integração dupla por Gauss-Legendre

$$I = \int_a^b \int_c^d f(x, y) dy dx = \frac{1}{4}(b-a)(d-c) \sum_{i=1}^{n_x} \sum_{j=1}^{n_y} A_i B_j f(x_i, y_j), \quad (5.32)$$

onde $x_i = \frac{b-a}{2}t_i + \frac{a+b}{2}$ e $y_j = \frac{d-c}{2}u_j + \frac{c+d}{2}$ e os pesos A_i , $i = 1, 2, \dots, n_x$ e B_j , $j = 1, 2, \dots, n_y$ e as abscissas t_i e u_j podem ser obtidos na Tabela 5.3 ou gerados pelo algoritmo mostrado na Figura 5.12.

Dispositivo prático

Um dispositivo prático para calcular uma integral dupla pela fórmula de Gauss-Legendre com n_x pontos em x e n_y em y é mostrado na Tabela 5.8. O valor da integral é, por (5.32),

$$I = \frac{1}{4}(b-a)(d-c)S, \text{ sendo } S = \sum_{i=1}^{n_x} A_i \sum_{j=1}^{n_y} B_j f(x_i, y_j).$$

Tabela 5.8 Dispositivo prático para integração dupla por Gauss-Legendre.

| j | 1 | 2 | ... | n_y |
|-------|-----------|-----------|---------------------|-----------------------|
| u_j | u_1 | u_2 | ... | u_{n_y} |
| y_j | y_1 | y_2 | ... | y_{n_y} |
| i | t_i | x_i | $A_i \setminus B_j$ | B_1 |
| 1 | t_1 | x_1 | A_1 | $f(x_1, y_1)$ |
| 2 | t_2 | x_2 | A_2 | $f(x_2, y_1)$ |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| n_x | t_{n_x} | x_{n_x} | A_{n_x} | $f(x_{n_x}, y_1)$ |
| | | | | $f(x_{n_x}, y_2)$ |
| | | | | ... |
| | | | | $f(x_{n_x}, y_{n_y})$ |

Exemplo 5.30 Calcular $\int_1^4 \int_0^2 (y^2 \log_{10}(3x)) dy dx$ usando Gauss-Legendre com $n_x = 3$ e $n_y = 4$.

$$x_i = \frac{b-a}{2}t_i + \frac{a+b}{2} = \frac{4-1}{2}t_i + \frac{1+4}{2} \sim x_i = 1,5t_i + 2,5 \text{ e}$$

$$y_j = \frac{d-c}{2}u_j + \frac{c+d}{2} = \frac{2-0}{2}u_j + \frac{0+2}{2} \sim y_j = u_j + 1.$$

| j | 1 | 2 | 3 | 4 |
|-------|---------|---------|---------------------|--------|
| u_j | -0,8611 | -0,3400 | 0,3400 | 0,8611 |
| y_j | 0,1389 | 0,6600 | 1,3400 | 1,8611 |
| i | t_i | x_i | $A_i \setminus B_j$ | B_1 |
| 1 | -0,7746 | 1,3381 | 0,5556 | 0,0116 |
| 2 | 0,0000 | 2,5000 | 0,8889 | 0,0169 |
| 3 | 0,7746 | 3,6619 | 0,5556 | 0,0201 |

O valor da integral é

$$I = \frac{1}{4}(b-a)(d-c)S = \frac{1}{4}(4-1)(2-0) \times 4,5107 \sim I = 6,7660.$$

5.6.3 Algoritmo

O algoritmo apresentado na Figura 5.19 calcula uma integral dupla pelas fórmulas de Gauss-Legendre. Os parâmetros de entrada são ax (limite inferior de integração no eixo x), bx (limite superior de integração no eixo x), nx (número de pontos usados na integração em x), ay (limite inferior de integração no eixo y), by (limite superior de integração no eixo y) e ny (número de pontos usados na integração em y). A função $f(x, y)$ deve ser especificada de acordo com a linguagem de programação adotada. Os parâmetros de saída são *Integral* (valor da integral) e *CondErro* (condição de erro), em que $CondErro = 0$ significa que não houve erro de consistência no número de pontos, ou seja, $nx \geq 1$ e $ny \geq 1$. Por outro lado, $CondErro = 1$ indica que $nx < 1$ ou $ny < 1$. O algoritmo *PesAbsGL* da Figura 5.12 é chamado para calcular os pesos e as abscissas. Obviamente, o algoritmo da Figura 5.19 pode ser adaptado de modo a utilizar valores fixos dos pesos e das abscissas para evitar o uso do algoritmo *PesAbsGL*.

Exemplo 5.31 Calcular $\int_2^6 \int_1^3 \sqrt{x^2 + y} \cos(xy) dy dx$ utilizando o algoritmo da Figura 5.19, com $n_x = 5$ e $n_y = 4$.

% Os parâmetros de entrada

$ax = 2$

$bx = 6$

$nx = 5$

$ay = 1$

$by = 3$

$ny = 4$

% fornecem os resultados

| Integracao dupla por Gauss-Legendre | | | | | | | | |
|-------------------------------------|---------|-----------|--------|---|---------|-----------|--------|--------------|
| i | t(i) | x(i) | A(i) | j | u(j) | y(j) | B(j) | f(x(i),y(j)) |
| 1 | -0.9062 | 2.188e+00 | 0.2369 | 1 | -0.8611 | 1.139e+00 | 0.3479 | -1.93747e+00 |
| | | | | 2 | -0.3400 | 1.660e+00 | 0.6521 | -2.24020e+00 |
| | | | | 3 | 0.3400 | 2.340e+00 | 0.6521 | 1.05584e+00 |
| | | | | 4 | 0.8611 | 2.861e+00 | 0.3479 | 2.76450e+00 |
| -2 | -0.5385 | 2.923e+00 | 0.4786 | 1 | -0.8611 | 1.139e+00 | 0.3479 | -3.05731e+00 |
| | | | | 2 | -0.3400 | 1.660e+00 | 0.6521 | 4.45596e-01 |
| | | | | 3 | 0.3400 | 2.340e+00 | 0.6521 | 2.80093e+00 |
| | | | | 4 | 0.8611 | 2.861e+00 | 0.3479 | -1.64659e+00 |
| 3 | 0.0000 | 4.000e+00 | 0.5689 | 1 | -0.8611 | 1.139e+00 | 0.3479 | -6.47030e-01 |
| | | | | 2 | -0.3400 | 1.660e+00 | 0.6521 | 3.93758e+00 |
| | | | | 3 | 0.3400 | 2.340e+00 | 0.6521 | -4.27352e+00 |
| | | | | 4 | 0.8611 | 2.861e+00 | 0.3479 | 1.88500e+00 |
| 4 | 0.5385 | 5.077e+00 | 0.4786 | 1 | -0.8611 | 1.139e+00 | 0.3479 | 4.54970e+00 |
| | | | | 2 | -0.3400 | 1.660e+00 | 0.6521 | -2.84341e+00 |
| | | | | 3 | 0.3400 | 2.340e+00 | 0.6521 | 4.10146e+00 |
| | | | | 4 | 0.8611 | 2.861e+00 | 0.3479 | -2.02780e+00 |
| 5 | 0.9062 | 5.812e+00 | 0.2369 | 1 | -0.8611 | 1.139e+00 | 0.3479 | 5.57848e+00 |
| | | | | 2 | -0.3400 | 1.660e+00 | 0.6521 | -5.80491e+00 |
| | | | | 3 | 0.3400 | 2.340e+00 | 0.6521 | 3.07129e+00 |
| | | | | 4 | 0.8611 | 2.861e+00 | 0.3479 | -3.65773e+00 |

Integral = 1.5683954289
CondErro = 0

Algoritmo Gauss-Legendre-Dupla

{ Objetivo: Integração dupla de função pelas fórmulas de Gauss-Legendre }

parâmetros de entrada ax , bx , nx , ay , by , ny

{ limite inferior em x , limite superior em x , número de pontos em x }

{ limite inferior em y , limite superior em y , número de pontos em y }

parâmetros de saída *Integral*, *CondErro*

{ valor da integral e condição de erro, sendo }

{ $CondErro = 0$ se não houve erro ($nx \geq 1$ e $ny \geq 1$) e }

{ $CondErro = 1$ se $nx < 1$ ou $ny < 1$ }

{ cálculo dos pesos e abscissas }

[*Avet*, *Tvet*, *CondErro*] \leftarrow *PesAbsGL*(*nx*) (ver Figura 5.12)

Integral $\leftarrow 0$; se $CondErro \neq 0$, *abandone*, *fimse*

se $ny = nx$ então

para $j \leftarrow 1$ até *trunca*($0,5 * (nx + 1)$) faça

Bvet(*j*) \leftarrow *Avet*(*j*); *Uvet*(*j*) \leftarrow *Tvet*(*j*)

fimpara

senão

[*Bvet*, *Uvet*, *CondErro*] \leftarrow *PesAbsGL*(*ny*)

se $CondErro \neq 0$, *abandone*, *fimse*

fimse

{ cálculo da integral dupla }

ex1 $\leftarrow (bx - ax)/2$; *ex2* $\leftarrow (ax + bx)/2$; *cy1* $\leftarrow (by - ay)/2$; *cy2* $\leftarrow (ay + by)/2$

se resto(*nx*, 2) = 0 então *cx1* $\leftarrow 1$; *cx2* $\leftarrow 0,5$, senão *cx1* $\leftarrow 0$; *cx2* $\leftarrow 1$, *fimse*

se resto(*ny*, 2) = 0 então *cy1* $\leftarrow 1$; *cy2* $\leftarrow 0,5$, senão *cy1* $\leftarrow 0$; *cy2* $\leftarrow 1$, *fimse*

para $i \leftarrow 1$ até *nx* faça

lx \leftarrow *trunca*($i - 0,5 * (nx + 1) + \text{sinal}(i - 0,5 * (nx + cx1)) * cx2$)

tx $\leftarrow \text{sinal}(i - 0,5 * (nx + cx1)) * \text{Avet}(\text{abs}(i - 0,5 * (nx + cx1)))$

x $\leftarrow ex1 * tx + ex2$; *Soma* $\leftarrow 0$

para $j \leftarrow 1$ até *ny* faça

ky \leftarrow *trunca*($j - 0,5 * (ny + 1) + \text{sinal}(j - 0,5 * (ny + cy1)) * cy2$)

ty $\leftarrow \text{sinal}(j - 0,5 * (ny + cy1)) * \text{Uvet}(\text{abs}(j - 0,5 * (ny + cy1)))$

y $\leftarrow cy1 * ty + cy2$

fx $\leftarrow f(x, y)$ { avaliar a função integrando em (x, y) }

Soma $\leftarrow Soma + Ayy * fx$

se $j = 1$ então escreva *i*, *tx*, *x*, *Axx*, *j*, *ty*, *y*, *Ayy*, *fx*

senão escreva *j*, *ty*, *y*, *Ayy*, *fx*, *fimse*

fimpara

Integral $\leftarrow Integral + Axx * Soma$

fimpara

Integral $\leftarrow ex1 * cy1 * Integral$

fimalgoritmo

Figura 5.19 Integração dupla pelas fórmulas de Gauss-Legendre.

(Ver significado das funções abs, resto, sinal e truncna na Tabela 1.1, na página 6.)

5.7 Comparação dos métodos para integração dupla

Serão utilizadas duas integrais para uma análise comparativa do desempenho das fórmulas de Newton-Cotes e Gauss-Legendre. O primeiro teste consiste em

$$\int_0^{\frac{\pi}{2}} \int_{-\frac{\pi}{2}}^{\pi} 2xy \sin(xy^2) dy dx,$$

cuja solução analítica é $I = (4 \sin(\pi^3/8) - \sin(\pi^3/2))/\pi^2 \approx -0,2921$.

A Tabela 5.9 mostra a diferença entre os resultados obtidos por diversos métodos de integração e o valor exato. As fórmulas de Newton-Cotes não foram compostas, isto é, o número de subintervalos $m_x (= m_y)$ é igual ao grau do polinômio $n_x (= n_y)$ utilizado para construir o método. Além do mais, para ter um mesmo número de avaliações da função integrando, o número de pontos usados em Gauss-Legendre é igual a $m_x + 1$.

Tabela 5.9 Integração dupla por Newton-Cotes e Gauss-Legendre.

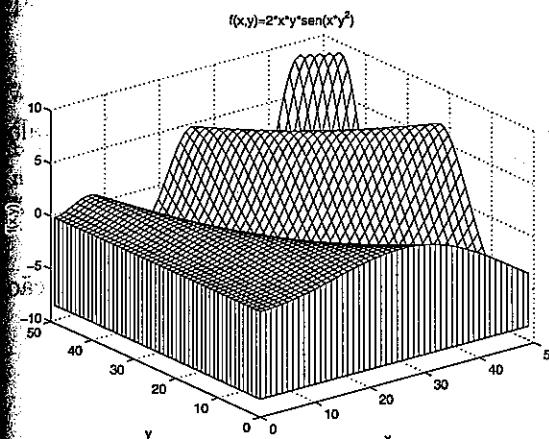
| Grau do polinômio | Número de subintervalos | Newton-Cotes | Número de pontos | Gauss-Legendre |
|-------------------|-------------------------|------------------------|------------------|------------------------|
| 1 | 1 | $5,090 \times 10^{-1}$ | 2 | $2,733 \times 10^0$ |
| 2 | 2 | $3,154 \times 10^{-1}$ | 3 | $1,123 \times 10^0$ |
| 3 | 3 | $6,787 \times 10^{-1}$ | 4 | $7,703 \times 10^{-2}$ |
| 4 | 4 | $4,335 \times 10^{-2}$ | 5 | $1,448 \times 10^{-2}$ |
| 5 | 5 | $1,644 \times 10^{-1}$ | 6 | $4,001 \times 10^{-3}$ |
| 6 | 6 | $1,480 \times 10^{-2}$ | 7 | $4,331 \times 10^{-4}$ |
| 7 | 7 | $2,935 \times 10^{-2}$ | 8 | $2,440 \times 10^{-5}$ |
| 8 | 8 | $2,500 \times 10^{-2}$ | 9 | $5,705 \times 10^{-7}$ |

A Figura 5.20(a) mostra um gráfico da função $f(x, y) = 2xy \sin(xy^2)$. Por sua vez, a Figura 5.20(b) exibe o desempenho de quatro métodos: Newton-Cotes com grau do polinômio $n_x (= n_y) = 2, 4$ e 8 com o número de subintervalos $m_x (= m_y)$ múltiplo do grau n_x e Gauss-Legendre com número de pontos $m_x + 1 = 3, 5, 7, \dots, 101$. A abscissa contém o número de pontos avaliados, e a ordenada, o logaritmo decimal da diferença entre o valor obtido pelo método e o valor exato. Com número de pontos ≥ 15 , o método de Gauss-Legendre produz resultados que se estabilizam em torno da precisão do computador, enquanto as fórmulas de Newton-Cotes só geram resultados satisfatórios para valores elevados de número de pontos.

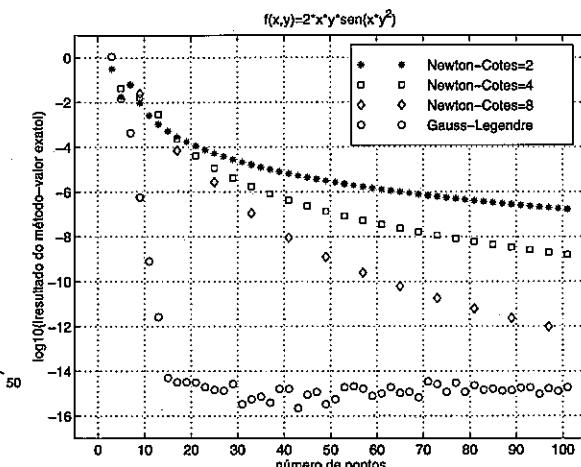
O segundo teste é a integral

$$\int_0^{\pi} \int_1^4 3y^2 \cos(x + y^3) dy dx,$$

com o valor exato $I = \cos(64) + \cos(\pi + 1) - \cos(1) - \cos(\pi + 64) \approx -0,2969$. A Figura 5.21(a) exibe um gráfico da função, a qual não é nada simples. A Figura 5.21(b) mostra que somente o método de Gauss-Legendre obteve valores mais exatos no cálculo dessa integral.

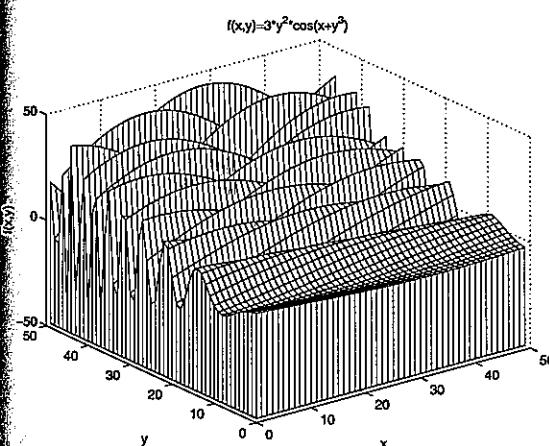


(a) Gráfico da função $f(x, y)$.

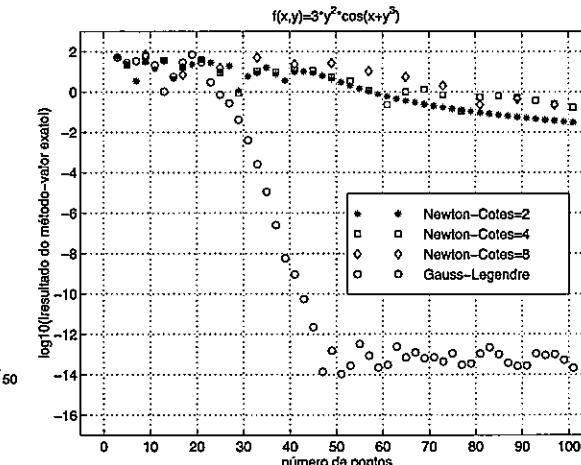


(b) Desempenho dos métodos.

$$\text{Figura 5.20 } \int_0^{\frac{\pi}{2}} \int_{-\frac{\pi}{2}}^{\pi} 2xy \sin(xy^2) dy dx.$$



(a) Gráfico da função $f(x, y)$.



(b) Desempenho dos métodos.

$$\text{Figura 5.21 } \int_0^{\pi} \int_1^4 3y^2 \cos(x + y^3) dy dx.$$

5.8 Exemplos de aplicação

Será mostrado, a seguir, como utilizar a integração numérica para calcular a densidade de uma distribuição de probabilidade. Além disso, será visto como calcular uma integral imprópria.

5.8.1 Distribuição de probabilidade

Definição do problema

Gerar uma tabela com os valores da distribuição de probabilidade normal padrão no intervalo $[0,6; 0,7]$, com incremento 0,005 e exibindo 5 decimais.

Modelagem matemática

A distribuição de probabilidade normal com média μ e variância σ^2 é calculada pela função integral

$$\phi(z) = \frac{1}{\sigma\sqrt{2\pi}} \int_{-\infty}^z e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx = 0,5 + \frac{1}{\sigma\sqrt{2\pi}} \int_0^z e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx.$$

A distribuição normal padrão possui média zero ($\mu = 0$) e variância unitária ($\sigma = 1$), resultando em

$$\phi(z) = 0,5 + \frac{1}{\sqrt{2\pi}} \int_0^z e^{-\frac{x^2}{2}} dx.$$

A Figura 5.22 apresenta o gráfico da função $f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$. A área sob a curva de $f(x)$, de $-\infty$ até x , fornece o valor procurado.

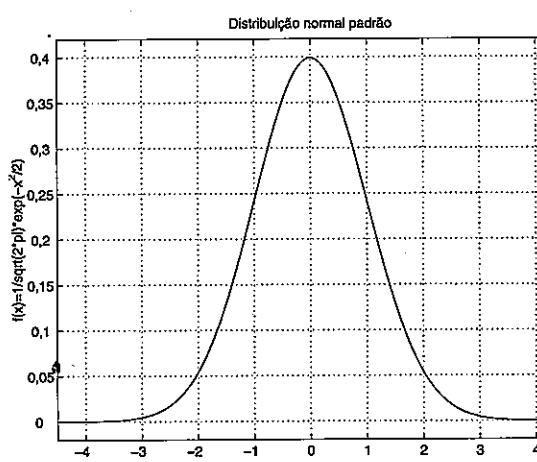


Figura 5.22 Distribuição de probabilidade normal padrão.

Solução numérica

A integral $\frac{1}{\sqrt{2\pi}} \int_0^z e^{-\frac{x^2}{2}} dx$ pode ser avaliada usando o algoritmo de quadratura de Gauss-Legendre mostrado na Figura 5.13. Considerando que os gráficos das Figuras 5.14(a) e 5.22 apresentam uma certa similaridade e os resultados da Tabela 5.5, então será utilizado $n = 5$.

Por exemplo, para $z = 0,6$

% Os parametros de entrada

a = 0

b = 0.6000

n = 5

% fornecem os resultados

Integracao numerica pelo metodo de Gauss-Legendre

| i | t(i) | x(i) | f(x(i)) | A(i) |
|---|----------|---------|---------|---------|
| 1 | -0.90618 | 0.02815 | 0.39878 | 0.23693 |
| 2 | -0.53847 | 0.13846 | 0.39514 | 0.47863 |
| 3 | 0.00000 | 0.30000 | 0.38139 | 0.56889 |
| 4 | 0.53847 | 0.46154 | 0.35864 | 0.47863 |
| 5 | 0.90618 | 0.57185 | 0.33877 | 0.23693 |

Integral = 0.2257468823

CondErro = 0

Portanto, $\phi(0,6) = 0,5 + 0,22575 = 0,72575$. Os valores calculados de $\phi(z)$, $0,6 \leq z \leq 0,7$ estão compilados na Tabela 5.10.

Tabela 5.10 Distribuição de probabilidade normal padrão.

| z | $\phi(z)$ | z | $\phi(z)$ | z | $\phi(z)$ |
|-------|-----------|-------|-----------|-------|-----------|
| 0,600 | 0,72575 | 0,635 | 0,73729 | 0,670 | 0,74857 |
| 0,605 | 0,72741 | 0,640 | 0,73891 | 0,675 | 0,75016 |
| 0,610 | 0,72907 | 0,645 | 0,74054 | 0,680 | 0,75175 |
| 0,615 | 0,73072 | 0,650 | 0,74215 | 0,685 | 0,75333 |
| 0,620 | 0,73237 | 0,655 | 0,74377 | 0,690 | 0,75490 |
| 0,625 | 0,73401 | 0,660 | 0,74537 | 0,695 | 0,75647 |
| 0,630 | 0,73565 | 0,665 | 0,74697 | 0,700 | 0,75804 |

Análise dos resultados

Os resultados apresentados na Tabela 5.10 estão de acordo com os valores publicados na literatura [1]. Portanto, à quadratura de Gauss-Legendre é um processo válido para gerar valores da distribuição de probabilidade normal padrão. Cumpre salientar, no entanto, que existem outros métodos para gerar estes valores [28].

5.8.2 Integral imprópria

Definição do problema

Calcular a integral imprópria $\int_1^\infty x^2 e^{-x^2} dx$, com 7 decimais exatas.

Modelagem matemática

Tanto as fórmulas de Newton-Cotes quanto a quadratura de Gauss-Legendre só devem ser aplicadas quando a função integrando $f(x)$ puder ser aproximada por um polinômio e for contínua no intervalo de integração.

No caso de integrais impróprias, uma maneira de resolver o problema é por meio de uma transformação de variável que conduza a um intervalo finito. Uma bela explanação sobre esta técnica é proposta por Acton [2].

Uma maneira alternativa de calcular uma integral imprópria $\int_a^\infty f(x) dx$ consiste em determinar intervalos $[a, b_1], [b_1, b_2], [b_2, b_3], \dots$, nos quais $f(x)$ apresenta um comportamento como um polinômio. Então, avalia-se a integral de $f(x)$ sobre esses intervalos, em seqüência,

$$\int_a^{b_1} f(x) dx + \int_{b_1}^{b_2} f(x) dx + \int_{b_2}^{b_3} f(x) dx + \dots + \int_{b_{n-1}}^{b_n} f(x) dx,$$

interrompendo quando o valor da última integral aproximar-se de zero.

Solução numérica

Um esboço da função $f(x) = x^2 e^{-x^2}$ é mostrado na Figura 5.23, onde se pode observar o comportamento assintótico de $f(x)$.

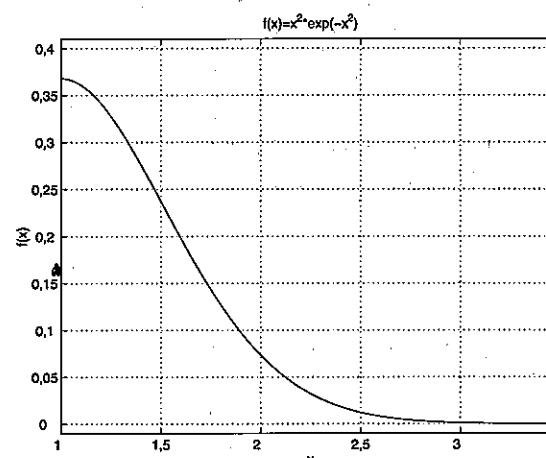


Figura 5.23 Integral imprópria.

A Tabela 5.11 apresenta os resultados da integração de $f(x) = x^2 e^{-x^2}$ sobre vários subintervalos, utilizando a quadratura de Gauss-Legendre com $n = 6$ (algoritmo da Figura 5.13).

Para o primeiro subintervalo $[1, 2]$

% Os parametros de entrada

a = 1

b = 2

n = 6

fornecem os resultados

Integracao numerica pelo metodo de Gauss-Legendre

| i | t(i) | x(i) | f(x(i)) | A(i) |
|---|----------|---------|---------|---------|
| 1 | -0.93247 | 1.03377 | 0.36705 | 0.17132 |
| 2 | -0.66121 | 1.16940 | 0.34836 | 0.36076 |
| 3 | -0.23862 | 1.38069 | 0.28333 | 0.46791 |
| 4 | 0.23862 | 1.61931 | 0.19049 | 0.46791 |
| 5 | 0.66121 | 1.83060 | 0.11744 | 0.36076 |
| 6 | 0.93247 | 1.96623 | 0.08096 | 0.17132 |

Integral = 0.2332527106

CondErro = 0

As integrais nos demais subintervalos são obtidas de modo similar, sendo os resultados mostrados na Tabela 5.11.

Tabela 5.11 Cálculo de uma integral imprópria.

| intervalo | integral | acumulado |
|-----------|----------------------------|---------------------------|
| [1, 2] | $2,332527 \times 10^{-1}$ | $2,332527 \times 10^{-1}$ |
| [2, 3] | $2,019350 \times 10^{-2}$ | $2,534462 \times 10^{-1}$ |
| [3, 5] | $1,949060 \times 10^{-4}$ | $2,536411 \times 10^{-1}$ |
| [5, 10] | $2,859342 \times 10^{-11}$ | $2,536411 \times 10^{-1}$ |
| [10, 20] | $3,546989 \times 10^{-45}$ | $2,536411 \times 10^{-1}$ |

Análise dos resultados

Pelos resultados da Tabela 5.11, nota-se que a partir da abscissa $x = 5$ o valor da integral é menor que 10^{-10} e, portanto, menor que a precisão de sete casas decimais. Assim,

$$\int_1^\infty x^2 e^{-x^2} dx \approx 0,2536411.$$

5.9 Exercícios

Seção 5.1

5.1. Calcular $\int_2^5 \frac{1}{x \log_e(x)} dx$ com $m = 6$ pelas regras abaixo.

- Trapézio.
- $1/3$ de Simpson.
- $3/8$ de Simpson.
- Comparar esses três resultados com o valor exato $\log_e(\log_e(5)) - \log_e(\log_e(2)) \approx 0,84240$.

5.2. Seja a função $f(x) = 10^x$.

- Achar o polinômio de Newton de grau 2 que passa pelos pontos de abscissas $-1, 0$ e 1 .
- Integrar, analiticamente, o polinômio obtido no item (a) no intervalo $[-1, 1]$.
- Calcular $\int_{-1}^1 10^x dx$ utilizando a regra do $1/3$ de Simpson com $m = 2$ subintervalos.
- Justificar por que os resultados dos itens (b) e (c) são iguais.

5.3. Calcular $\int_0^2 e^x dx$ com $E < 2 \times 10^{-3}$ usando uma das fórmulas de Newton-Cotes com o menor número de subintervalos.

5.4. Implementar o algoritmo apresentado na Figura 5.7 em qualquer linguagem de programação.

5.5. Avaliar $\int_0^3 (e^x + 2x) dx$ pela regra do $1/3$ de Simpson e com número de subintervalos $m = 2, 4, 6, 8$ usando o programa acima.

Seção 5.2

5.6. Calcular $\int_0^3 (e^x + 2x) dx$ utilizando quadratura de Gauss-Legendre com $n = 4$.

5.7. Calcular $\int_0^\pi (0,2x^4 + \sin(x) + 2) dx$ pela quadratura de Gauss-Legendre com $n = 5$.

5.8. Calcular $\int_0^\pi \sqrt{1+\cos(x)} dx$ usando quadra-

tura de Gauss-Legendre com $n = 7$.

5.9. Implementar, em qualquer linguagem de programação, o algoritmo da Figura 5.12 e o da Figura 5.13.

5.10. Resolver os Exercícios 5.6–5.8 usando os programas do Exercício 5.9.

Seção 5.3

Calcular as integrais dadas abaixo usando os métodos indicados e comparar o resultado com o valor analítico

5.11. $\int_0^1 e^x dx$ usando a regra do trapézio com $m = 1$ e Gauss-Legendre com $n = 2$.

5.12. $\int_0^3 (e^x + 2x) dx$ utilizando a regra do $1/3$ de Simpson com $m = 4$ e Gauss-Legendre com $n = 5$.

5.13. $\int_0^\pi (0,2x^4 + \sin(x) + 2) dx$ pela regra dos $3/8$ de Simpson com $m = 6$ e Gauss-Legendre com $n = 7$.

5.14. $\int_0^\pi \sqrt{1+\cos(x)} dx$ por Newton-Cotes que utiliza um polinômio de grau 4 com $m = 8$ e Gauss-Legendre com $n = 9$.

5.15. $\int_0^3 xe^x dx$ pelo método de Newton-Cotes que utiliza um polinômio de grau 5 com $m = 10$ e Gauss-Legendre com $n = 11$.

Seção 5.4

5.16. Implementar o algoritmo da Figura 5.17 em qualquer linguagem de programação.

Calcular as integrais a seguir usando o programa do exercício anterior, com $Toler = 10^{-10}$ e $IterMax = 10$.

5.17. $\int_0^\pi \sqrt{1+\cos(x)} dx$.

5.18. $\int_2^5 \frac{1}{x \log_e(x)} dx$.

5.19. $\int_0^\pi e^{1-x^2} \sin(10x) dx$.

5.20. $\int_{-\pi}^\pi \cos(2 + \cos^2(x)) dx$.

Seção 5.5

5.21. Calcular $\int_0^\pi \int_{-1}^1 \sin(x)y^2 dy dx$ usando a fórmula de Newton-Cotes com $n_x = n_y = 2$ e $m_x = m_y = 4$.

5.22. Avaliar $\int_0^3 \int_1^4 \sqrt{x+y} - e^{x-y} dy dx$ pelas fórmulas de Newton-Cotes com $n_x = 2$, $n_y = 3$ e $m_x = m_y = 6$.

5.23. Determinar

$\int_0^\pi \int_0^\pi \cos(x^2 + 1) \sin(xy) + x dy dx$ pelas fórmulas de Newton-Cotes com $n_x = 5$, $m_x = 5$, $n_y = 2$ e $m_y = 4$.

5.24. Implementar, em qualquer linguagem de programação, o algoritmo apresentado na Figura 5.18.

5.25. Calcular as integrais dos Exercícios 5.21 a 5.23 utilizando o programa implementado no Exercício 5.24.

Seção 5.6

5.26. Avaliar $\int_0^2 \int_1^3 \frac{\sqrt{xy}}{x+y} dy dx$ pela quadratura de Gauss-Legendre com $n_x = n_y = 2$. Exato = $3 \log_e(3/2)/2 \approx 0,60820$.

5.27. Calcular $\int_0^1 \int_0^2 e^{x^2-y^2} \sin(x+y) dy dx$ via quadratura de Gauss-Legendre com $n_x = 3$ e $n_y = 4$.

5.28. Determinar

$\int_1^2 \int_0^2 (x^3y^2 + x^2y) \log_e(xy) dy dx$

pela quadratura de Gauss-Legendre com $n_x = n_y = 4$.

5.29. Utilizando qualquer linguagem de programação, implementar o algoritmo da Figura 5.19.

5.30. Calcular as integrais dadas nos Exercícios 5.26 a 5.28 usando o programa elaborado no Exercício 5.29.

Seção 5.7

Utilizando os programas implementados a partir dos algoritmos das Figuras 5.18 e 5.19, comparar os resultados das integrais abaixo com o valor exato fornecido.

5.31. $\int_0^1 \int_0^1 (x^3y + xy^3) dy dx$ usando Newton-Cotes com $n_x = n_y = m_x = m_y = 1$ e Gauss-Legendre com $n_x = n_y = 2$. Exato = $1/4$.

5.32. $\int_{-1}^1 \int_0^1 (3x^5y^5 + x^4y + xy) dy dx$ usando Newton-Cotes com $n_x = n_y = m_x = m_y = 2$ e Gauss-Legendre com $n_x = n_y = 3$. Exato = $1/5$.

5.33. $\int_0^\pi \int_{-\pi}^\pi \sin(x-y) dy dx$ usando Newton-Cotes com $n_x = n_y = m_x = m_y = 3$ e Gauss-Legendre com $n_x = n_y = 4$. Exato = 0 .

5.34. $\int_1^2 \int_0^1 \frac{x}{2+y} dy dx$ usando Newton-Cotes com $n_x = n_y = m_x = m_y = 1$ e Gauss-Legendre com $n_x = n_y = 2$. Exato = $3 \log_e(3/2)/2 \approx 0,60820$.

5.35. $\int_0^1 \int_{-1}^1 x\sqrt{1+xy} dy dx$ usando Newton-Cotes com $n_x = n_y = m_x = m_y = 4$ e Gauss-Legendre com $n_x = n_y = 5$. Exato = $8(2\sqrt{2}-1)/15 \approx 0,97516$.

Gerais

5.36. Seja a função $f(x) = \sqrt{x} + 1$.

a) Achar o polinômio de Lagrange de grau 1 que passa pelos pontos de abscissas 0 e 1.

b) Integrar, analiticamente, o polinômio obtido no item (a) no intervalo $[0, 1]$.

c) Calcular $\int_0^1 \sqrt{x} + 1 \, dx$ utilizando a regra do trapézio.

d) Justificar por que os resultados dos itens (b) e (c) são iguais.

5.37. Deduzir a fórmula de integração numérica que utiliza um polinômio interpolador de grau 4 e comparar seus coeficientes com aqueles compilados na Tabela 5.1.

5.38. Propor a fórmula composta de Newton-Cotes para a regra de integração baseada em um polinômio interpolador de grau 4.

5.39. Seja a integral $\int_0^2 (4x^3 + 2x + 1) \, dx$.

a) Avaliar pela regra do 1/3 de Simpson com $m = 2$.

b) O resultado é exato? Por quê?

5.40. Considere a integral $\int_0^\pi (x^2 + \sin(x)) \, dx$.

a) Determinar o número de subintervalos para que a regra dos 3/8 de Simpson calcule a integral com $E < 10^{-3}$.

b) Calcular utilizando o programa implementado no Exercício 5.16.

c) Comparar os resultados acima com o valor exato.

5.41. Deduzir a fórmula do erro de integração para a regra do 1/3 de Simpson composta.

5.42. Deduzir a fórmula do erro de integração para a regra dos 3/8 de Simpson composta.

5.43. Considere a integral

$$\int_1^3 \int_{-1}^1 (4x^3y^3 + 2x^2y^2 + 1) \, dy \, dx.$$

a) Calcular pela fórmula de Newton-Cotes com $n_x = n_y = m_x = m_y = 1$.

b) Calcular por Gauss-Legendre com $n_x = n_y = 2$.

c) Verificar a exatidão dos dois resultados acima.

d) Por que um dos resultados é exato e o outro não?

5.44. Gerar uma tabela com os valores da distribuição de probabilidade normal padrão no intervalo $[0,5; 0,6]$, com incremento 0,005 e exibindo 5 decimais.

5.45. Calcular a integral $\int_0^\infty \sqrt{x}e^{-x} \, dx$.

Capítulo 6

Raízes de equações

A necessidade de encontrar valores de $x = \xi$ que satisfazem à equação $f(x) = 0$ aparece freqüentemente em uma ampla variedade de problemas provenientes das Ciências e das Engenharias. Esses valores especiais são chamados de raízes da equação $f(x) = 0$ ou zeros da função $f(x)$, os quais podem ser vistos na Figura 6.1. Para equações algébricas de grau até quatro, suas

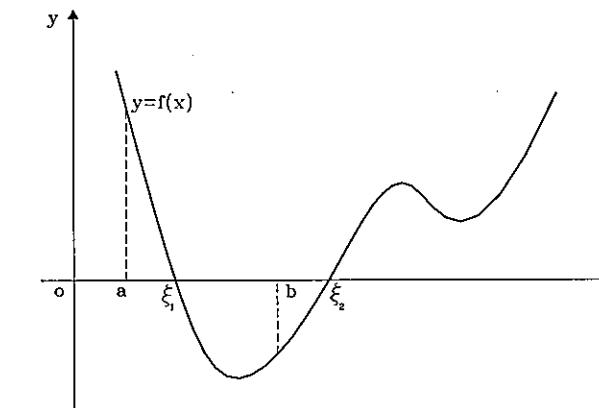


Figura 6.1 Raízes da equação $f(x) = 0$.

raízes podem ser calculadas por meio de uma expressão, tal como $x = (-b \pm \sqrt{b^2 - 4ac})/2a$ para determinar as duas raízes de $f(x) = ax^2 + bx + c = 0$. No entanto, para equações algébricas de grau superior a quatro e para a grande maioria das equações transcendentais, as raízes não podem ser calculadas analiticamente. Nesses casos, têm que ser usados métodos que encontrem uma solução aproximada para essas raízes.

6.1 Isolamento de raízes

O problema de calcular uma raiz pode ser dividido em duas fases

1. Isolamento da raiz, isto é, encontrar um intervalo $[a, b]$ que contenha uma, e somente uma, raiz de $f(x) = 0$ (ver Figura 6.1).
2. Refinamento da raiz, ou seja, a partir de um valor inicial $x_0 \in [a, b]$, gerar uma seqüência $\{x_0, x_1, x_2, \dots, x_k, \dots\}$ que converja para uma raiz exata ξ de $f(x) = 0$.

Existem alguns métodos para cálculo das raízes de equações polinomiais que não requerem que haja um prévio isolamento de cada raiz. No entanto, a maioria dos métodos para cálculo de raízes necessita que a mesma esteja confinada em um dado intervalo e, além do mais, essa raiz deve ser única em tal intervalo. Devido à existência de alguns teoremas da Álgebra que fornecem importantes informações sobre as equações polinomiais, o isolamento de raízes das equações algébricas e das transcendentas será visto separadamente.

6.1.1 Equações algébricas

Seja uma equação algébrica de grau n , $n \geq 1$, escrita na forma de potências

$$P(x) = c_n x^n + c_{n-1} x^{n-1} + c_{n-2} x^{n-2} + \dots + c_2 x^2 + c_1 x + c_0 = 0 \quad (6.1)$$

com os coeficientes c_i sendo reais e $c_n \neq 0$. Antes de serem abordados métodos para determinar o número e os limites das raízes reais de uma equação polinomial, serão vistos modos de avaliar um polinômio e algumas propriedades desse tipo de função.

Avaliação de polinômio

Para obter o valor de um polinômio $P(x) = c_n x^n + c_{n-1} x^{n-1} + c_{n-2} x^{n-2} + \dots + c_2 x^2 + c_1 x + c_0$ em um ponto $x = a$, usualmente, se faz

$$P(a) = c_n a^n + c_{n-1} a^{n-1} + c_{n-2} a^{n-2} + \dots + c_2 a^2 + c_1 a + c_0.$$

Dessa maneira, para avaliar $P(x)$ de grau n , em $x = a$, são necessárias $n(n+1)/2$ multiplicações e n adições.

Exemplo 6.1 Avaliar $P(x) = 3x^5 - 2x^4 + 5x^3 + 7x^2 - 3x + 1$ em $x = 2$.

$$P(2) = 3 \times 2^5 - 2 \times 2^4 + 5 \times 2^3 + 7 \times 2^2 - 3 \times 2 + 1 = 127,$$

sendo requeridas 15 multiplicações e 5 adições.

No entanto, uma maneira mais eficiente de avaliar um polinômio é o método de Horner, que consiste em reescrever o polinômio de forma a evitar potências.

Desse modo,

$$\begin{aligned} P(x) &= c_n x^n + c_{n-1} x^{n-1} + c_{n-2} x^{n-2} + \dots + c_2 x^2 + c_1 x + c_0, \\ &= (c_n x^{n-1} + c_{n-1} x^{n-2} + c_{n-2} x^{n-3} + \dots + c_2 x + c_1)x + c_0, \\ &= ((c_n x^{n-2} + c_{n-1} x^{n-3} + c_{n-2} x^{n-4} + \dots + c_2)x + c_1)x + c_0, \\ &\quad \dots \\ P(x) &= (\underbrace{\dots (c_n x + c_{n-1})x + c_{n-2}}_{n-1} x + \dots + c_2)x + c_1)x + c_0. \end{aligned}$$

O processo de Horner requer apenas n multiplicações e n adições para avaliar um polinômio de grau n .

Exemplo 6.2 Avaliar $P(x) = 3x^5 - 2x^4 + 5x^3 + 7x^2 - 3x + 1$ em $x = 2$ usando o processo de Horner.

$$P(x) = (((((3x - 2)x + 5)x + 7)x - 3)x + 1,$$

$$P(2) = (((((3 \times 2 - 2) \times 2 + 5) \times 2 + 7) \times 2 - 3) \times 2 + 1 = 127.$$

A Figura 6.2 mostra o algoritmo de Horner para avaliar um polinômio de grau n no ponto $x = a$. Os parâmetros de entrada são o grau n do polinômio, o vetor c contendo os coeficientes, sendo $P(x) = c(1)x^n + c(2)x^{n-1} + \dots + c(n)x + c(n+1)$ e o ponto a onde $P(x)$ deve ser avaliado. O parâmetro de saída é a ordenada $y = P(a)$.

```

Algoritmo Horner
{ Objetivo: Avaliar um polinômio de grau n no ponto a }
parâmetros de entrada n, c, a
{ grau, coeficientes e ponto a ser avaliado, onde c é tal que }
{ P(x) = c(1)x^n + c(2)x^{n-1} + ... + c(n)x + c(n+1) }
parâmetros de saída y { ordenada P(a) }
y ← c(1)
para i ← 2 até n + 1 faça
    y ← y * a + c(i)
fimpara
finalgoritmo

```

Figura 6.2 Algoritmo de Horner para avaliar polinômio.

Exemplo 6.3 Avaliar o polinômio $P(x) = 3x^5 - 2x^4 + 5x^3 + 7x^2 - 3x + 1$ do Exemplo 6.2, em $x = 2$ usando o algoritmo da Figura 6.2.

```
% Os parametros de entrada
n = 5
c = 3   -2      5      7      -3      1
a = 2
% produzem o resultado
y = 127
```

Propriedades gerais

Teorema 6.1 Uma equação algébrica de grau n tem exatamente n raízes, reais ou complexas, contando cada raiz de acordo com a sua multiplicidade.

A demonstração deste teorema é dada por Uspensky [38]. Uma raiz ξ de (6.1) tem multiplicidade m se

$$P(\xi) = P'(\xi) = P''(\xi) = \dots = P^{m-1}(\xi) = 0 \text{ e}$$

$$P^m(\xi) \neq 0$$

sendo

$$P^i(\xi) = \frac{d^i P(x)}{dx^i} \Big|_{x=\xi}, i = 1, 2, \dots, m.$$

Exemplo 6.4 Seja:

$$P(x) = x^4 + 2x^3 - 12x^2 + 14x - 5 \rightarrow P(1) = 0,$$

$$P'(x) = 4x^3 + 6x^2 - 24x + 14 \rightarrow P'(1) = 0,$$

$$P''(x) = 12x^2 + 12x - 24 \rightarrow P''(1) = 0 \text{ e}$$

$$P'''(x) = 24x + 12 \rightarrow P'''(1) = 36.$$

Assim, $\xi = 1$ é uma raiz de multiplicidade $m = 3$. Considerando também que $P(-5) = 0$, o polinômio de grau 4 acima pode ser escrito na forma fatorada $P(x) = (x - 1)^3(x + 5)$.

Teorema 6.2 Se os coeficientes de uma equação algébrica forem reais, então suas raízes complexas serão complexos conjugados em pares, ou seja, se $\xi_1 = a + bi$ for uma raiz de multiplicidade m , então $\xi_2 = a - bi$ também será uma raiz e com a mesma multiplicidade.

Exemplo 6.5 As raízes de $P(x) = x^2 - 4x + 13 = 0$ são

$$\xi = \frac{4 \pm \sqrt{(-4)^2 - 4 \times 1 \times 13}}{2} \rightarrow \begin{cases} \xi_1 = 2 + 3i \\ \xi_2 = 2 - 3i. \end{cases}$$

Corolário 6.1 Uma equação algébrica de grau ímpar com coeficientes reais tem, no mínimo, uma raiz real.

Exemplo 6.6 As raízes da equação $P(x) = x^3 - 9x^2 + 33x - 65 = 0$ são $\xi_1 = 5$, $\xi_2 = 2 + 3i$ e $\xi_3 = 2 - 3i$. Portanto, esta equação de grau 3 tem uma raiz real.

Relação entre raízes e coeficientes

Se ξ_i , $i = 1, 2, \dots, n$ forem as raízes de $P(x) = 0$, então ela pode ser escrita na forma fatorada

$$P(x) = c_n(x - \xi_1)(x - \xi_2) \dots (x - \xi_n) = 0.$$

Multiplicando os fatores,

$$\begin{aligned} P(x) &= c_n x^n - c_n(\xi_1 + \xi_2 + \dots + \xi_n)x^{n-1} \\ &\quad + c_n(\xi_1\xi_2 + \xi_1\xi_3 + \dots + \xi_1\xi_n + \xi_2\xi_3 + \dots + \xi_2\xi_n + \dots + \xi_{n-1}\xi_n)x^{n-2} \\ &\quad - c_n(\xi_1\xi_2\xi_3 + \xi_1\xi_2\xi_4 + \dots + \xi_1\xi_2\xi_n + \xi_1\xi_3\xi_4 + \dots + \xi_{n-2}\xi_{n-1}\xi_n)x^{n-3} \\ &\quad + \dots (-1)^n c_n(\xi_1\xi_2\xi_3 \dots \xi_n) = 0. \end{aligned}$$

Comparando a expressão acima com $P(x) = 0$ escrita na forma de potências (6.1) e aplicando a condição de igualdade das equações algébricas, tem-se que

$$\xi_1 + \xi_2 + \dots + \xi_n = -\frac{c_{n-1}}{c_n},$$

$$\xi_1\xi_2 + \xi_1\xi_3 + \dots + \xi_1\xi_n + \xi_2\xi_3 + \dots + \xi_2\xi_n + \dots + \xi_{n-1}\xi_n = \frac{c_{n-2}}{c_n},$$

$$\xi_1\xi_2\xi_3 + \xi_1\xi_2\xi_4 + \dots + \xi_1\xi_2\xi_n + \xi_1\xi_3\xi_4 + \dots + \xi_{n-2}\xi_{n-1}\xi_n = -\frac{c_{n-3}}{c_n},$$

...

$$\xi_1\xi_2\xi_3 \dots \xi_n = (-1)^n \frac{c_0}{c_n}.$$

As expressões acima relacionando os coeficientes de uma equação algébrica com as suas raízes são conhecidas como relações de Girard. Essas relações são válidas também para as raízes complexas.

Exemplo 6.7 As raízes da equação do Exemplo 6.6, $P(x) = x^3 - 9x^2 + 33x - 65 = 0$, são $\xi_1 = 5$, $\xi_2 = 2 + 3i$ e $\xi_3 = 2 - 3i$, assim as relações de Girard são

$$5 + (2 + 3i) + (2 - 3i) = 9 = -\frac{-9}{1},$$

$$5(2 + 3i) + 5(2 - 3i) + (2 + 3i)(2 - 3i) = 33 = \frac{33}{1} \text{ e}$$

$$5(2 + 3i)(2 - 3i) = 65 = (-1)^3 \frac{-65}{1}.$$

Exemplo 6.8 Sejam as equações algébricas de Lagrange definidas pela fórmula de recorrência (5.18) a partir de $L_0(x) = 1$ e $L_1(x) = x$

$$L_2(x) = \frac{3x^2 - 1}{2} = 0,$$

$$L_3(x) = \frac{5x^3 - 3x}{2} = 0,$$

$$L_4(x) = \frac{35x^4 - 30x^2 + 3}{8} = 0,$$

$$L_5(x) = \frac{63x^5 - 70x^3 + 15x}{8} = 0.$$

Todas as equações possuem $c_{n-1} = 0$, implicando, pelas relações de Girard, que a soma das raízes é nula, ou seja, as raízes são simétricas em relação à origem. Também, as equações de grau ímpar possuem $c_0 = 0$; como consequência, elas têm uma raiz nula. Essas propriedades podem ser verificadas na Figura 5.11, na página 235. ■

Limites das raízes reais

Uma equação algébrica na forma (6.1) pode ter suas raízes reais delimitadas usando o Teorema de Lagrange, cuja demonstração é dada por Demidovich e Maron [10].

Teorema 6.3 (Lagrange) Dada a equação $P(x) = c_n x^n + c_{n-1} x^{n-1} + c_{n-2} x^{n-2} + \dots + c_2 x^2 + c_1 x + c_0 = 0$, se $c_n > 0$ e k ($0 \leq k \leq n-1$) for o maior índice de coeficiente escolhido dentre os coeficientes negativos, então o limite superior das raízes positivas de $P(x) = 0$ pode ser dado por

$$L = 1 + \sqrt[n-k]{\frac{|B|}{c_n}},$$

onde B é o valor absoluto do maior coeficiente negativo em módulo.

Desse modo, se ξ_p for a maior das raízes positivas de $P(x) = 0$, então $\xi_p \leq L$.

Exemplo 6.9 Seja $P(x) = x^4 + 2x^3 - 13x^2 - 14x + 24 = 0$. Os coeficientes negativos são $c_2 = -13$ e $c_1 = -14$, portanto, $k = 2$, pois $2 > 1$, $B = |-14|$ e

$$L = 1 + \sqrt[4-2]{\frac{14}{1}} \rightarrow L = 4,74.$$

O Teorema de Lagrange garante que $P(x) = 0$ não tem nenhuma raiz maior que 4,74. ■

Se $c_i > 0$ ($i = 0, 1, \dots, n$), então $P(x) = 0$ não tem raízes positivas, pois $P(x) = \sum_{i=0}^n c_i x^i > 0$ para $c_i > 0$ e $x > 0$. Para determinar os limites superiores e inferiores das raízes positivas e negativas, são necessárias três equações auxiliares

$$P_1(x) = x^n P(1/x) = 0,$$

$$P_2(x) = P(-x) = 0 \text{ e}$$

$$P_3(x) = x^n P(-1/x) = 0.$$

Sendo ξ_i , $i = 1, 2, \dots, n$, as raízes de $P(x) = 0$, então $P(x)$ na forma fatorada é

$$P(x) = c_n(x - \xi_1)(x - \xi_2) \dots (x - \xi_n).$$

Desse modo,

$$P_1(x) = c_n x^n (1/x - \xi_1)(1/x - \xi_2) \dots (1/x - \xi_n),$$

$$P_1(x) = c_n (1 - x\xi_1)(1 - x\xi_2) \dots (1 - x\xi_n),$$

cujas raízes são $1/\xi_1, 1/\xi_2, \dots, 1/\xi_n$. Similarmente,

$$P_2(x) = c_n (-x - \xi_1)(-x - \xi_2) \dots (-x - \xi_n),$$

com raízes $-\xi_1, -\xi_2, \dots, -\xi_n$, e

$$P_3(x) = c_n x^n (-1/x - \xi_1)(-1/x - \xi_2) \dots (-1/x - \xi_n),$$

$$P_3(x) = c_n (-1 - x\xi_1)(-1 - x\xi_2) \dots (-1 - x\xi_n),$$

sendo as raízes $-1/\xi_1, -1/\xi_2, \dots, -1/\xi_n$.

Exemplo 6.10 Seja $P(x) = x^4 - 6x^3 - 5x^2 + 42x + 40 = 0$, com raízes $\xi_1 = -2$, $\xi_2 = -1$, $\xi_3 = 4$, $\xi_4 = 5$, então as equações auxiliares e suas respectivas raízes são

$$P_1(x) = x^4 P(1/x) = 40x^4 + 42x^3 - 5x^2 - 6x + 1,$$

$$(\xi_1 = -0,5; \xi_2 = -1, \xi_3 = 0,25; \xi_4 = 0,2),$$

$$P_2(x) = P(-x) = x^4 + 6x^3 - 5x^2 - 42x + 40,$$

$$(\xi_1 = 2, \xi_2 = 1, \xi_3 = -4, \xi_4 = -5),$$

$$P_3(x) = x^4 P(-1/x) = 40x^4 - 42x^3 - 5x^2 + 6x + 1,$$

$$(\xi_1 = 0,5; \xi_2 = 1, \xi_3 = -0,25; \xi_4 = -0,2). ■$$

Se $1/\xi_q$ for a maior das raízes positivas de $P_1(x) = 0$, então ξ_q será a menor das raízes positivas de $P(x) = 0$ (ver Exemplo 6.10). Sendo L_1 o limite superior das raízes positivas de $P_1(x) = 0$, calculado pelo Teorema 6.3, tem-se que

$$\frac{1}{\xi_q} \leq L_1 \rightarrow \xi_q \geq \frac{1}{L_1},$$

conseqüentemente, o limite inferior das raízes positivas de $P(x) = 0$ é $1/L_1$. Desse modo, se $P(x) = 0$ possuir raízes positivas ξ^+ , elas estarão no intervalo

$$\boxed{\frac{1}{L_1} \leq \xi^+ \leq L_1}.$$

Por outro lado, se $-\xi_r$ for a maior das raízes positivas de $P_2(x) = 0$, então ξ_r será a menor das raízes negativas de $P(x) = 0$ (ver Exemplo 6.10). Sendo L_2 o limite superior das raízes positivas de $P_2(x) = 0$, dado pelo Teorema 6.3

$$-\xi_r \leq L_2 \rightarrow \xi_r \geq -L_2.$$

Se $-1/\xi_s$ for a maior das raízes positivas de $P_3(x) = 0$, então ξ_s será a menor das raízes negativas de $P(x) = 0$ (ver Exemplo 6.10). Sendo L_3 o limite superior das raízes positivas de $P_3(x) = 0$, dado pelo Teorema 6.3

$$-\frac{1}{\xi_s} \leq L_3 \rightarrow \xi_s \leq -\frac{1}{L_3}.$$

Então, se $P(x) = 0$ tiver raízes negativas ξ^- , elas estarão no intervalo

$$\boxed{-L_2 \leq \xi^- \leq -\frac{1}{L_3}},$$

A Figura 6.3 mostra os limites das raízes reais de uma equação algébrica. É importante notar que esses limites não garantem a existência das raízes reais, mas tão-somente informam onde as raízes reais estarão caso existam.

Exemplo 6.11 Calcular os limites das raízes reais de $P(x) = x^4 + 2x^3 - 13x^2 - 14x + 24 = 0$ do Exemplo 6.9.

As equações auxiliares são

$$P_1(x) = x^4 P\left(\frac{1}{x}\right) = x^4 \left(\frac{1}{x^4} + \frac{2}{x^3} - \frac{13}{x^2} - \frac{14}{x} + 24\right) = 0 \rightarrow$$

$$P_1(x) = 24x^4 - 14x^3 - 13x^2 + 2x + 1 = 0,$$

$$L_1 = 1 + \sqrt[4]{\frac{14}{24}} \sim \frac{1}{L_1} = 0,63,$$

$$P_2(x) = P(-x) = (-x)^4 + 2(-x)^3 - 13(-x)^2 - 14(-x) + 24 = 0 \rightarrow$$

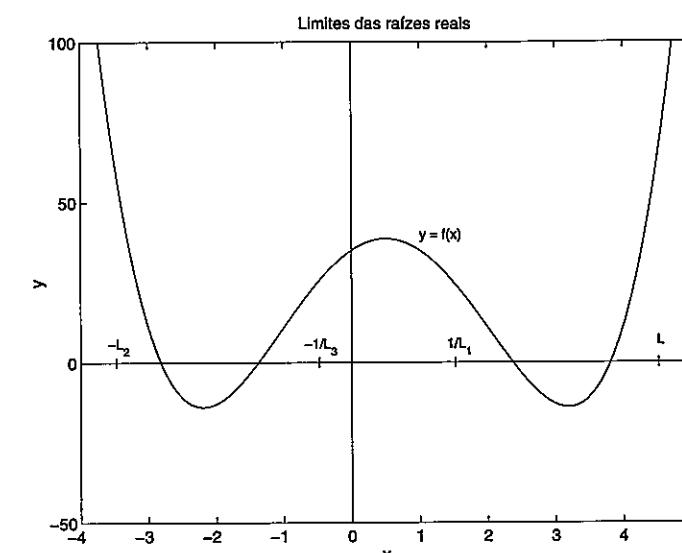


Figura 6.3 Limites das raízes reais de uma equação algébrica.

$$P_2(x) = x^4 - 2x^3 - 13x^2 + 14x + 24 = 0,$$

$$L_2 = 1 + \sqrt[4]{\frac{13}{1}} \sim -L_2 = -14 \text{ e}$$

$$P_3(x) = x^4 P\left(\frac{1}{-x}\right) = x^4 \left(\frac{1}{(-x)^4} + \frac{2}{(-x)^3} - \frac{13}{(-x)^2} - \frac{14}{(-x)} + 24\right) = 0 \rightarrow$$

$$P_3(x) = 24x^4 + 14x^3 - 13x^2 - 2x + 1 = 0,$$

$$L_3 = 1 + \sqrt[4]{\frac{13}{24}} \sim -\frac{1}{L_3} = -0,58.$$

Considerando que $L = 4,74$, conforme o Exemplo 6.9, então os limites das raízes reais são

$$0,63 \leq \xi^+ \leq 4,74 \text{ e } -14 \leq \xi^- \leq -0,58.$$

Pode ser construído um dispositivo prático para facilitar a determinação dos limites das raízes reais. O dispositivo é constituído de dois blocos. No primeiro bloco, são definidos os coeficientes de $P(x) = 0$ e de suas três equações auxiliares $P_1(x) = 0$, $P_2(x) = 0$ e $P_3(x) = 0$. Para tal,

1. colocar os coeficientes de $P(x) = 0$ na coluna $P(x)$, com c_n no topo,
2. inverter a ordem dos coeficientes da coluna $P(x)$ e colocá-los em $P_1(x)$,
3. trocar o sinal dos coeficientes de $P(x)$, cujos índices sejam ímpares e atribuí-los a $P_2(x)$,

4. inverter a ordem dos coeficientes da coluna $P_2(x)$ e colocá-los em $P_3(x)$ e
5. se algum $c_n < 0$, então trocar o sinal de todos os coeficientes da coluna para garantir que $c_n > 0$, conforme exigência do Teorema 6.3.

No segundo bloco, são atribuídos os parâmetros necessários para aplicar o Teorema 6.3 a cada uma das quatro equações. Assim,

- k é o índice do primeiro coeficiente negativo,
- n é o grau do polinômio,
- B é o valor absoluto do maior coeficiente negativo em módulo,
- L_i é o limite superior das raízes positivas de $P_i(x) = 0$ dado pelo Teorema 6.3 e
- L_ξ são os limites superiores e inferiores das raízes positivas e negativas de $P(x) = 0$, sendo que $L_{\xi(P)} = L$, $L_{\xi(P_1)} = 1/L_1$, $L_{\xi(P_2)} = -L_2$ e $L_{\xi(P_3)} = -1/L_3$.

Exemplo 6.12 Calcular os limites das raízes de $P(x) = x^4 + 2x^3 - 13x^2 - 14x + 24 = 0$ do Exemplo 6.11 usando o dispositivo prático.

| $n=4$ | $P(x)$ | $P_1(x)$ | $P_2(x)$ | $P_3(x)$ |
|---------|-----------|-----------|-----------|-----------|
| c_4 | 1 | 24 | 1 | 24 |
| c_3 | 2 | -14 | -2 | 14 |
| c_2 | -13 | -13 | -13 | -13 |
| c_1 | -14 | 2 | 14 | -2 |
| c_0 | 24 | 1 | 24 | 1 |
| k | 2 | 3 | 3 | 2 |
| $n-k$ | 2 | 1 | 1 | 2 |
| B | $ -14 $ | $ -14 $ | $ -13 $ | $ -13 $ |
| L_i | 4,74 | 1,58 | 14 | 1,74 |
| L_ξ | 4,74 | 0,63 | -14 | -0,58 |

O algoritmo mostrado na Figura 6.4, baseado no dispositivo prático, determina os limites das raízes reais de uma equação polinomial a partir dos seus coeficientes. Os parâmetros de entrada são o grau n do polinômio e o vetor c dos coeficientes, e $P(x) = c(1)x^n + c(2)x^{n-1} + \dots + c(n)x + c(n+1)$. O parâmetro de saída é o vetor L contendo os limites das raízes reais, de forma que $L(1) = 1/L_1$, $L(2) = L$, $L(3) = -L_2$ e $L(4) = -1/L_3$.

Exemplo 6.13 Calcular os limites das raízes reais da equação polinomial $P(x) = x^4 + 2x^3 - 13x^2 - 14x + 24 = 0$ do Exemplo 6.11 usando o algoritmo da Figura 6.4.

Algoritmo LimitesRaízes

```

{ Objetivo: Achar os limites das raízes reais de uma equação polinomial }
parâmetros de entrada n, c { grau do polinômio e coeficientes, sendo }
{ P(x) = c(1)x^n + c(2)x^{n-1} + ... + c(n)x + c(n+1) }
parâmetros de saída L
{ limites inferior e superior das raízes positivas e negativas, respectivamente }
se c(1) = 0 então escreva "coeficiente c(1) nulo", abandone, fimse
t ← n+1; c(t+1) ← 0
repita { se c(n+1) for nulo, então o polinômio é defacionado }
  se c(t) ≠ 0 então interrompa, fimse; t ← t-1
fimrepita
{ cálculo dos quatro limites das raízes reais }
para i ← 1 até 4 faça
  se i = 2 ou i = 4 então { inversão da ordem dos coeficientes }
    para j ← 1 até t/2 faça
      Aux ← c(j); c(j) ← c(t-j+1); c(t-j+1) ← Aux
    fimpara
  senão
    se i = 3 então
      { reinversão da ordem e troca de sinais dos coeficientes }
      para j ← 1 até t/2 faça
        Aux ← c(j); c(j) ← c(t-j+1); c(t-j+1) ← Aux
      fimpara
      para j ← t-1 até 1 passo -2 faça c(j) ← -c(j), fimpara
    fimse
  fimse
  { se c(1) for negativo, então é trocado o sinal de todos os coeficientes }
  se c(1) < 0 então
    para j ← 1 até t faça c(j) ← -c(j), fimpara
  fimse
  k ← 2 { cálculo de k, o maior índice dos coeficientes negativos }
  repita
    se c(k) < 0 ou k > t então interrompa, fimse
    k ← k+1
  fimrepita { cálculo de B, o maior coeficiente negativo em módulo }
  se k ≤ t então
    B ← 0
    para j ← 2 até t faça
      se c(j) < 0 e abs(c(j)) > B então B ← abs(c(j)), fimse
    fimpara
    { limite das raízes positivas de P(x) = 0 e das equações auxiliares }
    L(1) ← 1 + k-1/B/G(1)
  senão
    L(1) ← 10100
  fimse
  { limites das raízes positivas e negativas de P(x) = 0 }
  Aux ← L(1); L(1) ← 1/L(2); L(2) ← Aux; L(3) ← -L(3); L(4) ← -1/L(4)
fimalgoritmo

```

Figura 6.4 Algoritmo para achar os limites das raízes de equação polinomial.

(Ver significado da função abs na Tabela 1.1, na página 6.)

```
% Os parametros de entrada
n = 4
c = 1 2 -13 -14 24
% produzem os resultados
L = 0.6316 4.7417 -14.0000 -0.5760
```

Pelo gráfico mostrado na Figura 6.5(a), observa-se que o intervalo original pode ser reduzido a $[-5, 4]$. A Figura 6.5(b) revela claramente as quatro raízes isoladas nos intervalos $[-5, -3]$, $[-3, -1]$, $[0, 2]$ e $[2, 4]$.

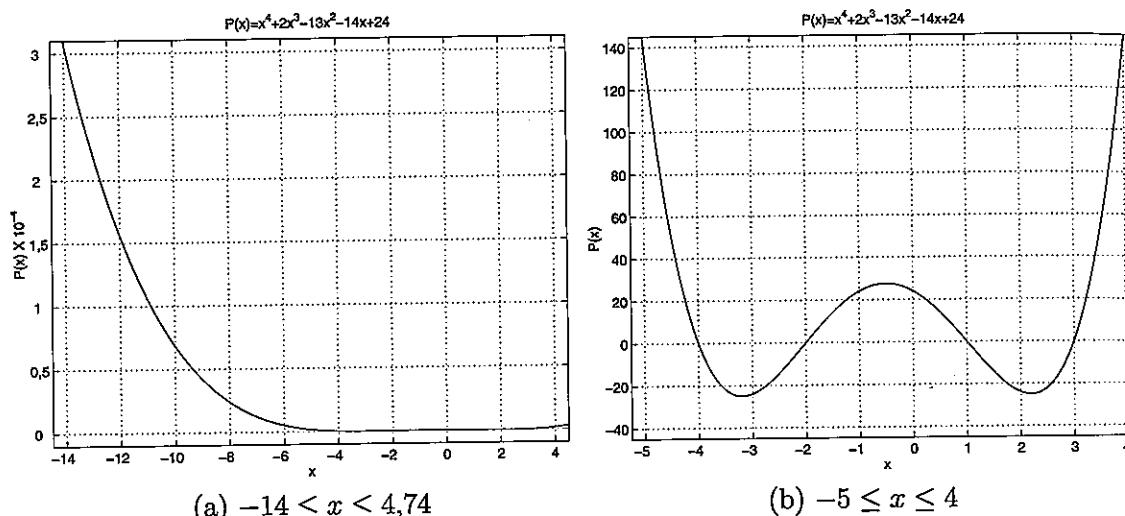


Figura 6.5 Gráficos do polinômio $P(x) = x^4 + 2x^3 - 13x^2 - 14x + 24$.

Número de raízes reais

A Regra de sinais de Descartes [10] é uma maneira simples para determinar o número de raízes reais de uma equação algébrica. Sabendo-se os limites e o número das raízes reais, a tarefa de isolar essas raízes ficará grandemente facilitada.

Teorema 6.4 (Regra de sinais de Descartes) O número de raízes reais positivas n^+ de $P(x) = 0$ é igual ao número de variações de sinais na sequência dos coeficientes ou é menor que este número por um inteiro par, sendo as raízes contadas de acordo com a sua multiplicidade e não sendo considerados os coeficientes nulos.

Corolário 6.2 Se $P(x) = 0$ não possuir coeficientes nulos, então o número de raízes reais negativas n^- (contando multiplicidades) é igual ao número de permanências de sinais na sequência dos coeficientes ou é menor que este número por um inteiro par.

A Regra de sinais de Descartes consegue discernir entre as raízes positivas e negativas; no entanto, não consegue separar as raízes reais das complexas, as quais aparecem em pares conjugados, conforme o Teorema 6.2. Por exemplo, se o número de variações de sinais for 5, então $n^+ = 5$ ou 3 ou 1.

Exemplo 6.14 Para $P(x) = x^4 + 2x^3 - 13x^2 - 14x + 24 = 0$, tem-se que

$$n^+ = 2 \text{ ou } 0, \text{ e } n^- = 2 \text{ ou } 0.$$

Para essa equação, se existirem duas raízes positivas, elas satisfarão a $0,63 \leq \xi^+ \leq 4,74$ e, se houver duas negativas, elas estarão no intervalo $-14 \leq \xi^- \leq -0,58$ (ver Exemplo 6.11 e Figura 6.5(b)). ■

Portanto, combinando a Regra de sinais de Descartes e o Teorema de Lagrange, conseguem-se importantes informações para o pleno isolamento das raízes, como será visto mais adiante.

Exemplo 6.15 Calcular os limites e o número de raízes reais de $P(x) = x^3 - 3x^2 - 6x + 8 = 0$.

Para aplicar o Teorema 6.3 e obter os L_i 's, o coeficiente $c_n = c_3$ deve ser positivo em cada um dos quatro polinômios. Como isso não é verdadeiro para $P_2(x)$, então todos os coeficientes desse polinômio devem ter os seus sinais trocados. Os zeros do novo polinômio serão iguais aos do polinômio original e, por conseguinte, os dois polinômios terão o mesmo L_2 .

| $n=3$ | $P(x)$ | $P_1(x)$ | $P_2(x)$ | $P_3(x)$ |
|---------|--------|----------|----------|----------|
| c_3 | 1 | 8 | -1 | 8 |
| c_2 | -3 | -6 | -3 | 6 |
| c_1 | -6 | -3 | 6 | -3 |
| c_0 | 8 | 1 | 8 | -1 |
| k | | | | |
| $n-k$ | | | | |
| B | | | | |
| L_i | | | | |
| L_ξ | | | | |

| $n=3$ | $P(x)$ | $P_1(x)$ | $P_2(x)$ | $P_3(x)$ |
|---------|--------|----------|----------|----------|
| c_3 | 1 | 8 | 1 | 8 |
| c_2 | -3 | -6 | 3 | 6 |
| c_1 | -6 | -3 | -6 | -3 |
| c_0 | 8 | 1 | -8 | -1 |
| k | 2 | 2 | 1 | 1 |
| $n-k$ | 1 | 1 | 2 | 2 |
| B | -6 | -6 | -8 | -3 |
| L_i | 7 | 1,75 | 3,83 | 1,61 |
| L_ξ | 7 | 0,57 | -3,83 | -0,62 |

Trocar sinal
de $P_2(x)$

Os limites das raízes são $0,57 \leq \xi^+ \leq 7$ e $-3,83 \leq \xi^- \leq -0,62$ e o número de raízes reais são $n^+ = 2$ ou 0 e $n^- = 1$, significando que existe uma raiz real negativa e as outras duas serão reais positivas ou complexas. ■

Exemplo 6.16 Achar os limites e o número de raízes reais de $P(x) = x^6 - 5x^5 + 7x^4 + 19x^3 - 98x^2 - 104x = 0$.

Como $c_0 = 0$, tem-se, pelas relações de Girard, que existe pelo menos uma raiz nula. Para obter as equações auxiliares, $P(x)$ deve ser deflacionado pela divisão por $(x - 0)$, resultando em $P(x) = x^5 - 5x^4 + 7x^3 + 19x^2 - 98x - 104 = 0$. Este novo $P(x)$ terá todas as raízes do antigo, com excessão da raiz nula. Neste exemplo, como $c_5 < 0$ para $P_i(x)$, $i = 1, 2, 3$, os sinais de todos os coeficientes de $P_1(x)$, $P_2(x)$ e $P_3(x)$ devem ser trocados para que se possa aplicar o Teorema 6.3.

| $n=5$ | $P(x)$ | $P_1(x)$ | $P_2(x)$ | $P_3(x)$ |
|---------|--------|----------|----------|----------|
| c_5 | 1 | -104 | -1 | -104 |
| c_4 | -5 | -98 | -5 | 98 |
| c_3 | 7 | 19 | -7 | 19 |
| c_2 | 19 | 7 | 19 | -7 |
| c_1 | -98 | -5 | 98 | -5 |
| c_0 | -104 | 1 | -104 | -1 |
| k | | | | |
| $n-k$ | | | | |
| B | | | | |
| L_i | | | | |
| L_ξ | | | | |

Trocar sinal de
 $P_1(x), P_2(x), P_3(x)$

| $n=5$ | $P(x)$ | $P_1(x)$ | $P_2(x)$ | $P_3(x)$ |
|---------|--------|----------|----------|----------|
| c_5 | 1 | 104 | 1 | 104 |
| c_4 | -5 | 98 | 5 | -98 |
| c_3 | 7 | -19 | 7 | -19 |
| c_2 | 19 | -7 | -19 | 7 |
| c_1 | -98 | 5 | -98 | 5 |
| c_0 | -104 | -1 | 104 | 1 |
| k | 4 | 3 | 2 | 4 |
| $n-k$ | 1 | 2 | 3 | 1 |
| B | -104 | -19 | -98 | -98 |
| L_i | 105 | 1,43 | 5,61 | 1,94 |
| L_ξ | 105 | 0,70 | -5,61 | -0,51 |

Assim, os limites das raízes são $0,70 \leq \xi^+ \leq 105$ e $-5,61 \leq \xi^- \leq -0,51$ e o número de raízes reais são $n^+ = 3$ ou 1 e $n^- = 2$ ou 0, garantindo pelo menos uma raiz real positiva. ■

6.1.2 Equações transcendentas

Infelizmente, as equações transcendentas não dispõem de teoremas, como visto para as algébricas, que forneçam informações sobre os limites e o número de raízes reais. Se as algébricas têm um número finito de raízes, o mesmo, às vezes, não ocorre com uma transcendente que pode ter um número infinito de raízes, como, por exemplo, $f(x) = \operatorname{sen}(x) = 0$ ou mesmo não ter raízes como $f(x) = \operatorname{sen}(x) - 2 = 0$.

O método gráfico é a maneira mais simples para achar um intervalo que contenha uma única raiz. Ele consiste em fazer um esboço da função no intervalo de interesse. No entanto, a maior dificuldade com as equações transcendentas consiste, justamente, em determinar este intervalo, já que não existe um “Teorema de Lagrange” para elas. O que se faz na prática é usar a intuição, o conhecimento a respeito da função e, como ocorre na maioria das vezes, o método da tentativa e erro, ou seja, a partir de um intervalo inicial, tenta-se um outro que isole a raiz observando-se a forma da curva.

O algoritmo descrito na Figura 6.6 fornece um intervalo $[a, b]$, no qual uma função $f(x)$ troca de sinal, ou seja, $f(a)f(b) < 0$. Este intervalo é expandido a partir de um dado valor z . Apesar de a raiz não estar necessariamente isolada, pois pode haver um número ímpar de raízes no intervalo, o algoritmo fornecerá um intervalo de partida para o isolamento de uma raiz.

O parâmetro de entrada é o ponto z a partir do qual o intervalo será gerado. Os parâmetros de saída são o limite inferior a e superior b do intervalo procurado e a condição de erro $\operatorname{CondErro}$. Se $\operatorname{CondErro} = 0$ significa que $f(a)f(b) \leq 0$, ou seja, existe um número ímpar de raízes no intervalo $[a, b]$. Mas, se $\operatorname{CondErro} = 1$, então $f(a)f(b) > 0$, significando que o intervalo $[a, b]$ não tem raízes ou tem um número par de raízes. A função $f(x)$ deve ser especificada em cada caso. No entanto, a maneira de fazer essa especificação vai depender da linguagem de programação utilizada para implementar o algoritmo **TrocaSinal**.

```

Algoritmo TrocaSinal
{ Objetivo: Achar um intervalo  $[a, b]$  onde uma função troca de sinal }
parâmetros de entrada z
{ ponto a partir do qual o intervalo será gerado }
parâmetros de saída a, b, CondErro
{ limite inferior e superior do intervalo e condição de erro, sendo }
{ CondErro = 0 se  $f(a)f(b) \leq 0$  e CondErro = 1 se  $f(a)f(b) > 0$ . }
se z = 0 então
  a ← -0,05; b ← 0,05
senão
  a ← 0,95 * z; b ← 1,05 * z
fimse
iter ← 0; Aureo ← 2 / (raiz2(5) - 1)
Fa ← f(a); Fb ← f(b) { avaliar a função em a e b }
escreva iter, a, b, Fa, Fb
repita
  se Fa * Fb ≤ 0 ou iter ≥ 20 então interrompa, fimse
  iter ← iter + 1
  se abs(Fa) < abs(Fb) então
    a ← a - Aureo * (b - a)
    Fa ← f(a) { avaliar a função em a }
  senão
    b ← b + Aureo * (b - a)
    Fb ← f(b) { avaliar a função em b }
  fimse
  escreva iter, a, b, Fa, Fb
fimrepita
se Fa * Fb ≤ 0 então
  CondErro ← 0
senão
  CondErro ← 1
fimse
finalgoritmo

```

Figura 6.6 Algoritmo para achar um intervalo onde uma função troca de sinal.

(Ver significado das funções abs e raiz₂ na Tabela 1.1, na página 6.)

Exemplo 6.17 Achar um intervalo, a partir de $z = 5$, onde $f(x) = 2x^3 - \cos(x+1) - 3$ troca de sinal, utilizando o algoritmo da Figura 6.6.

```

% 0 parâmetro de entrada
z = 5
% produz os resultados

```

Determinação de intervalo onde ocorre troca de sinal

| iter | a | b | Fa | Fb |
|------|---------|--------|-----------|----------|
| 0 | 4.7500 | 5.2500 | 210.4826 | 285.4068 |
| 1 | 3.9410 | 5.2500 | 119.1909 | 285.4068 |
| 2 | 1.8229 | 5.2500 | 10.0655 | 285.4068 |
| 3 | -3.7221 | 5.2500 | -105.2218 | 285.4068 |

a = -3.7221
b = 5.2500
CondErro = 0

De fato, a função muda de sinal no intervalo $[-3.7221; 5.2500]$, pois $f(-3.7221) = -105.2218$ e $f(5.2500) = 285.4068$. Pelo gráfico mostrado na Figura 6.7(a), percebe-se que o intervalo $[-1, 2]$ contém uma única raiz. A curva pode ser esboçada neste intervalo menor, Figura 6.7(b), para examinar o gráfico com uma melhor definição.

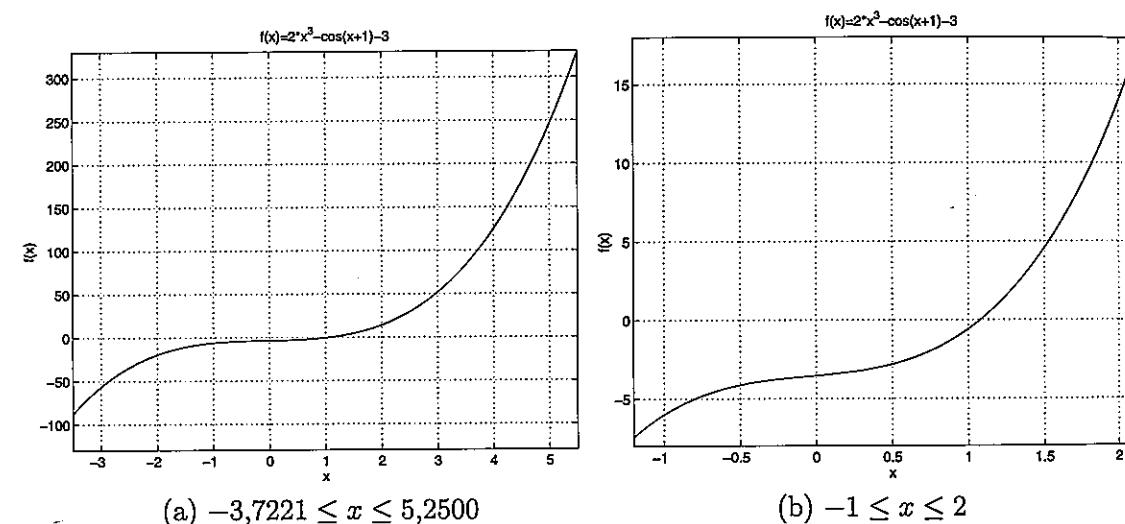


Figura 6.7 Esboços da função $f(x) = 2x^3 - \cos(x+1) - 3$ em dois intervalos.

Exemplo 6.18 Isolar, graficamente, os zeros da função $f(x) = 0,05x^3 - 0,4x^2 + 3 \sin(x)x$.

Inicialmente, pode-se definir um intervalo amplo para ter uma visão do comportamento da função. A Figura 6.8(a) apresenta um esboço para $-20 \leq x \leq 20$. Um melhor detalhamento pode ser obtido usando $-4 \leq x \leq 12$, o qual é mostrado na Figura 6.8(b). Podem ser observadas seis raízes no intervalo $[-4, 12]$ e cada uma destas raízes estará isolada nos seguintes intervalos: $[-4, -2], [-1, 1], [2, 4], [6, 8], [8, 10]$ e $[10, 12]$.

O método gráfico também pode ser usado para isolar as raízes de equações algébricas. Conhecendo os limites e o número de raízes reais pelo Teorema de Lagrange e pela Regra de sinais de Descartes, o isolamento por gráfico fica enormemente facilitado.

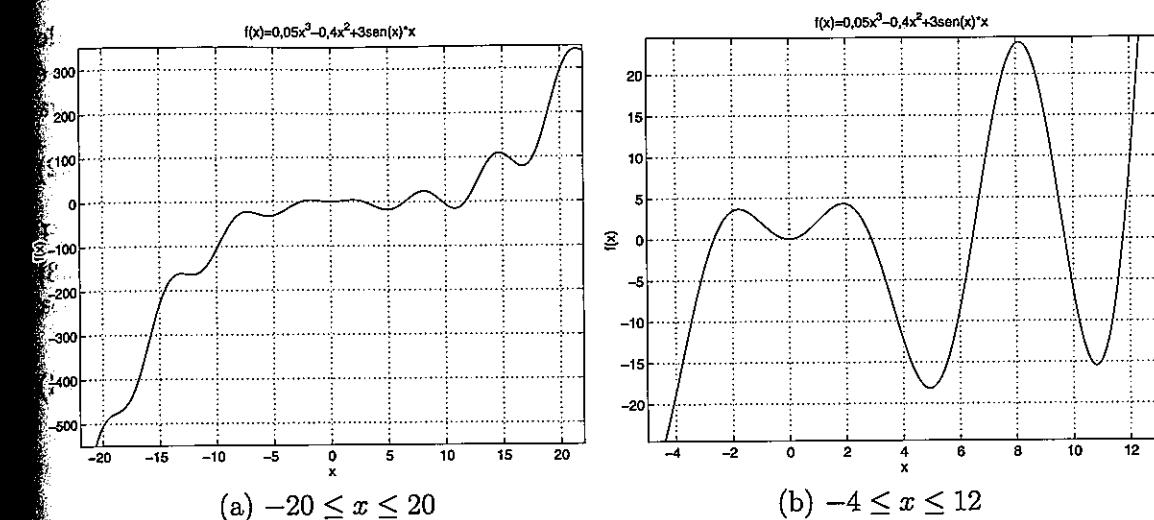


Figura 6.8 Esboços da função $f(x) = 0,05x^3 - 0,4x^2 + 3 \sin(x)x$.

6.1.3 Convergência da raiz

Se uma raiz ξ já estiver isolada em um dado intervalo $[a, b]$, então a próxima etapa consiste em gerar uma seqüência $\{x_0, x_1, x_2, \dots, x_k, \dots, \xi\} \in [a, b]$ que convinha para esta raiz exata ξ de $f(x) = 0$.

Critério de parada

Antes de serem abordados os métodos para produzirem esta seqüência, é necessário definir um critério de parada, ou seja, quando se deve interromper a geração da seqüência. A demonstração do teorema abaixo é dada por Demidovich e Maron [10].

Teorema 6.5 Sejam ξ uma raiz exata e x_k uma raiz aproximada de $f(x) = 0$, sendo ξ e $x_k \in [a, b]$ e $|f'(x)| \geq m > 0$ para $a \leq x \leq b$, com

$$m = \min_{a \leq x \leq b} |f'(x)|.$$

Então, o erro absoluto satisfaz

$$|x_k - \xi| \leq \frac{|f(x_k)|}{m}.$$

Exemplo 6.19 Avaliar o erro absoluto cometido ao considerar $x_k = 2,23$ como aproximação da raiz positiva de $f(x) = x^2 - 5 = 0$ no intervalo $[2, 3]$.

$$m = \min_{2 \leq x \leq 3} |2x| = 4.$$

Assim,

$$|2,23 - \xi| \leq \frac{0,0271}{4} = 0,0068 \rightarrow$$

$$2,23 - 0,0068 \leq \xi \leq 2,23 + 0,0068 \quad (\xi = \sqrt{5} \approx 2,2361).$$

O Teorema 6.5 é de aplicação muito restrita, haja vista que ele requer que seja avaliado o mínimo da derivada primeira da função $f(x)$ em questão. Na prática, a seqüência é interrompida quando seus valores satisfizerem a pelo menos um dos critérios

$$|x_k - x_{k-1}| \leq \varepsilon, \quad (6.2)$$

$$\left| \frac{x_k - x_{k-1}}{x_k} \right| \leq \varepsilon, \quad (6.3)$$

$$|f(x_k)| \leq \varepsilon \quad (6.4)$$

onde ε é a tolerância fornecida.

Ordem de convergência

Uma questão fundamental é definir a rapidez com que a seqüência gerada por um dado método, $\{x_0, x_1, x_2, \dots, x_k, \dots\}$, converge para a raiz exata ξ . Para tal, seja o erro da k -ésima iteração definido por

$$\epsilon_k = x_k - \xi \quad (6.5)$$

ou seja, a diferença entre a raiz ξ e a sua estimativa x_k . Um critério para avaliar a convergência é

$$\lim_{k \rightarrow \infty} |\epsilon_{k+1}| = K |\epsilon_k|^\gamma \quad (6.6)$$

onde K é a constante de erro assintótico e γ é a ordem de convergência do método gerador da seqüência. Por causa disso, quanto maior for o valor de γ , mais rápida a seqüência convergirá para a raiz da equação.

6.2 Método da bisseção

Seja uma função $f(x)$ contínua no intervalo $[a, b]$, sendo ξ a única raiz de $f(x) = 0$ neste intervalo. O método da bisseção consiste, simplesmente, em subdividir o intervalo ao meio a cada iteração e manter o subintervalo que contenha a raiz, ou seja, aquele em que $f(x)$ tenha sinais opostos nos extremos. Deste modo, obtém-se uma seqüência de intervalos encaixados $\{[a_1, b_1], [a_2, b_2], [a_3, b_3], \dots, [a_k, b_k]\}$ nos quais

$$f(a_i)f(b_i) < 0, \quad i = 1, 2, \dots, k. \quad (6.7)$$

Na k -ésima iteração, tem-se

$$b_k - a_k = \frac{b - a}{2^k}. \quad (6.8)$$

A seqüência $\{a_1, a_2, a_3, \dots, a_k\}$ é monotônica não decrescente limitada, enquanto $\{b_1, b_2, b_3, \dots, b_k\}$ é monotônica não crescente limitada. Então, por (6.8), essas duas seqüências possuem um limite comum ξ

$$\lim_{k \rightarrow \infty} a_k = \lim_{k \rightarrow \infty} b_k = \xi.$$

Passando ao limite da desigualdade (6.7) com $k \rightarrow \infty$ e considerando que $f(x)$ é contínua, então $|f(\xi)|^2 \leq 0 \rightarrow f(\xi) = 0$, ou seja, ξ é uma raiz de $f(x) = 0$. Os primeiros pontos da seqüência $\{x_1, x_2, x_3, \dots, x_k, \dots\}$ são mostrados na Figura 6.9.

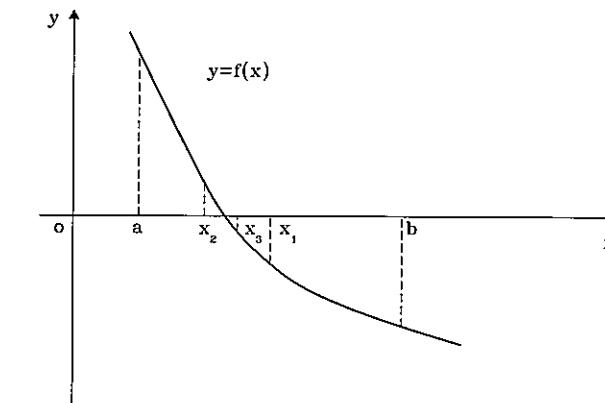


Figura 6.9 Interpretação gráfica do método da bisseção.

Uma das grandes vantagens do método da bisseção é que ele tem convergência garantida se $f(x)$ for contínua em $[a, b]$ e se $\xi \in [a, b]$. Ele é também um dos poucos métodos nos quais é possível determinar *a priori* o número de iterações necessárias para calcular a raiz com uma tolerância ε a partir de um intervalo $[a, b]$. Para tanto, substituindo $x_k = (b_k - a_k)/2$ em (6.8), tem-se que

$$|x_k - x_{k-1}| = \frac{b - a}{2^{k+1}}.$$

Utilizando o critério (6.2),

$$\frac{b - a}{2^{k+1}} \leq \varepsilon,$$

então o número de iterações para calcular uma raiz no intervalo $[a, b]$ com tolerância ε é

$$k \geq \log_2 \left(\frac{b - a}{\varepsilon} \right) - 1. \quad (6.9)$$

O algoritmo mostrado na Figura 6.10 calcula, pelo método da bisseção, a raiz de uma equação $f(x) = 0$ contida no intervalo $[a, b]$, com tolerância ε usando os critérios (6.2) e

(6.4). Os parâmetros de entrada são o limite inferior a e o superior b do intervalo que contém a raiz, a tolerância $Toler$ para o cálculo da raiz e o número máximo de iterações $IterMax$. A função $f(x)$ deve ser especificada de acordo com a linguagem de programação utilizada. Os parâmetros de saída são a raiz da equação $Raiz$, o número de iterações gastos $Iter$ e a condição de erro $CondErro$. Se $CondErro = 0$ significa que a raiz foi encontrada e $CondErro = 1$ indica que a raiz não foi encontrada com a tolerância e o número máximo de iterações fornecidos.

Exemplo 6.20 Calcular a raiz de $f(x) = 2x^3 - \cos(x+1) - 3 = 0$ do Exemplo 6.17, que está no intervalo $[-1, 2]$, com $\varepsilon \leq 0,01$ pelo algoritmo da Figura 6.10.

```
% Os parametros de entrada
a = -1
b = 2
Toler = 0.0100
IterMax = 100
% produzem os resultados
    Calculo de raiz de equacao pelo metodo da bissecao
iter   a       Fa      b       Fb      x       Fx      Delta_x
  0  -1.00000 -6.00000  2.00000 13.98999  0.50000 -2.8207e+00  1.50000
  1   0.50000 -2.82074  2.00000 13.98999  1.25000  1.5344e+00  0.75000
  2   0.50000 -2.82074  1.25000 13.98999  0.87500 -1.3606e+00  0.37500
  3   0.87500 -1.36062  1.25000 13.98999  1.06250 -1.2895e-01  0.18750
  4   1.06250 -0.12895  1.25000 13.98999  1.15625  6.4419e-01  0.09375
  5   1.06250 -0.12895  1.15625 13.98999  1.10938  2.4356e-01  0.04688
  6   1.06250 -0.12895  1.10938 13.98999  1.08594  5.3864e-02  0.02344
  7   1.06250 -0.12895  1.08594 13.98999  1.07422 -3.8393e-02  0.01172
  8   1.07422 -0.03839  1.08594 13.98999  1.08008  7.5211e-03  0.00586

Raiz      = 1.08008
Iter      = 8
CondErro = 0
```

A raiz é $\xi \approx x_8 = 1,08008$. Por (6.9), o número de iterações $k \geq \log((2 - (-1))/0,01) - 1 \approx 7,23$, ou seja, para alcançar o critério de parada (6.2) $|x_k - x_{k-1}| \leq \varepsilon$ (ver coluna $Delta_x$) foram necessárias 8 iterações. Poderiam ser gastos mais iterações para atender ao outro critério de parada (6.4) $|f(x_k)| \leq \varepsilon$.

Exemplo 6.21 Determinar a maior raiz de $f(x) = 0,05x^3 - 0,4x^2 + 3 \sin(x)x = 0$ com $\varepsilon \leq 0,005$, usando o algoritmo da Figura 6.10.

Pela Figura 6.8(b) do Exemplo 6.18, tem-se que $\xi \in [10, 12]$.

```
% Os parametros de entrada
a = 10
b = 12
Toler = 0.0050
IterMax = 100
% produzem os resultados
```

```
Algoritmo Bisseção
{ Objetivo: Calcular a raiz de uma equação pelo método da bisseção }
parâmetros de entrada a, b, Toler, IterMax
{ limite inferior, limite superior, tolerância e número máximo de iterações }
parâmetros de saída Raiz, Iter, CondErro
{ raiz, número de iterações gastos e condição de erro, sendo }
{ CondErro = 0 se a raiz foi encontrada e CondErro = 1 em caso contrário. }
Fa ← f(a); Fb ← f(b) { avaliar a função em a e b }
se Fa * Fb > 0 então
    escreva "função não muda de sinal nos extremos do intervalo dado"
    abandone
fimse
DeltaX ← abs(b - a)/2; Iter ← 0
repita
    x ← (a + b)/2; Fx ← f(x) { avaliar a função em x }
    escreva Iter, a, Fa, b, Fb, x, Fx, DeltaX
    se (DeltaX ≤ Toler e abs(Fx) ≤ Toler) ou Iter ≥ IterMax então
        interrompa
    fimse
    se Fa * Fx > 0 então
        a ← x; Fa ← Fx
    senão
        b ← x
    fimse
    DeltaX ← DeltaX/2; Iter ← Iter + 1
fimrepita
Raiz ← x
{ teste de convergência }
se DeltaX ≤ Toler e abs(Fx) ≤ Toler então
    CondErro ← 0
senão
    CondErro ← 1
fimse
fimalgoritmo
```

Figura 6.10 Algoritmo do método da bisseção.

(Ver significado da função abs na Tabela 1.1, na página 6.)

| Calculo de raiz de equacao pelo metodo da bissecao | | | | | | | |
|--|----------|-----------|----------|---------|----------|-------------|---------|
| iter | a | Fa | b | Fb | x | Fx | Delta_x |
| 0 | 10.00000 | -6.32063 | 12.00000 | 9.48337 | 11.00000 | -1.4850e+01 | 1.00000 |
| 1 | 11.00000 | -14.84968 | 12.00000 | 9.48337 | 11.50000 | -7.0594e+00 | 0.50000 |
| 2 | 11.50000 | -7.05935 | 12.00000 | 9.48337 | 11.75000 | 2.0128e-01 | 0.25000 |
| 3 | 11.50000 | -7.05935 | 11.75000 | 9.48337 | 11.62500 | -3.6975e+00 | 0.12500 |

| | | | | | | | |
|----|----------|----------|----------|---------|----------|-------------|---------|
| 4 | 11.62500 | -3.69752 | 11.75000 | 9.48337 | 11.68750 | -1.8136e+00 | 0.06250 |
| 5 | 11.68750 | -1.81359 | 11.75000 | 9.48337 | 11.71875 | -8.2229e-01 | 0.03125 |
| 6 | 11.71875 | -0.82229 | 11.75000 | 9.48337 | 11.73438 | -3.1451e-01 | 0.01562 |
| 7 | 11.73438 | -0.31451 | 11.75000 | 9.48337 | 11.74219 | -5.7611e-02 | 0.00781 |
| 8 | 11.74219 | -0.05761 | 11.75000 | 9.48337 | 11.74609 | 7.1585e-02 | 0.00391 |
| 9 | 11.74219 | -0.05761 | 11.74609 | 9.48337 | 11.74414 | 6.9247e-03 | 0.00195 |
| 10 | 11.74219 | -0.05761 | 11.74414 | 9.48337 | 11.74316 | -2.5359e-02 | 0.00098 |
| 11 | 11.74316 | -0.02536 | 11.74414 | 9.48337 | 11.74365 | -9.2209e-03 | 0.00049 |
| 12 | 11.74365 | -0.00922 | 11.74414 | 9.48337 | 11.74390 | -1.1491e-03 | 0.00024 |

Raiz = 11.74390
 Iter = 12
 CondErro = 0

A raiz da equação é $\xi \approx x_{12} = 11.74390$.

Apesar de o método da bissecção ser robusto, ele não é eficiente devido à sua convergência lenta. Pode ser observado que $f(x)$ não decresce, monotonicamente, como no último exemplo, no qual $|f(x_3)| > |f(x_2)|$. Isso ocorre porque somente o sinal de $f(x_{k-1})$ é usado para o cálculo do próximo x_k , sem levar em consideração o seu valor.

O método da bissecção é mais usado para reduzir o intervalo antes de usar um outro método de convergência mais rápida, dos quais alguns serão vistos a seguir.

6.3 Métodos baseados em aproximação linear

A velocidade de convergência da seqüência $\{x_i\}$ para a raiz ξ de uma equação $f(x) = 0$ pode ser aumentada usando-se um esquema diferente da bissecção. Um esquema consiste em aproximar $f(x)$ por um polinômio linear no intervalo de interesse $[x_0, x_1]$. Se o intervalo for pequeno, essa aproximação é válida para a maioria das funções. Deste modo, uma estimativa da raiz ξ é tomada como o valor onde a reta cruza o eixo das abscissas.

A equação do polinômio de grau 1 que passa pelos pontos de coordenadas $[x_0, f(x_0)]$ e $[x_1, f(x_1)]$ é dada por (3.1), conforme mostrado no Capítulo 3,

$$y = f(x_1) + \frac{f(x_1) - f(x_0)}{x_1 - x_0}(x - x_1).$$

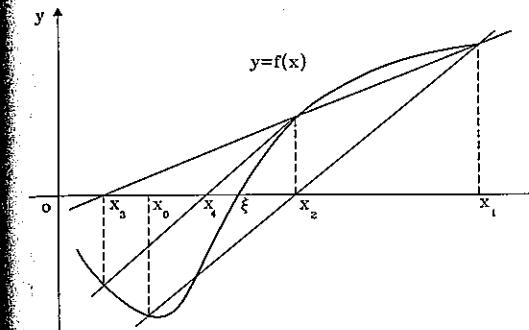
Então o valor da abscissa x_2 , para o qual $y = 0$, é tomado como uma aproximação da raiz

$$x_2 = x_1 - \frac{f(x_1)}{f(x_1) - f(x_0)}(x_1 - x_0).$$

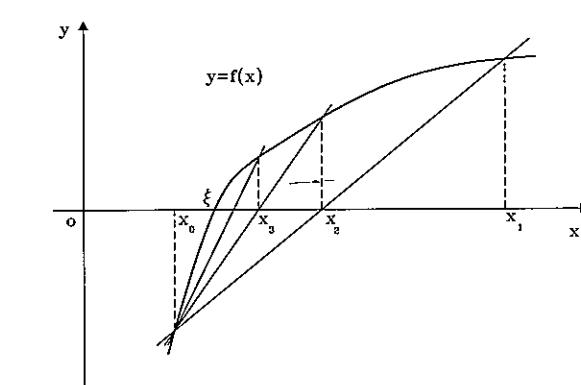
Na próxima iteração, um dos pontos extremos do intervalo $[x_0, x_1]$ será substituído por x_2 , que é uma melhor estimativa da raiz ξ . O processo se repete, gerando, desta forma, uma seqüência $\{x_i\}$ até que o critério de parada seja satisfeito. Existe uma família de métodos baseados nesta aproximação linear, dentre os quais o método da secante, o da *regula falsa* (posição falsa) e o pégaso.

6.3.1 Método da secante

O método da secante usa os pontos obtidos nas duas últimas iterações como pontos-base por onde passará o polinômio linear, conforme mostra a Figura 6.11(a).



(a) Método da secante



(b) Método da *regula falsa*

Figura 6.11 Métodos para cálculo da raiz baseados em aproximação linear.

O algoritmo da Figura 6.12 determina, pelo método da secante, a raiz de uma equação $f(x) = 0$ contida em um intervalo $[a, b]$, com tolerância ε usando os critérios (6.2) e (6.4). Os parâmetros de entrada são o limite inferior a e o superior b do intervalo que isola a raiz, a tolerância *Toler* para o cálculo da raiz e o número máximo de iterações *IterMax*. A função $f(x)$ deve ser especificada de acordo com a linguagem de programação utilizada. Os parâmetros de saída são a raiz da equação *Raiz*, o número de iterações gastos *Iter* e a condição de erro *CondErro*, em que *CondErro* = 0 significa que a raiz foi calculada com sucesso e *CondErro* = 1 indica que a raiz não foi encontrada com a tolerância e o número máximo de iterações dados.

Durante as iterações, o ponto (a, F_a) é substituído por (b, F_b) e (b, F_b) é trocado pelo recém-calculado (x, F_x) . Por essa razão, antes das iterações, é verificado se $|F_a| < |F_b|$ de modo a ser abandonado, na primeira iteração, o ponto mais distante da raiz.

Exemplo 6.22 Determinar a raiz de $f(x) = 2x^3 - \cos(x + 1) - 3 = 0$ do Exemplo 6.17 pelo método da secante mostrado na Figura 6.12, com $\varepsilon \leq 0,01$, sabendo-se que $\xi \in [-1, 2]$.

```
% Os parametros de entrada
a = -1
b = 2
Toler = 0.0100
IterMax = 100
% produzem os resultados
```

```

Algoritmo Secante
{ Objetivo: Calcular a raiz de uma equação pelo método da secante }
parâmetros de entrada a, b, Toler, IterMax
{ limite inferior, limite superior, tolerância e número máximo de iterações }
parâmetros de saída Raiz, Iter, CondErro
{ raiz, número de iterações gastas e condição de erro, sendo }
{ CondErro = 0 se a raiz foi encontrada e CondErro = 1 em caso contrário. }
Fa ← f(a); Fb ← f(b) { avaliar a função em a e b }
se abs(Fa) < abs(Fb) então
    t ← a; a ← b; b ← t; t ← Fa; Fa ← Fb; Fb ← t
fimse
Iter ← 0; x ← b; Fx ← Fb
repita
    DeltaX ← -Fx / (Fb - Fa) * (b - a)
    x ← x + DeltaX; Fx ← f(x) { avaliar a função em x }
    escreva Iter, a, Fa, b, Fb, x, Fx, DeltaX
    se (abs(DeltaX) ≤ Toler e abs(Fx) ≤ Toler) ou Iter ≥ IterMax então
        interrompa
    fimse
    a ← b; Fa ← Fb; b ← x; Fb ← Fx; Iter ← Iter + 1
fimrepita
Raiz ← x
{ teste de convergência }
se abs(DeltaX) ≤ Toler e abs(Fx) ≤ Toler então
    CondErro ← 0
senão
    CondErro ← 1
fimse
fimalgoritmo

```

Figura 6.12 Algoritmo do método da secante.

(Ver significado da função abs na Tabela 1.1, na página 6.)

| Calculo de raiz de equacão pelo metodo da secante | | | | | | | |
|---|----------|----------|----------|----------|----------|------------|------------|
| iter | a | Fa | b | Fb | x | Fx | Delta_x |
| 0 | 2.00000 | 13.98999 | -1.00000 | -6.00000 | -0.09955 | -3.623e+00 | 9.005e-01 |
| 1 | -1.00000 | -6.00000 | -0.09955 | -3.62323 | 1.27313 | 1.773e+00 | 1.373e+00 |
| 2 | -0.09955 | -3.62323 | 1.27313 | 1.77312 | 0.82210 | -1.640e+00 | -4.510e-01 |
| 3 | 1.27313 | 1.77312 | 0.82210 | -1.64011 | 1.03883 | -3.068e-01 | 2.167e-01 |
| 4 | 0.82210 | -1.64011 | 1.03883 | -0.30676 | 1.08869 | 7.576e-02 | 4.986e-02 |
| 5 | 1.03883 | -0.30676 | 1.08869 | 0.07576 | 1.07881 | -2.438e-03 | -9.875e-03 |

Raiz = 1.07881

Iter = 5

CondErro = 0

Assim, a raiz da equação é $\xi \approx x_5 = 1,07881$.

O método da secante pode apresentar alguns problemas. Se a função não for, aproximadamente, linear no intervalo que contém a raiz, uma aproximação sucessiva pode sair deste intervalo como mostrado na Figura 6.11(a).

6.3.2 Método da *regula falsi*

Uma maneira de evitar problemas é garantir que a raiz esteja isolada no intervalo inicial e continue dentro dos novos intervalos gerados. O método da *regula falsi* retém o ponto no qual o valor da função tem sinal oposto ao valor da função no ponto mais recente, garantindo, desta forma, que a raiz continue isolada entre dois pontos, como mostrado na Figura 6.11(b).

O algoritmo do método da *regula falsi*, apresentado na Figura 6.13, encontra a raiz de uma equação $f(x) = 0$ contida em um intervalo $[a, b]$, com tolerância ε usando os critérios (6.2) e (6.4). Os parâmetros de entrada são o limite inferior a e o superior b do intervalo que contém a raiz, a tolerância $Toler$ para o cálculo da raiz e o número máximo de iterações $IterMax$. A função $f(x)$ deve ser especificada de acordo com a linguagem de programação adotada. Os parâmetros de saída são a raiz de $f(x) = 0$ *Raiz*, o número de iterações gastas *Iter* e a condição de erro *CondErro*, em que *CondErro* = 0 mostra que a raiz foi calculada com êxito e *CondErro* = 1 indica que a raiz não foi encontrada com a tolerância e o número máximo de iterações fornecidos.

Exemplo 6.23 Achar a raiz de $f(x) = 2x^3 - \cos(x + 1) - 3 = 0$ do Exemplo 6.17 usando o método da *regula falsi*, dado na Figura 6.13, com $\varepsilon \leq 0,01$, sabendo-se que $\xi \in [-1, 2]$.

% Os parâmetros de entrada

a = -1

b = 2

Toler = 0.0100

IterMax = 100

% produzem os resultados

| Calculo de raiz de equacão pelo metodo da regula falsi | | | | | | | |
|--|----------|----------|---------|----------|----------|------------|------------|
| iter | a | Fa | b | Fb | x | Fx | Delta_x |
| 0 | -1.00000 | -6.00000 | 2.00000 | 13.98999 | -0.09955 | -3.623e+00 | -2.100e+00 |
| 1 | -0.09955 | -3.62323 | 2.00000 | 13.98999 | 0.33235 | -3.163e+00 | 4.319e-01 |
| 2 | 0.33235 | -3.16277 | 2.00000 | 13.98999 | 0.63985 | -2.407e+00 | 3.075e-01 |
| 3 | 0.63985 | -2.40710 | 2.00000 | 13.98999 | 0.83952 | -1.551e+00 | 1.997e-01 |
| 4 | 0.83952 | -1.55114 | 2.00000 | 13.98999 | 0.95534 | -8.810e-01 | 1.158e-01 |
| 5 | 0.95534 | -0.88102 | 2.00000 | 13.98999 | 1.01723 | -4.631e-01 | 6.189e-02 |
| 6 | 1.01723 | -0.46306 | 2.00000 | 13.98999 | 1.04872 | -2.333e-01 | 3.149e-02 |
| 7 | 1.04872 | -0.23328 | 2.00000 | 13.98999 | 1.06432 | -1.150e-01 | 1.560e-02 |
| 8 | 1.06432 | -0.11498 | 2.00000 | 13.98999 | 1.07195 | -5.607e-02 | 7.628e-03 |
| 9 | 1.07195 | -0.05607 | 2.00000 | 13.98999 | 1.07565 | -2.719e-02 | 3.704e-03 |
| 10 | 1.07565 | -0.02719 | 2.00000 | 13.98999 | 1.07745 | -1.315e-02 | 1.793e-03 |
| 11 | 1.07745 | -0.01315 | 2.00000 | 13.98999 | 1.07831 | -6.355e-03 | 8.666e-04 |

Raiz = 1.07831

Iter = 11

CondErro = 0

```

Algoritmo RegulaFalsi
{ Objetivo: Calcular a raiz de uma equação pelo método da regula falsi }
parâmetros de entrada a, b, Toler, IterMax
{ limite inferior, limite superior, tolerância e número máximo de iterações }
parâmetros de saída Raiz, Iter, CondErro
{ raiz, número de iterações gastos e condição de erro, sendo }
{ CondErro = 0 se a raiz foi encontrada e CondErro = 1 em caso contrário. }
Fa ← f(a); Fb ← f(b); { avaliar a função em a e b }
se Fa * Fb > 0 então
    escreva "função não muda de sinal nos extremos do intervalo dado"
    abandone
fimse
se Fa > 0 então
    t ← a; a ← b; b ← t; t ← Fa; Fa ← Fb; Fb ← t
fimse
Iter ← 0; x ← b; Fx ← Fb
repita
    DeltaX ← -Fx / (Fb - Fa) * (b - a)
    x ← x + DeltaX; Fx ← f(x); { avaliar a função em x }
    escreva Iter, a, Fa, b, Fb, x, Fx, DeltaX
    se (abs(DeltaX) ≤ Toler e abs(Fx) ≤ Toler) ou Iter ≥ IterMax então
        interrompa
    fimse
    se Fx < 0 então
        a ← x; Fa ← Fx
    senão
        b ← x; Fb ← Fx
    fimse; Iter ← Iter + 1
fimrepita
Raiz ← x
{ teste de convergência }
se abs(DeltaX) ≤ Toler e abs(Fx) ≤ Toler então
    CondErro ← 0
senão
    CondErro ← 1
fimse
finalgoritmo

```

Figura 6.13 Algoritmo do método da *regula falsi*.

(Ver significado da função abs na Tabela 1.1, na página 6.)

A raiz da equação é $\xi \approx x_{11} = 1,07831$. Neste exemplo, a convergência para a raiz só se fez de um lado do intervalo, tornando este método mais lento que o da secante, embora mais robusto. Quanto mais longe o ponto fixo for da raiz, mais lenta será a convergência.

Conforme será visto mais adiante, o método da *regula falsi* tem uma ordem de convergência menor que o da secante porque o ponto mantido fixo não é geralmente um dos mais recentes.

6.3.3 Método pégaso

De modo similar aos outros métodos baseados em aproximação linear, no método pégaso [12] a seqüência $\{x_i\}$ é obtida pela fórmula de recorrência

$$x_{k+1} = x_k - \frac{f(x_k)}{f(x_k) - f(x_{k-1})}(x_k - x_{k-1}), \quad k = 1, 2, 3, \dots$$

Os pontos $[x_{k-1}, f(x_{k-1})]$ e $[x_k, f(x_k)]$ pelos quais será traçada a reta para obter x_{k+1} são escolhidos de modo que $f(x_{k-1})$ e $f(x_k)$ tenham sempre sinais opostos, garantindo assim que $\xi \in [x_{k-1}, x_k]$. Além do mais, o valor de $f(x_{k-1})$ é reduzido por um fator igual a $f(x_k)/(f(x_k) + f(x_{k+1}))$ de modo a evitar a retenção de um ponto, como ocorre na *regula falsi*. Deste modo, a reta pode ser traçada por um ponto não pertencente à curva de $f(x)$.

A Figura 6.14 ilustra graficamente o método pégaso. Deve ser observado que a estimativa x_4 da raiz é obtida usando os pontos de coordenadas $[x_3, f(x_3)]$ e $[x_1, p]$, sendo $p = f(x_1) \times f(x_2)/(f(x_2) + f(x_3))$, o qual não pertence à função $f(x)$. É claro, pela figura, que x_4 é uma melhor aproximação da raiz do que x'_4 que seria obtido pelo método da *regula falsi* (linha tracejada).

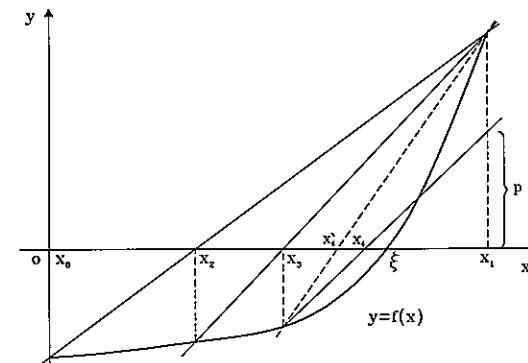


Figura 6.14 Interpretação gráfica do método pégaso.

O algoritmo da Figura 6.15 calcula, pelo método pégaso, uma raiz da equação $f(x) = 0$ pertencente ao intervalo $[a, b]$, com tolerância ϵ usando os critérios (6.2) e (6.4). Os parâmetros de entrada são o limite inferior *a* e o superior *b* do intervalo que isola a raiz, a tolerância *Toler* para o cálculo da raiz e o número máximo de iterações *IterMax*. A função *f(x)* deve ser especificada de acordo com a linguagem de programação utilizada. Os parâmetros de saída são a raiz da equação *Raiz*, o número de iterações gastas *Iter* e a

condição de erro $CondErro$, em que $CondErro = 0$ indica que a raiz foi calculada com sucesso e $CondErro = 1$ avisa que a raiz não foi encontrada com a tolerância e o número máximo de iterações dados.

Algoritmo Pégaso

```
{ Objetivo: Calcular a raiz de uma equação pelo método pégaso }
parâmetros de entrada a, b, Toler, IterMax
{ limite inferior, limite superior, tolerância e número máximo de iterações }
parâmetros de saída Raiz, Iter, CondErro
{ raiz, número de iterações gastos e condição de erro, sendo }
{ CondErro = 0 se a raiz foi encontrada e CondErro = 1 em caso contrário. }
Fa ← f(a); Fb ← f(b); { avaliar a função em a e b }
x ← b; Fx ← Fb; Iter ← 0
repita
    DeltaX ← -Fx / (Fb - Fa) * (b - a)
    x ← x + DeltaX; Fx ← f(x); { avaliar a função em x }
    escreva Iter, a, Fa, b, Fb, x, Fx, DeltaX
    se (abs(DeltaX) ≤ Toler e abs(Fx) ≤ Toler) ou Iter ≥ IterMax então
        interrompa
    fimse
    se Fx * Fb < 0 então
        a ← b; Fa ← Fb
    senão
        Fa ← Fa * Fb / (Fb + Fx)
    fimse
    b ← x; Fb ← Fx
    Iter ← Iter + 1
fimrepita
Raiz ← x
{ teste de convergência }
se abs(DeltaX) ≤ Toler e abs(Fx) ≤ Toler então
    CondErro ← 0
senão
    CondErro ← 1
fimse
finalgoritmo
```

Figura 6.15 Algoritmo do método pégaso.

(Ver significado da função abs na Tabela 1.1, na página 6.)

Exemplo 6.24 Calcular com $\varepsilon \leq 0,01$, a raiz de $f(x) = 2x^3 - \cos(x+1) - 3 = 0$ do Exemplo 6.17, pelo método pégaso mostrado na Figura 6.15, sabendo-se que $\xi \in [-1, 2]$.

% Os parametros de entrada

a = -1

b = 2

Toler = 0.0100

IterMax = 100

% produzem os resultados

Calculo de raiz de equacao pelo metodo pegaso

| iter | a | Fa | b | Fb | x | Fx | Delta_x |
|------|----------|----------|----------|----------|----------|------------|------------|
| 0 | -1.00000 | -6.00000 | 2.00000 | 13.98999 | -0.09955 | -3.623e+00 | -2.100e+00 |
| 1 | 2.00000 | 13.98999 | -0.09955 | -3.62323 | 0.33235 | -3.163e+00 | 4.319e-01 |
| 2 | 2.00000 | 7.46964 | 0.33235 | -3.16277 | 0.82842 | -1.608e+00 | 4.961e-01 |
| 3 | 2.00000 | 4.95180 | 0.82842 | -1.60817 | 1.11563 | 2.954e-01 | 2.872e-01 |
| 4 | 0.82842 | -1.60817 | 1.11563 | 0.29537 | 1.07106 | -6.294e-02 | 4.457e-02 |
| 5 | 1.11563 | 0.29537 | 1.07106 | -0.06294 | 1.07889 | -1.807e-03 | 7.828e-03 |

Raiz = 1.07889

Iter = 5

CondErro = 0

Portanto, a raiz da equação é $\xi \approx x_5 = 1,07889$. ■

Exemplo 6.25 Achar o ponto de máximo μ do polinômio $P(x) = x^4 + 2x^3 - 13x^2 - 14x + 24$, pelo método pégaso, com $\varepsilon \leq 10^{-5}$, sabendo-se que $\mu \in [-1, 1]$, de acordo com a Figura 6.5(b).

A condição de máximo de uma função é que a derivada primeira se anule e que a derivada segunda seja negativa. Assim, o problema é equivalente a calcular uma raiz de $P'(x) = 0$

$$P'(x) = 4x^3 + 6x^2 - 26x - 14 = 0.$$

% Os parametros de entrada

a = -1

b = 1

Toler = 1.0000e-05

IterMax = 100

% produzem os resultados

Calculo de raiz de equacao pelo metodo pegaso

| iter | a | Fa | b | Fb | x | Fx | Delta_x |
|------|----------|----------|----------|-----------|----------|------------|------------|
| 0 | -1.00000 | 14.00000 | 1.00000 | -30.00000 | -0.36364 | -3.944e+00 | -1.364e+00 |
| 1 | -1.00000 | 12.37317 | -0.36364 | -3.94440 | -0.51746 | 5.064e-01 | -1.538e-01 |
| 2 | -0.36364 | -3.94440 | -0.51746 | 0.50640 | -0.49996 | -1.135e-03 | 1.750e-02 |
| 3 | -0.51746 | 0.50640 | -0.49996 | -0.00114 | -0.50000 | 4.764e-08 | -3.914e-05 |
| 4 | -0.49996 | -0.00114 | -0.50000 | 0.00000 | -0.50000 | 0.0000e+00 | 1.643e-09 |

Raiz = -0.50000

Iter = 4

CondErro = 0

Como $P''(x) = 12x^2 + 12x - 26$, então $P''(-0,5) = -29 < 0$, confirmando que $\mu \approx x_4 = -0,5$ é um ponto de máximo. ■

6.3.4 Ordem de convergência

Para determinar as ordens de convergência dos métodos baseados em aproximação linear, considere uma estimativa x_2 da raiz ξ obtida por uma reta passando pelos pontos de coordenadas $[x_0, f(x_0)]$ e $[x_1, f(x_1)]$

$$x_2 = x_1 - \frac{f(x_1)}{f(x_1) - f(x_0)}(x_1 - x_0) = \frac{x_1 f(x_0) - x_0 f(x_1)}{f(x_0) - f(x_1)}.$$

Expandindo $f(x_k)$ em série de Taylor com relação à raiz ξ e considerando o erro da k -ésima iteração dado por (6.5), tem-se

$$\epsilon_2 + \xi = \frac{(\epsilon_1 + \xi) \left(\epsilon_0 f'(\xi) + \epsilon_0^2 \frac{f''(\xi)}{2} + \dots \right) - (\epsilon_0 + \xi) \left(\epsilon_1 f'(\xi) + \epsilon_1^2 \frac{f''(\xi)}{2} + \dots \right)}{(\epsilon_0 - \epsilon_1) f'(\xi) + (\epsilon_0^2 - \epsilon_1^2) \frac{f''(\xi)}{2} + \dots}$$

Simplificando

$$\epsilon_2 = \frac{\frac{f''(\xi)}{2} \epsilon_0 \epsilon_1 (\epsilon_0 - \epsilon_1) + \dots}{f'(\xi) (\epsilon_0 - \epsilon_1) + \frac{f''(\xi)}{2} (\epsilon_0 - \epsilon_1) (\epsilon_0 + \epsilon_1) + \dots}.$$

Dividindo por $f'(\xi)(\epsilon_0 - \epsilon_1)$

$$\begin{aligned} \epsilon_2 &= \frac{\frac{f''(\xi)}{2f'(\xi)} \epsilon_0 \epsilon_1 + \dots}{1 + \frac{f''(\xi)}{2f'(\xi)} (\epsilon_0 + \epsilon_1) + \dots}, \\ \epsilon_2 &\approx \frac{f''(\xi)}{2f'(\xi)} \epsilon_0 \epsilon_1. \end{aligned} \quad (6.10)$$

No caso do método da *regula falsi*, como a raiz deve ficar sempre isolada em um intervalo, então x_0 será geralmente fixo durante várias iterações. Conseqüentemente, o erro ϵ_0 também será fixo, resultando que o erro da k -ésima iteração será da forma

$$\epsilon_{k+1} = K_r \epsilon_k.$$

Assim, por (6.6) verifica-se que o método da *regula falsi* apresenta convergência de primeira ordem, conforme Acton [2]. Para o método da secante, como os valores de x_k e x_{k-1} são sempre atualizados, (6.10) pode ser generalizada por

$$\epsilon_{k+1} = C \epsilon_{k-1} \epsilon_k.$$

Por (6.6)

$$|\epsilon_{k+1}| = K |\epsilon_k|^\gamma.$$

Usando esta equação na equação anterior, tem-se

$$K |\epsilon_k|^\gamma = |C| |\epsilon_k| \left(\frac{|\epsilon_k|}{K} \right)^{\frac{1}{\gamma}}.$$

Rearranjando,

$$K^{1+\frac{1}{\gamma}} |\epsilon_k|^\gamma = |C| |\epsilon_k|^{1+\frac{1}{\gamma}}.$$

Como a ordem de convergência γ deve ser positiva e pela equação acima

$$\gamma = 1 + \frac{1}{\gamma} \rightarrow \gamma^2 - \gamma - 1 = 0 \rightsquigarrow \gamma = \frac{1 + \sqrt{5}}{2} \approx 1,61803.$$

Conseqüentemente, o método da secante tem ordem de convergência igual à relação áurea, como mostrado por Hildebrand [22]. Além disso,

$$K^{1+\frac{1}{\gamma}} = |C|.$$

Como $1 + \frac{1}{\gamma} = \gamma$, então

$$K^\gamma = |C| \rightarrow K = |C|^{\frac{1}{\gamma}} = |C|^{\gamma-1}.$$

Portanto, por (6.6) e (6.10), o método da secante apresenta

$$|\epsilon_{k+1}| \approx \left| \frac{f''(\xi)}{2f'(\xi)} \right|^{\gamma-1} |\epsilon_k|^\gamma.$$

Segundo Dowell e Jarratt [12], o método pégaso tem ordem de convergência 1,642, mostrando que ele é superior ao método da secante e ao da *regula falsi*.

6.4 Métodos baseados em aproximação quadrática

Na Seção 6.3 foram vistos métodos para cálculo de raízes baseados na aproximação da função $f(x)$ por um polinômio interpolador de grau 1. Naqueles métodos, uma estimativa da raiz é o ponto onde a reta intercepta o eixo das abscissas. A estimativa da raiz de $f(x) = 0$ pode ser ainda melhor se em vez de um polinômio interpolador de grau 1 for utilizado um polinômio de grau 2.

6.4.1 Método de Muller

O método de Muller [33] consiste em aproximar a função $f(x)$, na vizinhança da raiz $\xi \in [x_0, x_2]$, por um polinômio quadrático. Este polinômio é construído de modo a passar pelos três pontos de coordenadas $[x_0, f(x_0)]$, $[x_1, f(x_1)]$ e $[x_2, f(x_2)]$. Deste modo, o zero do polinômio é usado como uma estimativa da raiz ξ de $f(x) = 0$. O processo é, então, repetido usando sempre os três pontos mais próximos da raiz.

Um polinômio de segundo grau que passa pelos três pontos de coordenadas $[x_{i-2}, f(x_{i-2})]$, $[x_{i-1}, f(x_{i-1})]$ e $[x_i, f(x_i)]$ na forma

$$P_2(v) = av^2 + bv + c \quad (6.11)$$

onde $v = x - x_{i-1}$ pode ser construído de acordo com a Figura 6.16. Para cada um dos três pontos, tem-se que

$$\begin{aligned} P_2(x_{i-2}) &= f(x_{i-2}) \rightarrow a(x_{i-2} - x_{i-1})^2 + b(x_{i-2} - x_{i-1}) + c = f(x_{i-2}), \\ P_2(x_{i-1}) &= f(x_{i-1}) \rightarrow a(0)^2 + b(0) + c = f(x_{i-1}) \rightarrow c = f(x_{i-1}) \text{ e} \\ P_2(x_i) &= f(x_i) \rightarrow a(x_i - x_{i-1})^2 + b(x_i - x_{i-1}) + c = f(x_i). \end{aligned} \quad (6.12)$$

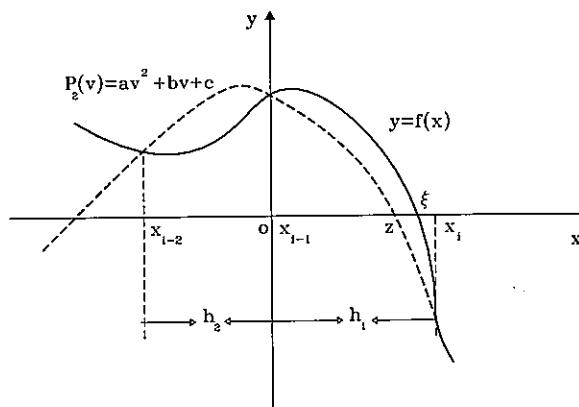


Figura 6.16 Interpretação gráfica do método de Muller.

Definindo

$$h_1 = x_i - x_{i-1} \text{ e}$$

$$h_2 = x_{i-1} - x_{i-2}$$

e, em vista de (6.12), é obtido o seguinte sistema linear em termos das incógnitas a e b

$$h_2^2 a - h_2 b = f(x_{i-2}) - f(x_{i-1}),$$

$$h_1^2 a + h_1 b = f(x_i) - f(x_{i-1}),$$

cuja solução é

$$a = \frac{1}{h_1(h_1 + h_2)} (f(x_i) - (r+1)f(x_{i-1}) + rf(x_{i-2})) \quad (6.13)$$

sendo $r = h_1/h_2$ e

$$b = \frac{1}{h_1} (f(x_i) - f(x_{i-1})) - ah_1, \quad (6.14)$$

onde a é dado por (6.13). Os dois zeros do polinômio de grau 2 em v (6.11) são

$$z = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.$$

Para ser obtida a raiz mais próxima de x_{i-1} , o sinal na expressão acima deve ser escolhido de modo a tornar o numerador o menor possível. Assim, em vista da transformação $v = x - x_{i-1}$, a próxima estimativa da raiz ξ de $f(x) = 0$ é

$$x_{i+1} = x_{i-1} + \frac{-b + \operatorname{sinal}(b)\sqrt{b^2 - 4ac}}{2a},$$

onde a , b e c são dados por (6.13), (6.14) e (6.12), respectivamente. Na próxima iteração, devem ser utilizados os três pontos mais próximos de ξ .

O algoritmo mostrado na Figura 6.17 determina, pelo método de Muller, uma raiz da equação $f(x) = 0$ que está no intervalo $[a, c]$, com tolerância ε usando os critérios (6.2) e (6.4). Os parâmetros de entrada são o limite inferior a e o superior c do intervalo que contém a raiz, a tolerância $Toler$ para o cálculo da raiz e o número máximo de iterações $IterMax$. A função $f(x)$ deve ser especificada de acordo com a linguagem de programação escolhida. Os parâmetros de saída são a raiz de $f(x) = 0$ $Raiz$, o número de iterações gastos $Iter$ e a condição de erro $CondErro$, em que $CondErro = 0$ significa que a raiz foi determinada com sucesso e $CondErro = 1$ mostra que a raiz não foi encontrada com a tolerância e o número máximo de iterações fornecidos.

Exemplo 6.26 Calcular com $\varepsilon \leq 0,01$, a raiz de $f(x) = 2x^3 - \cos(x+1) - 3 = 0$ do Exemplo 6.17, pelo método de Muller apresentado na Figura 6.17, sabendo-se que $\xi \in [-1, 2]$.

% Os parâmetros de entrada

$$a = -1$$

$$b = 2$$

$$Toler = 0.0100$$

$$IterMax = 100$$

% produzem os resultados

| Calculo de raiz de equação pelo método de Muller | | | | | | Delta_x |
|--|----------|---------|---------|---------|--------------|-------------|
| iter | a | b | c | x | Fx | |
| 0 | -1.00000 | 0.50000 | 2.00000 | 0.86331 | -1.42476e+00 | 3.63315e-01 |
| 1 | 0.50000 | 0.86331 | 2.00000 | 1.05488 | -1.86933e-01 | 1.91564e-01 |
| 2 | 0.86331 | 1.05488 | 2.00000 | 1.07803 | -8.58214e-03 | 2.31508e-02 |
| 3 | 1.05488 | 1.07803 | 2.00000 | 1.07912 | -4.55606e-05 | 1.08694e-03 |

$$Raiz = 1.07912$$

$$Iter = 3$$

$$CondErro = 0$$

Assim, a raiz da equação é $\xi \approx x_3 = 1,07912$.

Exemplo 6.27 Achar a raiz de $f(x) = 0,05x^3 - 0,4x^2 + 3 \operatorname{sen}(x)x = 0$ do Exemplo 6.18, com $\varepsilon \leq 10^{-10}$, que se encontra no intervalo $[10, 12]$, usando o método de Muller.

Algoritmo Muller

```

{ Objetivo: Calcular a raiz de uma equação pelo método de Muller }
parametros de entrada a, c, Toler, IterMax
{ limite inferior, limite superior, tolerância e número máximo de iterações }
parametros de saída Raiz, Iter, CondErro
{ raiz, número de iterações gastas e condição de erro, sendo }
{ CondErro = 0 se a raiz foi encontrada e CondErro = 1 em caso contrário. }
{ avaliar a função em a, c e b }
Fa ← f(a); Fc ← f(c); b ← (a + c)/2; Fb ← f(b)
x ← b; Fx ← Fb; DeltaX ← c - a; Iter ← 0
repita
  h1 ← c - b; h2 ← b - a; r ← h1/h2; t ← x
  A ← ((Fc - (r + 1) * Fb + r * Fa)) / ((h1 + (h1 + h2)))
  B ← (Fc - Fb) / h1 - A * h1
  C ← Fb; z ← (-B + sinal(B) * raiz2(B^2 - 4 * A * C)) / (2 * A)
  x ← b + z; DeltaX ← x - t; Fx ← f(x); { avaliar a função em x }
  escreva Iter, a, b, c, x, Fx, DeltaX
  se (abs(DeltaX) ≤ Toler e abs(Fx) ≤ Toler) ou Iter ≥ IterMax então
    interrompa
  fimse
  se x > b então
    a ← b; Fa ← Fb
  senão
    c ← b; Fc ← Fb
  fimse
  b ← x; Fb ← Fx; Iter ← Iter + 1
fimrepita
Raiz ← x
{ teste de convergência }
se abs(DeltaX) ≤ Toler e abs(Fx) ≤ Toler então
  CondErro ← 0
senão
  CondErro ← 1
fimse
finalgoritmo

```

Figura 6.17 Algoritmo do método de Muller.

(Ver significado das funções abs, raiz₂ e sinal na Tabela 1.1, na página 6.)

```

% Os parametros de entrada
a = 10
b = 12
Toler = 1.0000e-10
IterMax = 100

```

% produzem os resultados

| Calculo de raiz de equacao pelo metodo de Muller | | | | | | |
|--|----------|----------|----------|----------|--------------|--------------|
| iter | a | b | c | x | Fx | Delta_x |
| 0 | 10.00000 | 11.00000 | 12.00000 | 11.74014 | -1.25090e-01 | 7.40141e-01 |
| 1 | 11.00000 | 11.74014 | 12.00000 | 11.74398 | 1.54925e-03 | 3.83681e-03 |
| 2 | 11.74014 | 11.74398 | 12.00000 | 11.74393 | -1.45315e-07 | -4.68547e-05 |
| 3 | 11.74014 | 11.74393 | 11.74398 | 11.74393 | 1.06581e-14 | 4.39453e-09 |
| 4 | 11.74393 | 11.74393 | 11.74398 | 11.74393 | 1.06581e-14 | 0.00000e+00 |

Raiz = 11.74393

Iter = 4

CondErro = 0

A raiz da equação é $\xi \approx x_4 = 11,74393$.

Hildebrand [22] mostrou que, para o método de Muller, a expressão (6.6) apresenta a forma

$$|\epsilon_{k+1}| \approx \left| \frac{f'''(\xi)}{6f'(\xi)} \right|^{\frac{\gamma-1}{2}} |\epsilon_k|^\gamma$$

onde γ é a raiz positiva da equação

$$\gamma = 1 + \frac{1}{\gamma} + \frac{1}{\gamma^2} \rightarrow \gamma^3 - \gamma^2 - \gamma - 1 = 0,$$

indicando que o método de Muller tem ordem de convergência $\gamma \approx 1,8393$.**6.4.2 Método de van Wijngaarden-Dekker-Brent**

Este método é o resultado da combinação da interpolação inversa quadrática e da bissecção implementados de forma a garantir que a raiz continue sempre isolada. Ele foi proposto inicialmente por van Wijngaarden, Dekker e outros e melhorado posteriormente por Brent [6]. Na interpolação quadrática, a forma analítica de um polinômio $P_2(x) \approx f(x) = y$ é determinada a partir de três pontos de coordenadas $[x_{i-2}, f(x_{i-2})]$, $[x_{i-1}, f(x_{i-1})]$ e $[x_i, f(x_i)]$; assim, para obter um valor aproximado de $f(t)$, basta avaliar $P_2(t)$. Por outro lado, na interpolação inversa quadrática, o polinômio interpolador de grau 2

$$\Pi_2(y) \approx f^{-1}(y) = x \quad (6.15)$$

é construído a partir dos pontos de coordenadas $[f(x_{i-2}), x_{i-2}]$, $[f(x_{i-1}), x_{i-1}]$ e $[f(x_i), x_i]$ (ver Seção 3.13.2). Neste caso, para ter um valor aproximado de $f^{-1}(z)$ é necessário avaliar $\Pi_2(z)$. Este polinômio $\Pi_2(y)$ pode ser obtido por interpolação de Lagrange

$$\begin{aligned} \Pi_2(y) = & \frac{(y - f(x_{i-1}))(y - f(x_i))}{x_{i-2}(f(x_{i-2}) - f(x_{i-1}))(f(x_{i-2}) - f(x_i))} \\ & + \frac{(y - f(x_{i-2}))(y - f(x_i))}{x_{i-1}(f(x_{i-1}) - f(x_{i-2}))(f(x_{i-1}) - f(x_i))} \\ & + \frac{(y - f(x_{i-2}))(y - f(x_{i-1}))}{x_i(f(x_i) - f(x_{i-2}))(f(x_i) - f(x_{i-1}))}, \end{aligned}$$

conforme mostrado na Seção 3.2. Como $y = f(x) \rightarrow x = f^{-1}(y)$, então uma aproximação da raiz ξ de $f(x) = 0$ é o ponto de abscissa correspondente à $f^{-1}(0)$. Em vista de (6.15), esta aproximação é dada por $x = \Pi_2(0)$.

Na Figura 6.18, é mostrado um algoritmo baseado nos programas das referências [32, 34] para calcular, pelo método de van Wijngaarden-Dekker-Brent, uma raiz da equação $f(x) = 0$ pertencente ao intervalo $[a, b]$, com tolerância ε usando critérios semelhantes a (6.2) e (6.4). Os parâmetros de entrada são o limite inferior a e o superior b do intervalo que isola a raiz, a tolerância $Toler$ para o cálculo da raiz e o número máximo de iterações $IterMax$. A função $f(x)$ deve ser especificada de acordo com a linguagem de programação usada. Os parâmetros de saída são a raiz da equação $Raiz$, o número de iterações gastos $Iter$ e a condição de erro $CondErro$, em que $CondErro = 0$ indica que a raiz foi calculada com sucesso e $CondErro = 1$ avisa que a raiz não foi encontrada com a tolerância e o número máximo de iterações dados.

Exemplo 6.28 Calcular pelo método de van Wijngaarden-Dekker-Brent, descrito na Figura 6.18, a menor raiz de $P(x) = x^4 + 2x^3 - 13x^2 - 14x + 24 = 0$ do Exemplo 6.11, com $\varepsilon \leq 10^{-10}$, sabendo-se que $\xi \in [-5, -3]$ de acordo com a Figura 6.5(b).

```
% Os parametros de entrada
a = -5
b = -3
Toler = 1.0000e-10
IterMax = 100
% produzem os resultados
    Calculo de raiz pelo metodo de van Wijngaarden-Dekker-Brent
iter      a          c          b          Fb          z
0   -5.00000  -5.00000  -3.00000  -2.40000e+01  -1.00000e+00
1   -3.00000  -5.00000  -3.28571  -2.47397e+01  -8.57143e-01
2   -3.28571  -3.28571  -4.14286  1.12453e+01  4.28571e-01
3   -4.14286  -4.14286  -3.87500  -7.85522e+00  -1.33929e-01
4   -3.87500  -4.14286  -3.98516  -1.02599e+00  -7.88495e-02
5   -3.98516  -3.98516  -4.00032  2.26777e-02  7.58292e-03
6   -4.00032  -4.00032  -4.00000  -2.86125e-04  -1.63983e-04
7   -4.00000  -4.00032  -4.00000  -7.80927e-08  -1.61940e-04
8   -4.00000  -4.00032  -4.00000  0.00000e+00  -1.61940e-04

Raiz      = -4.00000
Iter      = 8
CondErro = 0
```

ou seja, a menor raiz de $P(x) = x^4 + 2x^3 - 13x^2 - 14x + 24 = 0$ é $\xi = x_8 = -4$.

Exemplo 6.29 Calcular a raiz de $f(x) = 0,05x^3 - 0,4x^2 + 3\operatorname{sen}(x)x = 0$ do Exemplo 6.18, com $\varepsilon \leq 10^{-10}$, que se encontra no intervalo $[10, 12]$, utilizando o método de van Wijngaarden-Dekker-Brent.

```
Algoritmo van Wijngaarden-Dekker-Brent
{ Objetivo: Calcular a raiz pelo método de van Wijngaarden-Dekker-Brent }
parâmetros de entrada a, b, Toler, IterMax
{ limite inferior, limite superior, tolerância e número máximo de iterações }
parâmetros de saída Raiz, Iter, CondErro
{ raiz, número de iterações gastos e condição de erro, sendo }
{ CondErro = 0 se a raiz foi encontrada e CondErro = 1 em caso contrário. }
Fa ← f(a); Fb ← f(b); { avaliar a função em a e b }
se Fa * Fb > 0 então
    escreva "a função não muda de sinal nos extremos do intervalo dado"
    abandone
fimse
c ← b; Fc ← Fb; Iter ← 0
repita
{ altera a, b e c para que b seja a melhor estimativa da raiz }
se Fb * Fc > 0 então c ← a; Fc ← Fa; d ← b - a; e ← d; fimse
se abs(Fc) < abs(Fb) então
    a ← b; b ← c; c ← a; Fa ← Fb; Fb ← Fc; Fc ← Fa
fimse
Tol ← 2 * Toler + max(abs(b), 1); z ← ((c - b)/2)
escreva Iter, a, c, b, Fb, z
{ teste de convergência }
se abs(z) ≤ Tol ou Fb = 0 ou Iter ≥ IterMax então interrompa, fimse
{ escolha entre interpolação e bissecção }
se abs(c) ≥ Tol e abs(Fc) > abs(Fb) então
    s ← Fb/Fa
    se a = c então { interpolação linear }
        p ← 2 * z * s; q ← 1 - s
    senão { interpolação inversa quadrática }
        q ← Fa/Fc; r ← Fb/Fc; p ← s * (2 * z * q * ((q - r) - ((b - a) * (r - 1)))
        q ← ((q - 1) * (r - 1)) * (s - 1)
    fimse
    se p > 0 então q ← -q, senão p ← -p, fimse
    se 2 * p < min(3 * z * q - abs(Tol * q), abs(c * q)) então { aceita interpolação }
        e ← d; d ← p/q
    senão { usa bissecção devido à falha na interpolação }
        d ← z; c ← z
    fimse
    senão { bissecção }
        d ← z; e ← z
    fimse
    a ← b; Fa ← Fb
    se abs(d) > Tol então b ← b + d, senão b ← b + sinal(z) * Tol, fimse
    Iter ← Iter + 1; Fb ← f(b) { avaliar a função em b }
fimrepita
Raiz ← b
se abs(z) ≤ Tol ou Fb = 0 então CondErro ← 0, senão CondErro ← 1, fimse
finalgoritmo
```

Figura 6.18 Algoritmo do método de van Wijngaarden-Dekker-Brent.

(Ver significado das funções abs, max, min e sinal na Tabela 1.1, na página 6.)

```
% Os parametros de entrada
a = 10
b = 12
Toler = 1.0000e-10
IterMax = 100
% produzem os resultados
Calculo de raiz pelo metodo de van Wijngaarden-Dekker-Brent
iter      a          c          b          Fb          z
0  12.00000  12.00000  10.00000 -6.32063e+00  1.00000e+00
1  10.79988  10.79988  12.00000  9.48337e+00 -6.00061e-01
2  12.00000  12.00000  11.54358 -5.94963e+00  2.28208e-01
3  11.54358  12.00000  11.71954 -7.96853e-01  1.40231e-01
4  11.71954  11.71954  11.74464  2.34449e-02 -1.25507e-02
5  11.74464  11.74464  11.74392 -2.86520e-04  3.58711e-04
6  11.74392  11.74464  11.74393 -1.00128e-07  3.54380e-04
7  11.74393  11.74393  11.74393  1.06581e-14 -1.51400e-09
```

```
Raiz      = 11.74393
Iter      = 7
CondErro = 0
```

A raiz procurada é $\xi \approx x_7 = 11,74393$.

Segundo Brent [6], a convergência pelo método é garantida desde que haja uma raiz no intervalo. A combinação da certeza de convergência do método da bisseção com a rapidez de um método de ordem de convergência maior como o da interpolação inversa quadrática resulta em um esquema robusto e eficiente.

O método de van Wijngaarden-Dekker-Brent é recomendado como sendo o mais adequado para o cálculo do zero de uma função de uma variável quando a sua derivada não estiver disponível [32, 34].

6.5 Métodos baseados em tangente

Com exceção do método da bisseção, os demais métodos descritos nas seções anteriores são baseados na aproximação de um arco da curva de $f(x)$ por polinômios lineares e quadráticos. A seguir, serão apresentados dois métodos baseados no cálculo da tangente à curva de $f(x)$.

6.5.1 Método de Newton

Sejam ξ a única raiz de $f(x) = 0$ no intervalo $[a, b]$ e x_k uma aproximação desta raiz, sendo $x_0 \in [a, b]$. Além disso, as derivadas $f'(x)$ e $f''(x)$ devem existir, ser contínuas e com sinal constante neste intervalo. Geometricamente, o método de Newton é equivalente a aproximar um arco da curva por uma reta tangente traçada a partir de um ponto da curva, o que faz com que ele seja conhecido também como o método das tangentes. Considere a Figura 6.19, na qual

$$\tan(\alpha) = \frac{f(x_0)}{x_0 - x_1} = f'(x_0) \rightarrow x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \quad \text{e}$$

$$\tan(\beta) = \frac{f(x_1)}{x_1 - x_2} = f'(x_1) \rightarrow x_2 = x_1 - \frac{f(x_1)}{f'(x_1)}.$$

A generalização das expressões acima fornece a fórmula de recorrência do método de Newton

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \dots \quad (6.16)$$

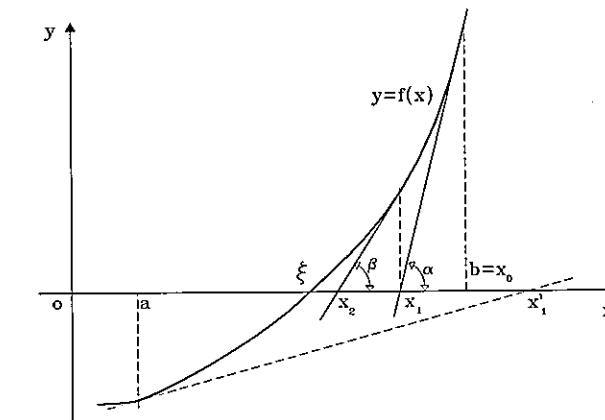


Figura 6.19 Interpretação gráfica do método de Newton.

A fórmula do método de Newton pode também ser deduzida analiticamente. Para isso seja

$$\xi = x_k + \delta_k \quad (6.17)$$

tal que δ_k tenha um valor pequeno. Fazendo uma expansão em série de Taylor

$$f(\xi) = f(x_k + \delta_k) \approx f(x_k) + f'(x_k)\delta_k = 0 \rightarrow \delta_k = -\frac{f(x_k)}{f'(x_k)}.$$

Substituindo essa correção em (6.17), obtém-se (6.16). Pela Figura 6.19, a seqüência produzida por (6.16) convergirá para a raiz ξ se o valor inicial for $x_0 = b$. No entanto, para aquela figura, o processo pode não convergir se $x_0 = a$, pois ter-se-á $x'_1 \notin [a, b]$. A questão da escolha do valor inicial de modo a garantir a convergência para a raiz é resolvida pelo teorema, a seguir, apresentado por Demidovich e Maron [10].

Teorema 6.6 Se $f(a)f(b) < 0$, e $f'(x)$ e $f''(x)$ forem não nulas e preservarem o sinal em $[a, b]$, então partindo-se da aproximação inicial $x_0 \in [a, b]$ tal que $f(x_0)f''(x_0) > 0$ é possível construir, pelo método de Newton (6.16), uma seqüência $\{x_i\}$ que converja para a raiz ξ de $f(x) = 0$.

Por este teorema, o valor inicial x_0 deve ser um ponto no qual a função tenha o mesmo sinal de sua derivada segunda, isto é, se $f''(x_0) > 0$, então x_0 é tal que $f(x_0) > 0$, e por outro lado, se $f''(x_0) < 0$, então $f(x_0) < 0$.

Um algoritmo do método de Newton é mostrado na Figura 6.20, no qual a raiz de uma equação $f(x) = 0$ é calculada com tolerância ϵ usando os critérios (6.2) e (6.4). Os parâmetros de entrada são o valor inicial x_0 , a tolerância $Toler$ para o cálculo da raiz e o número máximo de iterações $IterMax$. A função $f(x)$ e sua derivada $f'(x)$ devem ser especificadas de acordo com a linguagem de programação utilizada. Os parâmetros de saída são a raiz de $f(x) = 0$ $Raiz$, o número de iterações gastos $Iter$ e a condição de erro $CondErro$, em que $CondErro = 0$ mostra que a raiz foi calculada com sucesso e $CondErro = 1$ avisa que a raiz não foi encontrada com a tolerância e o número máximo de iterações fornecidos.

```

Algoritmo Newton
{ Objetivo: Calcular a raiz de uma equação pelo método de Newton }
parâmetros de entrada x0, Toler, IterMax
{ valor inicial, tolerância e número máximo de iterações }
parâmetros de saída Raiz, Iter, CondErro
{ raiz, número de iterações gastas e condição de erro, sendo }
{ CondErro = 0 se a raiz foi encontrada e CondErro = 1 em caso contrário. }
{ avaliar a função e sua derivada em x0 }
Fx ← f(x0); DFx ← f'(x0); x ← x0; Iter ← 0
escreva Iter, x, DFx, Fx
repita
  DeltaX ← -Fx/DFx; x ← x + DeltaX
  Fx ← f(x); DFx ← f'(x); { avaliar a função e sua derivada em x }
  Iter ← Iter + 1
  escreva Iter, x, DFx, Fx, DeltaX
  se (abs(DeltaX) ≤ Toler e abs(Fx) ≤ Toler) ou DFx = 0 ou Iter ≥ IterMax
    então interrompa
  fimse
  fimrepita
  Raiz ← x
  { teste de convergência }
  se abs(DeltaX) ≤ Toler e abs(Fx) ≤ Toler então
    CondErro ← 0
  senão
    CondErro ← 1
  fimse
finalgoritmo

```

Figura 6.20 Algoritmo do método de Newton.

(Ver significado da função abs na Tabela 1.1, na página 6.)

Exemplo 6.30 Determinar a maior raiz de $P(x) = x^4 + 2x^3 - 13x^2 - 14x + 24 = 0$ com $\epsilon \leq 10^{-5}$, utilizando o método de Newton.

Pelo Exemplo 6.11 e Figura 6.5(b), sabe-se que $\xi \in [2, 4]$, $f(2) < 0$ e $f(4) > 0$. As derivadas são $P'(x) = 4x^3 + 6x^2 - 26x - 14$ e $P''(x) = 12x^2 + 12x - 26 > 0$, $2 \leq x \leq 4$. Assim, $x_0 = 4$, pois $P(4)P''(4) > 0$.

% Os parâmetros de entrada

x0 = 4

Toler = 1.0000e-05

IterMax = 100

% produzem os resultados

Calculo de raiz de equação pelo método de Newton

| iter | x | DFx | Fx | Delta_x |
|------|---------|-------------|-------------|--------------|
| 0 | 4.00000 | 2.34000e+02 | 1.44000e+02 | |
| 1 | 3.38462 | 1.21825e+02 | 3.64693e+01 | -6.15385e-01 |
| 2 | 3.08526 | 8.03682e+01 | 6.40563e+00 | -2.99358e-01 |
| 3 | 3.00555 | 7.06567e+01 | 3.90611e-01 | -7.97036e-02 |
| 4 | 3.00003 | 7.00030e+01 | 1.80793e-03 | -5.52830e-03 |
| 5 | 3.00000 | 7.00000e+01 | 3.93537e-08 | -2.58264e-05 |
| 6 | 3.00000 | 7.00000e+01 | 1.42109e-14 | -5.62196e-10 |

Raiz = 3.00000

Iter = 6

CondErro = 0

A raiz da equação é $\xi = x_6 = 3$.

Exemplo 6.31 Calcular o ponto de inflexão ι da função $f(x) = 2x^3 + 3x^2 - e^x + 3$ com $\epsilon \leq 10^{-5}$ pelo método de Newton, usando o algoritmo mostrado na Figura 6.20.

A condição de ponto de inflexão de uma função é que a derivada segunda se anule. Deste modo, deve-se achar uma raiz de $f''(x) = 0$

$$g(x) = f''(x) = 12x - e^x + 6 = 0.$$

Pela Figura 6.21(a) verifica-se que $\iota \in [-2, 1]$. As derivadas são $g'(x) = 12 - e^x$ e $g''(x) = -e^x < 0 \forall x$. Como $g(-2) \approx -1,1353 < 0$ e $g(1) \approx 5,2817 > 0$, então $x_0 = -2$ porque $g(-2)g''(-2) > 0$.

% Os parâmetros de entrada

x0 = -2

Toler = 1.0000e-05

IterMax = 100

% produzem os resultados

Calculo de raiz de equação pelo método de Newton

| iter | x | DFx | Fx | Delta_x |
|------|----------|--------------|--------------|-------------|
| 0 | -2.00000 | 1.18647e+001 | -1.81353e+01 | |
| 1 | -0.47148 | 1.13759e+001 | -2.81878e-01 | 1.52852e+00 |
| 2 | -0.44671 | 1.13603e+001 | -1.93175e-04 | 2.47785e-02 |
| 3 | -0.44669 | 1.13603e+001 | -9.24905e-11 | 1.70045e-05 |
| 4 | -0.44669 | 1.13603e+001 | 0.00000e+00 | 8.14158e-12 |

Raiz = -0.44669

Iter = 4

CondErro = 0

O ponto de inflexão é $\iota \approx x_4 = -0,44669$.

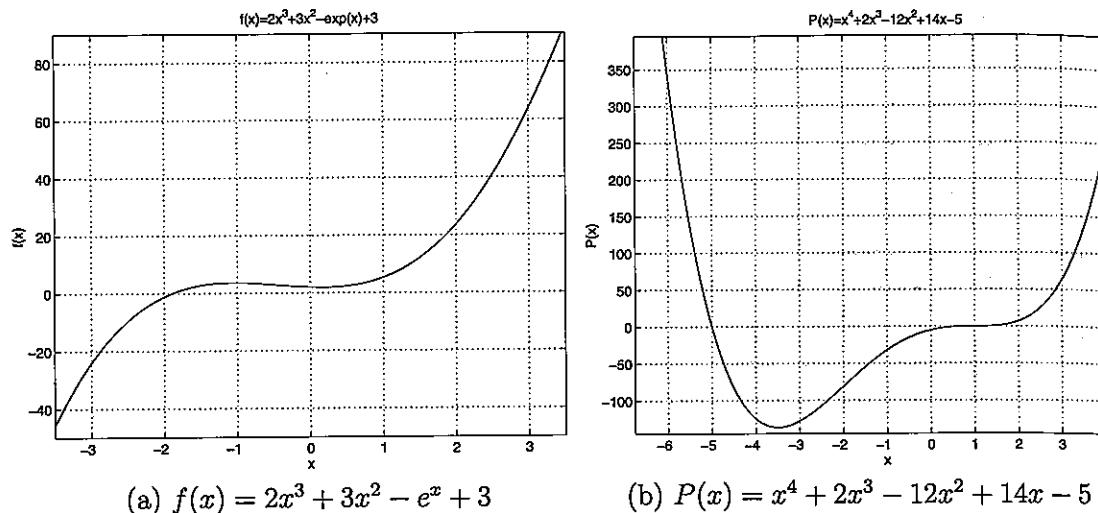


Figura 6.21 Esboços para isolamento de raízes.

Para determinar a ordem de convergência do método de Newton, considere (6.16)

$$x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

que em vista do erro da k -ésima iteração (6.5) torna-se

$$\epsilon_{k+1} = \epsilon_k - \frac{f(x_k)}{f'(x_k)}. \quad (6.18)$$

Expandindo $f(x_k)$ em série de Taylor em torno da raiz ξ , tem-se

$$f(x_k) = f(\xi) + \epsilon_k f'(\xi) + \epsilon_k^2 \frac{f''(\xi)}{2} + \dots,$$

$$f'(x_k) = f'(\xi) + \epsilon_k f''(\xi) + \dots$$

Substituindo as duas expressões acima em (6.18)

$$\epsilon_{k+1} = \epsilon_k - \frac{\epsilon_k f'(\xi) + \epsilon_k^2 \frac{f''(\xi)}{2} + \dots}{f'(\xi) + \epsilon_k f''(\xi) + \dots}$$

$$\epsilon_{k+1} = \frac{\epsilon_k f'(\xi) + \epsilon_k^2 f''(\xi) + \dots - \epsilon_k f'(\xi) - \epsilon_k^2 \frac{f''(\xi)}{2} - \dots}{f'(\xi) + \epsilon_k f''(\xi) + \dots}$$

$$|\epsilon_{k+1}| \approx \frac{|f''(\xi)|}{2|f'(\xi)|} |\epsilon_k|^2.$$

Conseqüentemente, em vista de (6.6), o método de Newton tem convergência quadrática. Isso significa que, nas proximidades da raiz, o número de dígitos corretos da estimativa da raiz praticamente dobra a cada iteração.

6.5.2 Método de Schröder

O método de Newton apresenta uma convergência apenas linear quando uma raiz tem multiplicidade $m > 1$, isto porque a medida que $f(x_k) \rightarrow 0$, o denominador $f'(x_k) \rightarrow 0$ (ver Exemplo 6.4). Uma modificação simples proposta por Schröder [35] permite o cálculo de uma raiz de multiplicidade m , mantendo a convergência quadrática, utilizando a fórmula de recorrência

$$x_{k+1} = x_k - m \frac{f(x_k)}{f'(x_k)}, \quad k = 0, 1, 2, \dots \quad (6.19)$$

O algoritmo do método de Schröder é basicamente igual ao de Newton mostrado na Figura 6.20, mas com um parâmetro extra m para definir a multiplicidade, o qual é usado no comando

$$\Delta x \leftarrow -m * Fx / DFx$$

substituto de

$$\Delta x \leftarrow -Fx / DFx.$$

Exemplo 6.32 Calcular a raiz de $P(x) = x^4 + 2x^3 - 12x^2 + 14x - 5 = 0$ de multiplicidade $m = 3$, com tolerância $\varepsilon \leq 10^{-5}$ (ver Exemplo 6.4), pelo método de Schröder usando um algoritmo adaptado da Figura 6.20.

Pela Figura 6.21(b), verifica-se que $\xi \in [0, 5, 1, 5]$. As derivadas são $P'(x) = 4x^3 + 6x^2 - 24x + 14$ e $P''(x) = 12x^2 + 12x - 24 > 0 \forall x > 1$. Conseqüentemente, para satisfazer ao Teorema 6.6, $x_0 = 1,5$ porque $P(1,5)P''(1,5) > 0$.

% Os parametros de entrada

$m = 3$

$x_0 = 1.5$

Toler = 1.0000e-05

IterMax = 100

% produzem os resultados

| Calculo de raiz de equacao pelo metodo de Schroder | | | | |
|--|---------|-------------|-------------|--------------|
| iter | x | DFx | Fx | Delta_x |
| 0 | 1.50000 | 5.00000e+00 | 8.12500e-01 | |
| 1 | 1.01250 | 2.82031e-03 | 1.17432e-05 | -4.87500e-01 |
| 2 | 1.00001 | 1.34883e-09 | 4.44089e-15 | -1.24913e-02 |
| 3 | 1.00000 | 2.68212e-11 | 0.00000e+00 | -9.87718e-06 |

Raiz = 1.00000

Iter = 3

CondErro = 0

Portanto, $\xi = x_3 = 1$. O método de Newton gasta 26 iterações para calcular esta raiz com a mesma tolerância $\varepsilon \leq 10^{-5}$. ■

6.6 Comparação dos métodos para cálculo de raízes

Um estudo comparativo do desempenho de métodos utilizando uma série de equações está longe de ser perfeito, justamente pela dependência do resultado na escolha dessas equações. Por isso, a determinação da ordem de convergência é mais adequada, pois não é baseada em nenhum empirismo.

Mesmo assim, é interessante verificar o desempenho dos métodos estudados neste capítulo. Para tal foram escolhidas cinco equações, das quais duas com raiz de multiplicidade $m > 1$, e o intervalo que isola a raiz desejada

$$f_1(x) = 2x^4 + 4x^3 + 3x^2 - 10x - 15 = 0, \xi \in [0, 3].$$

$$f_2(x) = x^5 - 2x^4 - 9x^3 + 22x^2 + 4x - 24 = 0, \xi \in [0, 5], \text{ com } m = 3.$$

$$f_3(x) = 5x^3 + x^2 - e^{1-2x} + \cos(x) + 20 = 0, \xi \in [-5, 5].$$

$$f_4(x) = \sin(x)x + 4 = 0, \xi \in [1, 5].$$

$$f_5(x) = (x - 3)^5 \log_e(x) = 0, \xi \in [2, 5], \text{ com } m = 5.$$

Para todos os métodos foram utilizados o mesmo número máximo de iterações (500), tolerância ($\varepsilon = 10^{-10}$) e critério de parada, com exceção do método de van Wijngaarden-Dekker-Brent que usa um critério ligeiramente diferente. Para o método de Newton, x_0 foi escolhido como o ponto médio do intervalo dado, sem considerar o Teorema 6.6.

Os resultados estão apresentados nas Tabelas 6.1–6.5, nas quais Iter é o número de iterações gastas, Erro é a condição de erro (se $CondErro = 1$ ou Raiz está fora do intervalo dado, então Erro é sim) e t_{rel} é o tempo relativo ao tempo gasto pelo método da bissecção.

Tabela 6.1 Comparação de métodos usando $f_1(x) = 2x^4 + 4x^3 + 3x^2 - 10x - 15$.

| Método | Raiz | Iter | Erro | t_{rel} |
|---------------------|----------|------|------|-----------|
| bisseção | 1,49288 | 37 | | 1,00 |
| secante | -1,30038 | 8 | sim | 0,28 |
| <i>regula falsi</i> | 1,49288 | 77 | | 2,06 |
| pégaso | 1,49288 | 10 | | 0,35 |
| Muller | 1,49288 | 4 | | 0,25 |
| W-D-Brent | 1,49288 | 9 | | 0,63 |
| Newton | 1,49288 | 4 | | 0,20 |

O método da bissecção mostrou sua robustez, pois não falhou apesar de não ser o mais eficiente. A secante, embora seja rápida, encontrou uma raiz fora do intervalo dado (Tabela 6.1). A *regula falsi* apresentou uma convergência muito lenta e falhou três vezes. O pégaso, além de

Tabela 6.2 Comparação de métodos usando $f_2(x) = x^5 - 2x^4 - 9x^3 + 22x^2 + 4x - 24$.

| Método | Raiz | Iter | Erro | t_{rel} |
|---------------------|---------|------|------|-----------|
| bisseção | 1,99999 | 35 | | 1,00 |
| secante | 2,00000 | 47 | | 1,36 |
| <i>regula falsi</i> | 1,82374 | 500 | sim | 13,42 |
| pégaso | 1,99999 | 60 | | 1,76 |
| Muller | 2,00001 | 500 | sim | 17,85 |
| W-D-Brent | 2,00001 | 57 | | 3,59 |
| Newton | 2,00001 | 37 | | 1,55 |
| Schröder | 2,00000 | 4 | sim | 0,23 |

Tabela 6.3 Comparação de métodos usando $f_3(x) = 5x^3 + x^2 - e^{1-2x} + \cos(x) + 20$.

| Método | Raiz | Iter | Erro | t_{rel} |
|---------------------|----------|------|------|-----------|
| bisseção | -0,92956 | 41 | | 1,00 |
| secante | -0,92956 | 21 | | 0,56 |
| <i>regula falsi</i> | 0,69661 | 500 | sim | 12,33 |
| pégaso | -0,92956 | 19 | | 0,57 |
| Muller | -0,92956 | 32 | | 1,16 |
| W-D-Brent | -0,92956 | 8 | | 0,51 |
| Newton | -0,92956 | 11 | | 0,48 |

Tabela 6.4 Comparação de métodos usando $f_4(x) = \sin(x)x + 4$.

| Método | Raiz | Iter | Erro | t_{rel} |
|---------------------|---------|------|------|-----------|
| bisseção | 4,32324 | 36 | | 1,00 |
| secante | 4,32324 | 7 | | 0,27 |
| <i>regula falsi</i> | 4,32324 | 9 | | 0,32 |
| pégaso | 4,32324 | 7 | | 0,28 |
| Muller | 4,32324 | 6 | | 0,35 |
| W-D-Brent | 4,32324 | 7 | | 0,57 |
| Newton | 4,32324 | 6 | | 0,30 |

ser robusto, foi competitivo com relação ao sofisticado método de van Wijngaarden-Dekker-Brent.

O método de Muller não foi robusto, embora eficiente, pois falhou nos casos onde a raiz possui multiplicidade (Tabelas 6.2 e 6.5). O método de van Wijngaarden-Dekker-Brent foi robusto, mas também foi menos eficiente na presença de multiplicidade.

O método de Schröder é uma efetiva modificação do método de Newton para evitar problemas com raízes de multiplicidade, conforme mostram os resultados das Tabelas 6.2 e 6.5, pelas quais ele foi, sem dúvida, o mais eficiente.

Tabela 6.5 Comparação de métodos usando $f_5(x) = (x - 3)^5 \log_e(x)$.

| Método | Raiz | Iter | Erro | t_{rel} |
|--------------|---------|------|------|-----------|
| bisseção | 3,00000 | 34 | | 1,00 |
| secante | 3,00000 | 137 | | 3,90 |
| regula falsi | 2,67570 | 500 | sim | 13,89 |
| pégaso | 3,00000 | 187 | | 5,47 |
| Muller | 3,01289 | 500 | sim | 18,82 |
| W-D-Brent | 3,00000 | 80 | | 5,45 |
| Newton | 3,00000 | 95 | | 4,45 |
| Schröder | 3,00000 | 4 | | 0,26 |

6.7 Exemplos de aplicação

Será mostrado como o cálculo de raiz de equação pode ser aplicado para determinar a taxa de juros de um financiamento, estudo de grande valia para o consumidor, e achar o comprimento de um cabo suspenso que é um problema de interesse para as Engenharias Civil e Elétrica.

6.7.1 Juros de financiamento

Definição do problema

O preço à vista de uma mercadoria é R\$ 1.100,00. No entanto, ela pode ser financiada por dois planos

Plano 1: entrada de R\$ 100,00 e mais 6 prestações de R\$ 224,58.

Plano 2: sem entrada e 10 prestações de R\$ 163,19.

Qual dos dois planos de financiamento é melhor para o consumidor?

Modelagem matemática

O melhor plano será aquele que tiver a menor taxa de juros. Da Matemática Financeira tem-se que

$$\frac{1 - (1 + j)^{-p}}{j} = \frac{v - e}{m},$$

sendo j a taxa de juros, p o prazo, v o preço à vista, e a entrada e m a mensalidade. Rearranjando a equação acima

$$f(j) = \frac{1 - (1 + j)^{-p}}{j} - \frac{v - e}{m} = 0.$$

Assim, o problema de determinar a taxa de juros do financiamento recai em calcular a raiz de uma equação transcendente.

Solução numérica

Inicialmente, cada raiz é isolada para depois ser refinada por um dos métodos vistos nas seções anteriores. As equações para cada plano são

$$f_1(j) = \frac{1 - (1 + j)^{-6}}{j} - \frac{1.100 - 100}{224,58} = 0 \text{ e}$$

$$f_2(j) = \frac{1 - (1 + j)^{-10}}{j} - \frac{1.100 - 0}{163,19} = 0.$$

Os esboços das duas funções são apresentados na Figura 6.22, indicando que cada raiz está isolada no intervalo [0,05; 0,1].

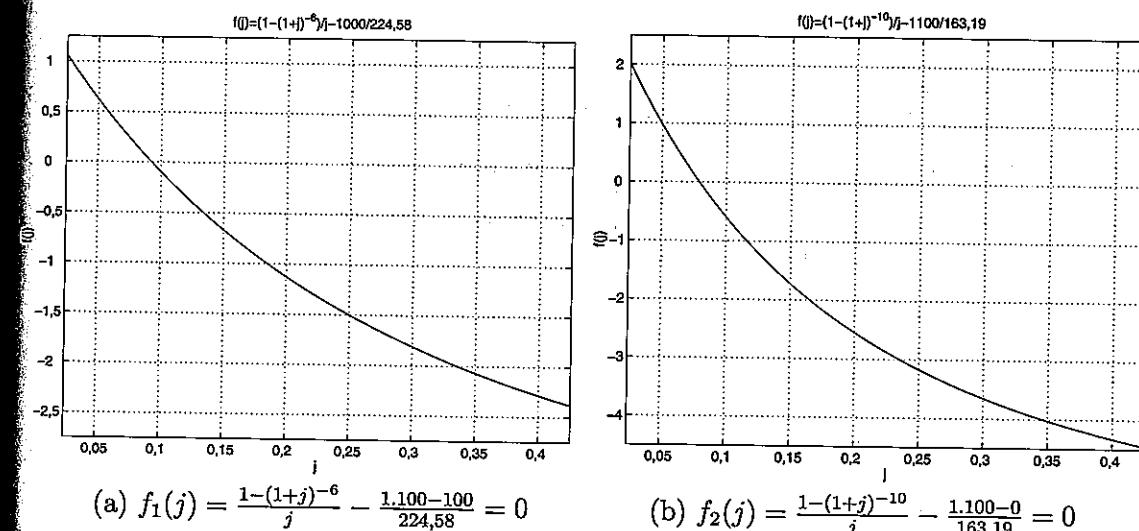


Figura 6.22 Isolamento de raízes para cálculo de juros.

Cada raiz pode agora ser refinada usando, por exemplo, o método pégaso implementado a partir do algoritmo da Figura 6.15, com $\epsilon \leq 10^{-5}$.

$$\text{Para o plano 1: } f_1(j) = \frac{1 - (1 + j)^{-6}}{j} - \frac{1.100 - 100}{224,58} = 0$$

```
% Os parametros de entrada
a = 0.0500
b = 0.1000
Toler = 1.0000e-05
IterMax = 100
```

```
% produzem os resultados
    Calculo de raiz de equacao pelo metodo pegaso
iter   a      Fa     b      Fb     x      Fx     Delta_x
  0  0.05000  0.62294  0.10000 -0.09750  0.09323 -9.758e-03 -6.766e-03
  1  0.05000  0.56626  0.09323 -0.00976  0.09250 -9.373e-05 -7.324e-04
  2  0.05000  0.56088  0.09250 -0.00009  0.09249  1.384e-07 -7.101e-06

Raiz    = 0.09249
Iter    = 2
CondErro = 0
```

A taxa de juros do financiamento pelo plano 1 é $j_1 = 9,25\%$.

$$\text{Para o plano 2: } f_2(j) = \frac{1 - (1 + j)^{-10}}{j} - \frac{1.100 - 0}{163,19} = 0$$

% Os parametros de entrada

```
a = 0.0500
b = 0.1000
Toler = 1.0000e-05
IterMax = 100
```

% produzem os resultados

```
    Calculo de raiz de equacao pelo metodo pegaso
iter   a      Fa     b      Fb     x      Fx     Delta_x
  0  0.05000  0.98113  0.10000 -0.59604  0.08110 -6.381e-02 -1.890e-02
  1  0.05000  0.88624  0.08110 -0.06381  0.07901 -6.079e-04 -2.089e-03
  2  0.05000  0.87788  0.07901 -0.00061  0.07899  3.996e-06 -2.008e-05
  3  0.07901 -0.00061  0.07899  0.00000  0.07899 -2.778e-10  1.311e-07
```

```
Raiz    = 0.07899
Iter    = 3
CondErro = 0
```

Para este plano, a taxa de juros é $j_2 = 7,90\%$.

Análise dos resultados

O total pago por cada plano é

Plano 1: R\$ 100,00 + 6 × R\$ 224,58 = R\$ 1.447,48.

Plano 2: 10 × R\$ 163,19 = R\$ 1.631,90.

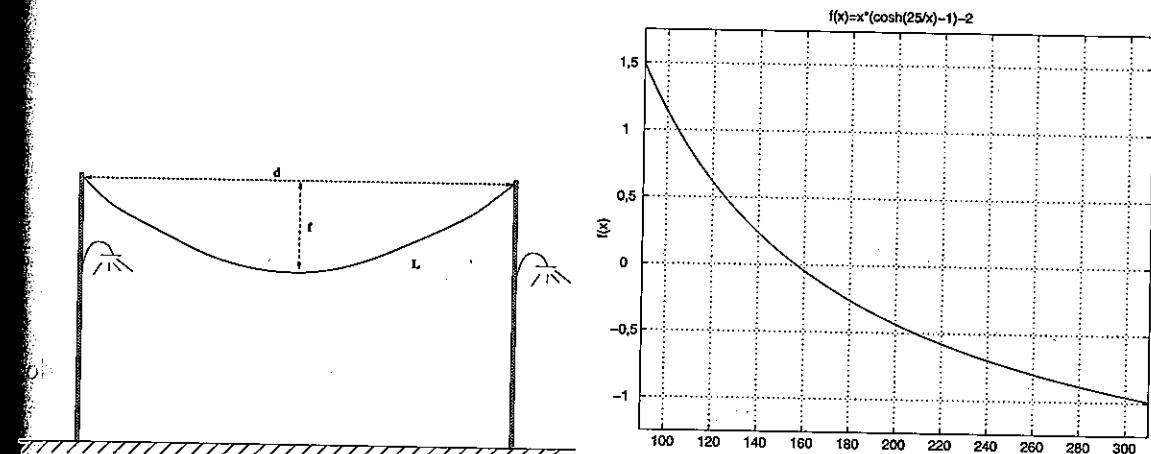
Em princípio, o plano 1 parece melhor, pois é desembolsada uma quantia menor. No entanto, isto é um engano porque a taxa de juros do plano 1 é maior. Assim, o plano 2 é melhor por possuir uma menor taxa de juros.

Sempre que possível, o consumidor deve optar pelo pagamento à vista, a menos que ele consiga um investimento que remunere o seu capital acima das taxas de juros praticadas pelas lojas.

6.7.2 Cabo suspenso

Definição do problema

Determinar o comprimento de um cabo suspenso entre dois pontos distantes 50 m que estão no mesmo nível e fazendo uma flecha de 2 m, conforme Figura 6.23(a).



(a) O problema do cabo suspenso.

(b) Isolamento da raiz.

Figura 6.23 Comprimento de um cabo suspenso.

Modelagem matemática

O comprimento L é dado pela equação

$$L = 2\alpha \operatorname{senh}\left(\frac{d}{2\alpha}\right),$$

sendo d a distância entre os dois pontos e α a raiz da equação

$$g(x) = x \left(\cosh\left(\frac{d}{2x}\right) - 1 \right) - f = 0,$$

onde f é a flecha.

Solução numérica

Pela Figura 6.23(b), nota-se que a raiz α está isolada no intervalo $[140, 160]$. Utilizando o método pégaso, implementado a partir do algoritmo da Figura 6.15, obtém-se

```
% Os parametros de entrada
a = 140
b = 160
Toler = 1.0000e-05
IterMax = 100
```

```
% produzem os resultados
Calculo de raiz de equacao pelo metodo pegaso
iter   a      Fa     b      Fb     x      Fx     Delta_x
0 140.00000  0.23808 160.00000 -0.04290 156.94652 -4.662e-03 -3.053e+00
1 140.00000  0.21474 156.94652 -0.00466 156.58642 -5.405e-05 -3.601e-01
2 140.00000  0.21228 156.58642 -0.00005 156.58219  1.055e-07 -4.222e-03
3 156.58642 -0.00005 156.58219  0.00000 156.58220 -2.828e-12  8.224e-06

Raiz    =156.58220
Iter    = 3
CondErro = 0
```

Sendo $\alpha = 156,58220$, então o comprimento do cabo é

$$L = 2\alpha \operatorname{senh} \left(\frac{d}{2\alpha} \right) = 2 \times 156,58220 \times \operatorname{senh} \left(\frac{50}{2 \times 156,58220} \right) \sim L = 50,21270.$$

Análise dos resultados

O comprimento do cabo suspenso entre dois pontos no mesmo nível distantes 50 m e fazendo uma flecha de 2 m é igual a 50,21 m.

6.8 Exercícios

Seção 6.1

6.1. Calcular os limites e o número de raízes reais de $P(x) = 2x^3 - 3x^2 - 6x + 5 = 0$.

6.2. Implementar, em qualquer linguagem de programação, o algoritmo para determinar os limites das raízes reais de uma equação algébrica mostrado na Figura 6.4.

6.3. Calcular os limites das raízes reais da equação polinomial do Exercício 6.1 utilizando o programa do Exercício 6.2.

6.4. Implementar o algoritmo para encontrar um intervalo onde uma função troca de sinal, descrito na Figura 6.6, utilizando qualquer linguagem de programação.

6.5. Isolar as duas raízes da equação transcendente $f(x) = e^{-x} + x^2 - 10 = 0$ usando o programa do Exercício 6.4.

Seção 6.2

Usando o método da bisseção com $\varepsilon \leq 0,001$, calcular pelo menos uma raiz de cada equação abaixo.

6.6. $f(x) = e^{2x} - 2x^3 - 5 = 0$.

6.7. $f(x) = 2x^3 - 5x^2 - x + 3 = 0$.

6.8. $f(x) = 5x^2 + \log_{10}(x+1) - 2 = 0$.

6.9. Implementar o algoritmo da Figura 6.10 em qualquer linguagem de programação.

6.10. Resolver os Exercícios 6.6–6.8 usando o programa do Exercício 6.9.

Seção 6.3

Achar pelo menos uma raiz positiva de cada equação abaixo com $\varepsilon \leq 10^{-4}$ usando os três métodos: secante, *regula falsi* e pégaso. Fazer a compa-

ração do número de iterações gasto por cada método.

6.11. $f(x) = 2x^3 - 5x^2 - 10x + 20 = 0$.

6.12. $f(x) = 5 \log_{10}(x) + 3x^4 - 7 = 0$.

6.13. $f(x) = 2^x + \cos(x)x^2 = 0$.

6.14. Implementar, em qualquer linguagem de programação, os algoritmos descritos nas Figuras 6.12, 6.13 e 6.15.

6.15. Resolver os Exercícios 6.11–6.13 utilizando os programas escritos no Exercício 6.14.

Seção 6.4

Achar pelo menos uma raiz de cada equação abaixo com $\varepsilon \leq 10^{-5}$ usando os métodos de Muller e de van Wijngaarden-Dekker-Brent e comparar o número de iterações gasto por cada método.

6.16. $f(x) = e^{-\cos(x)} + 2x^4 - x^2 - 5 = 0$.

6.17. $f(x) = 3x^x - \log_{10}(2+\sqrt{x}) + 3 \tan(x) - 4 = 0$.

6.18. $f(x) = 3x^4 + 2x^3 + 4x^2 + 25x - 30 = 0$.

6.19. Implementar, usando qualquer linguagem de programação, os algoritmos das Figuras 6.17 e 6.18.

6.20. Resolver os Exercícios 6.16–6.18 utilizando os programas do Exercício 6.19.

Seção 6.5

Achar pelo menos uma raiz positiva de cada equação abaixo com $\varepsilon \leq 10^{-5}$ pelos métodos de Newton e de Schröder e comparar o número de iterações gastas por cada método.

6.21. $f(x) = 4x^3 + x + \cos(x) - 10 = 0$.

6.22. $f(x) = x^4 - 2x^3 + 2x - 1 = 0$.

6.23. $f(x) = (x - 2)(e^{x-2} - 1) = 0$.

6.24. Implementar o método de Newton descrito na Figura 6.20 e o método de Schröder em qualquer linguagem de programação.

6.25. Resolver os Exercícios 6.21–6.23 usando os programas do Exercício 6.24.

Seção 6.6

Usando os algoritmos implementados, anteriormente, dos métodos bisseção, secante, *regula falsi*, pégaso, Muller, van Wijngaarden-Dekker-Brent, Newton e Schröder, comparar o desempenho destes métodos para o cálculo de uma raiz das equações abaixo com $\text{Toler} = 10^{-10}$ e $\text{IterMax} = 500$.

6.26. $f(x) = 5x^3 - 2x^2 + 8x - 10 = 0$, $\xi \in [0, 2]$.

6.27. $f(x) = 2x^3 + 5x^2 - \sin(x) - 30 = 0$, $\xi \in [1, 4]$.

6.28. $f(x) = e^{-x^2} \cos(x) = 0$, $\xi \in [-1, 2]$.

6.29. $f(x) = (x+1)(x-1)(x-3)^5 = 0$, $\xi \in [2, 5]$.

6.30. $f(x) = (x+2)^3 \sqrt{x^2 + 1} = 0$, $\xi \in [-3, 0]$.

Gerais

6.31. Sendo $\xi_1, \xi_2, \dots, \xi_n$ as raízes da equação algébrica $P(x) = c_n x^n + c_{n-1} x^{n-1} + \dots + c_2 x^2 + c_1 x + c_0 = 0$, mostrar que $1/\xi_1, 1/\xi_2, \dots, 1/\xi_n$ são as raízes de $P_1(x) = x^n P(1/x) = 0$.

6.32. Sendo $\xi_1, \xi_2, \dots, \xi_n$ as raízes da equação algébrica $P(x) = c_n x^n + c_{n-1} x^{n-1} + \dots + c_2 x^2 + c_1 x + c_0 = 0$, mostrar que $-\xi_1, -\xi_2, \dots, -\xi_n$ são as raízes de $P_2(x) = P(-x) = 0$.

6.33. Sendo $\xi_1, \xi_2, \dots, \xi_n$ as raízes da equação algébrica $P(x) = c_n x^n + c_{n-1} x^{n-1} + \dots + c_2 x^2 +$

$c_1 x + c_0 = 0$, mostrar que $-1/\xi_1, -1/\xi_2, \dots, -1/\xi_n$ são as raízes de $P_3(x) = x^n P(-1/x) = 0$.

6.34. Achar os zeros do polinômio característico da matriz B do Exercício 2.43.

6.35. Determinar os autovalores da matriz

$$M = \begin{bmatrix} 4 & -2 & 3 \\ -2 & 5 & -2 \\ 3 & -2 & 8 \end{bmatrix}.$$

6.36. Calcular os zeros do polinômio de Legende de grau $n = 3$ e comparar com os valores de t_i compilados na Tabela 5.3, na página 237.

6.37. Demonstrar que a raiz $\sqrt[p]{a}$, $a > 0$ pode ser calculada pela fórmula de recorrência

$$x_{k+1} = \frac{1}{p} \left((p-1)x_k + \frac{a}{x_k^{p-1}} \right),$$

com $x_0 > 0$.

6.38. Determinar o volume molar \bar{V} do CO_2 a $t = 60^\circ\text{C}$ e $P = 25$ atm usando a equação de estado de van der Waals $\left(P + \frac{a}{V^2}\right)(\bar{V} - b) = RT$, sabendo que para o dióxido de carbono $a = 3,6 \text{ atm} \times 10^2 \times \text{mol}^{-2}$ e $b = 0,0428 \text{ l} \times \text{mol}^{-1}$. Deve ser lembrado que $T = t + 273,15$ e $R = 0,08205 \text{ atm} \times \text{l} \times \text{mol}^{-1} \times \text{K}^{-1}$.

6.39. O pH de soluções diluídas de um ácido fraco é a raiz positiva da equação $[\text{H}_3\text{O}^+]^3 + K_a [\text{H}_3\text{O}^+]^2 - (K_a C_a + K_w)[\text{H}_3\text{O}^+] - K_w K_a = 0$, sendo $pH = -\log_{10}[\text{H}_3\text{O}^+]$, K_a a constante de dissociação do ácido, C_a a concentração do ácido e K_w o produto iônico da água. Calcular o pH de uma solução de ácido bórico a 25°C , sabendo que $K_a = 6,5 \times 10^{-10} \text{ M}$, $C_a = 2,0 \times 10^{-5} \text{ M}$ e $K_w = 1,0 \times 10^{-14} \text{ M}$.

6.40. O preço à vista de uma mercadoria é R\$ 3.120,00, mas pode ser financiada com uma entrada de R\$ 910,52 e mais 12 prestações mensais de R\$ 260,00. Calcular a taxa de juros.

Capítulo 7

Equações diferenciais ordinárias

As equações diferenciais ordinárias (EDO) são ferramentas fundamentais para a modelagem matemática de vários fenômenos físicos, químicos, biológicos etc., quando estes fenômenos são descritos em termos de taxa de variação. Por exemplo, a seguinte EDO descreve a taxa de variação da corrente i em função do tempo t em um circuito RL

$$\frac{di(t)}{dt} = \frac{V - i(t)R}{L}, \quad (7.1)$$

onde V é a tensão entre dois pontos do circuito, R é a resistência e L é a indutância. Este é um exemplo de equação diferencial ordinária de primeira ordem. Ela é ordinária, visto que a corrente i é função apenas de uma variável independente, o tempo t ; caso contrário, se a função fosse definida em termos de duas ou mais variáveis, ter-se-ia uma equação diferencial parcial como, por exemplo, a equação de Laplace

$$\frac{\partial^2 u(x, y)}{\partial x^2} + \frac{\partial^2 u(x, y)}{\partial y^2} = 0.$$

Além disso, a EDO (7.1) é de primeira ordem, pois a derivada de maior ordem, $di(t)/dt$, é de ordem 1. Quando a equação contiver uma derivada de ordem n , ela é dita EDO de ordem n . Assim,

$$L \frac{d^2 i(t)}{dt^2} + R \frac{di(t)}{dt} + \frac{1}{C} i(t) = \frac{dV(t)}{dt}$$

é uma EDO de segunda ordem, sendo C a capacidade do circuito.

A solução de uma EDO é uma função que satisfaz à equação diferencial e que também satisfaz a certas condições iniciais na função. Ao resolver uma EDO, analiticamente, encontra-se uma solução geral contendo constantes arbitrárias e, então, determinam-se essas constantes de modo que a expressão combine com as condições iniciais.

7.1 Solução numérica de EDO

Os métodos analíticos são restritos apenas a algumas formas especiais de função, visto que nem toda EDO tem solução analítica. Os métodos numéricos não possuem tal limitação, contudo, a solução numérica é obtida como uma tabela de valores da função em vários valores da variável independente, em vez de uma relação funcional como na solução analítica. Deste modo, praticamente qualquer EDO pode ser resolvida numericamente; no entanto, se as condições iniciais forem alteradas, toda a tabela deve ser recalculada.

Neste capítulo, serão vistos métodos numéricos para a solução de equações diferenciais ordinárias sujeitas às condições iniciais. Para um estudo mais abrangente sobre EDO, pode ser consultado o excelente livro de John Denholm Lambert [31].

7.1.1 Problema de valor inicial

O problema de valor inicial (PVI), de primeira ordem, apresenta a forma

$$\begin{cases} y' = f(x, y), \\ y(a) = \eta, \\ a \leq x \leq b \text{ e } -\infty \leq y \leq \infty. \end{cases} \quad (7.2)$$

A solução do PVI é uma função $y = y(x)$ contínua e diferenciável que satisfaz a (7.2). O Teorema 7.1 [31, Teorema 1.1] estabelece as condições suficientes para a existência de uma única solução do PVI.

Teorema 7.1 (Lipschitz) Seja $f(x, y)$ uma função definida e contínua para todo (x, y) na região D definida por $a \leq x \leq b$ e $-\infty \leq y \leq \infty$, sendo a e b números finitos, e seja uma constante L tal que

$$\|f(x, y) - f(x, y^*)\| \leq L \|y - y^*\| \quad (7.3)$$

seja válida para todo (x, y) , $(x, y^*) \in D$. Então, para algum η , existe uma única solução $y(x)$ do PVI (7.2), onde $y(x)$ é contínua e diferenciável para todo $(x, y) \in D$.

A inequação (7.3) é conhecida como uma condição de Lipschitz e L como uma constante de Lipschitz.

Serão apresentados métodos numéricos para calcular uma aproximação y_i da solução exata $y(x_i)$ do PVI nos pontos

$$x_i = a + ih, \quad h = \frac{b-a}{m}, \quad i = 0, 1, 2, \dots, m,$$

onde m é o número de subintervalos de $[a, b]$ e h é o incremento ou passo. Deste modo, a solução numérica do PVI será uma tabela contendo os pares (x_i, y_i) sendo que $y_i \approx y(x_i)$.

7.1.2 Método de Euler

Seja uma expansão da solução exata $y(x)$, em série de Taylor, em torno do valor inicial x_0

$$y(x_0 + h) = y(x_0) + hy'(x_0) + \frac{h^2}{2}y''(x_0) + \frac{h^3}{6}y'''(x_0) + \dots$$

Truncando a série após o termo de derivada primeira, sendo $x_1 = x_0 + h$ e y_1 uma aproximação de $y(x_1)$ e sabendo que $y' = f(x, y)$, tem-se

$$y_1 = y_0 + hf(x_0, y_0).$$

As sucessivas aproximações y_i de $y(x_i)$ podem, então, ser obtidas pela fórmula de recorrência

$$y_{i+1} = y_i + hf(x_i, y_i), \quad (7.4)$$

que é conhecida como método de Euler¹.

Um algoritmo para a solução de problema de valor inicial pelo método de Euler é mostrado na Figura 7.1. Os parâmetros de entrada são o limite inferior a , o limite superior b , o número de subintervalos m e o valor inicial y_0 . A função derivada $y' = f(x, y)$ deve ser especificada de acordo com a linguagem de programação adotada. Os parâmetros de saída são as abscissas $VetX$ e as soluções $VetY$ do PVI.

Algoritmo Euler

{ Objetivo: Resolver um PVI pelo método de Euler }

parâmetros de entrada a, b, m, y_0

{ limite inferior, limite superior, número de subintervalos e valor inicial }

parâmetros de saída $VetX, VetY$ { abscissas e solução do PVI }

$h \leftarrow (b-a)/m; x \leftarrow a; y \leftarrow y_0$

$Fxy \leftarrow f(x, y)$ { avaliar $f(x, y)$ em $x = x_0$ e $y = y_0$ }

$VetX(1) \leftarrow x; VetY(1) \leftarrow y$

para $i \leftarrow 1$ até m faça

$x \leftarrow a + i * h$

$y \leftarrow y + h * Fxy$

$Fxy \leftarrow f(x, y)$ { avaliar $f(x, y)$ em $x = x_i$ e $y = y_i$ }

escreva i, x, y, Fxy

$VetX(i+1) \leftarrow x; VetY(i+1) \leftarrow y$

fimpara

finalgoritmo

Figura 7.1 Método de Euler para a solução de problema de valor inicial.

¹L. Euler propôs este método em 1768.

Exemplo 7.1 Calcular a solução do PVI

$$y' = x - 2y + 1, \text{ com } y(0) = 1, \quad (7.5)$$

no intervalo $[0, 1]$, com $m = 10$ subintervalos, utilizando o algoritmo da Figura 7.1.

% Os parametros de entrada

a = 0

b = 1

m = 10

y0 = 1

% produzem os resultados

 Método de Euler

| i | x | y | f(x,y) |
|----|---------|---------|----------|
| 0 | 0.00000 | 1.00000 | -1.00000 |
| 1 | 0.10000 | 0.90000 | -0.70000 |
| 2 | 0.20000 | 0.83000 | -0.46000 |
| 3 | 0.30000 | 0.78400 | -0.26800 |
| 4 | 0.40000 | 0.75720 | -0.11440 |
| 5 | 0.50000 | 0.74576 | 0.00848 |
| 6 | 0.60000 | 0.74661 | 0.10678 |
| 7 | 0.70000 | 0.75729 | 0.18543 |
| 8 | 0.80000 | 0.77583 | 0.24834 |
| 9 | 0.90000 | 0.80066 | 0.29867 |
| 10 | 1.00000 | 0.83053 | 0.33894 |

A solução exata do PVI (7.5) é

$$y(x) = \frac{1}{4}(3e^{-2x} + 2x + 1). \quad (7.6)$$

A Tabela 7.1(a) mostra a diferença entre o valor aproximado y_i dado pelo método de Euler com $m = 10$ subintervalos ($h = (1 - 0)/10 = 0,1$) e a solução exata $y(x_i)$ dada por (7.6), enquanto a Tabela 7.1(b) faz a mesma comparação usando $m = 100$ subintervalos ($h = (1 - 0)/100 = 0,01$).

O método de Euler com $h = 0,1$ só forneceu uma decimal exata para o PVI (7.5). No entanto, a redução do passo h para $0,01$ melhorou a solução numérica do PVI. Isso mostra que a exatidão da solução será melhorada quando o valor do passo for reduzido.

7.1.3 Definições

Definição 7.1 (Passo simples) Um método será de passo simples quando a aproximação y_{i+1} for calculada a partir somente do valor y_i do passo anterior. Sendo ϕ a função incremento, um método de passo simples é definido na forma

$$y_{i+1} = y_i + h\phi(x_i, y_i; h).$$

Definição 7.2 (Passo múltiplo) Sejam os p valores $y_i, y_{i-1}, y_{i-2}, \dots, y_{i-p+1}$ previamente calculados por algum método. Um método é de passo múltiplo se estes p valores $y_i, y_{i-1}, y_{i-2}, \dots, y_{i-p+1}$ forem utilizados para calcular y_{i+1} , para $i = p - 1, p, p + 1, \dots, m - 1$.

Tabela 7.1 Comparação da solução pelo método de Euler.

| i | x_i | y_i | $ y_i - y(x_i) $ | i | x_i | y_i | $ y_i - y(x_i) $ |
|----|-------|---------|------------------|-----|-------|---------|------------------|
| 0 | 0,0 | 1,00000 | 0,00000 | 0 | 0,0 | 1,00000 | 0,00000 |
| 1 | 0,1 | 0,90000 | 0,01400 | 10 | 0,1 | 0,9128 | 0,00120 |
| 2 | 0,2 | 0,83000 | 0,02270 | 20 | 0,2 | 0,8507 | 0,00200 |
| 3 | 0,3 | 0,78400 | 0,02760 | 30 | 0,3 | 0,8091 | 0,00250 |
| 4 | 0,4 | 0,75720 | 0,02980 | 40 | 0,4 | 0,7843 | 0,00270 |
| 5 | 0,5 | 0,7458 | 0,03010 | 50 | 0,5 | 0,7731 | 0,00280 |
| 6 | 0,6 | 0,7466 | 0,02930 | 60 | 0,6 | 0,7732 | 0,00270 |
| 7 | 0,7 | 0,7573 | 0,02770 | 70 | 0,7 | 0,7823 | 0,00260 |
| 8 | 0,8 | 0,7758 | 0,02560 | 80 | 0,8 | 0,7990 | 0,00240 |
| 9 | 0,9 | 0,8007 | 0,02330 | 90 | 0,9 | 0,8217 | 0,00220 |
| 10 | 1,0 | 0,8305 | 0,02100 | 100 | 1,0 | 0,8495 | 0,00200 |

(a) $m = 10 \rightarrow h = 0,1$.

(b) $m = 100 \rightarrow h = 0,01$.

Definição 7.3 (Erro local) Supondo que o valor calculado por um método de passo k seja exato, isto é, $y_{i+j} = y(x_{i+j})$ para $j = 0, 1, \dots, k - 1$, então o erro local em x_{i+k} é definido por

$$e_{i+k} = y(x_{i+k}) - y_{i+k}.$$

Definição 7.4 (Ordem) Um método de passo simples terá ordem q se a função incremento ϕ for tal que

$$y(x + h) = y(x) + h\phi(x, y; h) + O(h^{q+1}).$$

Definição 7.5 (Consistência) Um método numérico é dito consistente com o PVI (7.2) se a sua ordem $q \geq 1$.

Definição 7.6 (Convergência) Um método de passo k é convergente se, para o PVI (7.2),

$$\lim_{h \rightarrow 0} y_i = y(x_i), \text{ se } ih = x - a$$

é válido para todo $x \in [a, b]$, e os valores iniciais são tais que

$$\lim_{h \rightarrow 0} y_j(h) = \eta, \text{ se } j = 0, 1, \dots, k - 1.$$

A consistência significa que a solução numérica corresponde à solução do PVI. A consistência de um método limita a magnitude do erro local cometido em cada passo e a estabilidade controla a propagação do erro durante os cálculos. Um método é convergente se ele for consistente e estável.

Por (7.4) e pelas Definições 7.1 e 7.4, o método de Euler é de passo simples e tem ordem 1.

7.2 Métodos de Runge-Kutta

Pela Tabela 7.1, é fácil notar que a exatidão dos resultados pode ser melhorada se o passo h for reduzido. No entanto, se a exatidão requerida for elevada, esta metodologia pode acarretar um grande esforço computacional. Uma melhor exatidão pode ser obtida mais eficientemente por uma formulação denominada métodos de Runge-Kutta². Eles são métodos de passo simples de acordo com a Definição 7.1. Os chamados métodos explícitos de s estágios apresentam a forma geral

$$y_{i+1} = y_i + h\phi(x_i, y_i; h), \text{ onde } \phi(x, y; h) = b_1 k_1 + b_2 k_2 + \dots + b_s k_s, \text{ com} \quad (7.7)$$

$$k_1 = f(x, y),$$

$$k_2 = f(x + c_2 h, y + a_{21} h k_1),$$

$$k_3 = f(x + c_3 h, y + h(a_{31} k_1 + a_{32} k_2)),$$

...

$$k_s = f(x + c_s h, y + h(a_{s1} k_1 + a_{s2} k_2 + \dots + a_{s,s-1} k_{s-1})),$$

sendo a , b e c constantes definidas para cada método particular. Usualmente, essas constantes são exibidas na notação de Butcher, segundo a Tabela 7.2.

Tabela 7.2 Constantes do método de Runge-Kutta na notação de Butcher.

| | | | | |
|-------|----------|----------|-----|-------------|
| 0 | | | | |
| c_2 | a_{21} | | | |
| c_3 | a_{31} | a_{32} | | |
| : | : | : | .. | |
| c_s | a_{s1} | a_{s2} | ... | $a_{s,s-1}$ |
| | b_1 | b_2 | ... | b_{s-1} |
| | | | | b_s |

Os métodos de Runge-Kutta podem ser classificados de acordo com a sua ordem (Definição 7.4).

7.2.1 Métodos de segunda ordem

Seja a expansão em série de Taylor, na qual as derivadas em y são escritas em termos de f , a partir de $dy/dx = f(x, y)$,

$$y_{i+1} = y_i + hf(x_i, y_i) + \frac{h^2}{2}f'(x_i, y_i) + \dots$$

Como

$$f'(x, y) \equiv \frac{df}{dx} = \frac{\partial f}{\partial x} \frac{dx}{dx} + \frac{\partial f}{\partial y} \frac{dy}{dx} \rightarrow f'(x, y) = \frac{\partial f}{\partial x} + f \frac{\partial f}{\partial y},$$

²C. D. T. Runge desenvolveu o primeiro método em 1895 e M. W. Kutta elaborou a formulação geral em 1901.

então, simplificando a notação de modo que $f_i = f(x_i, y_i)$, sendo $\frac{\partial f_i}{\partial x} = \frac{\partial f}{\partial x}(x_i, y_i)$ e $\frac{\partial f_i}{\partial y} = \frac{\partial f}{\partial y}(x_i, y_i)$, tem-se

$$y_{i+1} = y_i + hf_i + h^2 \left(\frac{1}{2} \frac{\partial f_i}{\partial x} + \frac{1}{2} f_i \frac{\partial f_i}{\partial y} \right). \quad (7.8)$$

Por outro lado, (7.7) pode ser escrita em termos de k_1 e k_2 ,

$$y_{i+1} = y_i + b_1 h f(x_i, y_i) + b_2 h f(x_i + c_2 h, y_i + a_{21} h f(x_i, y_i)).$$

Expandindo $f(x, y)$, em série de Taylor, em termos de (x_i, y_i) e retendo somente os termos de derivada primeira

$$f(x_i + c_2 h, y_i + a_{21} h f(x_i, y_i)) \approx f_i + c_2 h \frac{\partial f_i}{\partial x} + a_{21} h f_i \frac{\partial f_i}{\partial y},$$

e substituindo na equação anterior

$$y_{i+1} = y_i + b_1 h f_i + b_2 h \left(f_i + c_2 h \frac{\partial f_i}{\partial x} + a_{21} h f_i \frac{\partial f_i}{\partial y} \right).$$

Rearranjando,

$$y_{i+1} = y_i + h(b_1 + b_2)f_i + h^2 \left(b_2 c_2 \frac{\partial f_i}{\partial x} + b_2 a_{21} f_i \frac{\partial f_i}{\partial y} \right). \quad (7.9)$$

Comparando (7.8) e (7.9), obtém-se um sistema não linear com 3 equações e 4 incógnitas, a partir do qual pode-se gerar uma variedade de métodos de segunda ordem

$$b_1 + b_2 = 1,$$

$$b_2 c_2 = 1/2,$$

$$b_2 a_{21} = 1/2.$$

Um exemplo de método de Runge-Kutta de segunda ordem é o chamado método de Euler modificado [31], cujas constantes são mostradas na Tabela 7.3.

Tabela 7.3 Constantes do método de Euler modificado.

| | | | |
|-----|-----|--|--|
| 0 | | | |
| 1/2 | 1/2 | | |
| | | | |

O método de Euler modificado apresenta a forma

$$y_{i+1} = y_i + hf \left(x_i + \frac{h}{2}, y_i + \frac{h}{2}f(x_i, y_i) \right). \quad (7.10)$$

Um outro método de Runge-Kutta de segunda ordem é o método de Euler melhorado [31], com suas constantes mostradas na Tabela 7.4.

Tabela 7.4 Constantes do método de Euler melhorado.

| | |
|-----|-----|
| 0 | 1 |
| 1 | 1 |
| 1/2 | 1/2 |

O método de Euler melhorado tem a forma

$$y_{i+1} = y_i + \frac{h}{2}(f(x_i, y_i) + f(x_i + h, y_i + hf(x_i, y_i))). \quad (7.11)$$

Exemplo 7.2 Comparar a solução do PVI

$$y' = -2xy^2, \text{ com } y(0) = 0,5,$$

no intervalo $[0, 1]$, com $m = 10$ subintervalos, utilizando os métodos de Euler, Euler modificado e Euler melhorado, sabendo que a solução exata é

$$y(x) = \frac{1}{x^2 + 2}.$$

A Tabela 7.5 compila os resultados obtidos pelos três métodos.

Tabela 7.5 Comparaçāo de três métodos de Euler.

(E: Euler, Emod: Euler modificado e Emel: Euler melhorado).

| i | x_i | $ y_i^E - y(x_i) $ | $ y_i^{Emod} - y(x_i) $ | $ y_i^{Emel} - y(x_i) $ |
|----|-------|-----------------------|-------------------------|-------------------------|
| 0 | 0,0 | 0 | 0 | 0 |
| 1 | 0,1 | $2,49 \times 10^{-3}$ | $1,24 \times 10^{-5}$ | $1,24 \times 10^{-5}$ |
| 2 | 0,2 | $4,80 \times 10^{-3}$ | $4,76 \times 10^{-5}$ | $2,32 \times 10^{-5}$ |
| 3 | 0,3 | $6,73 \times 10^{-3}$ | $9,83 \times 10^{-5}$ | $2,97 \times 10^{-5}$ |
| 4 | 0,4 | $8,11 \times 10^{-3}$ | $1,55 \times 10^{-4}$ | $2,89 \times 10^{-5}$ |
| 5 | 0,5 | $8,88 \times 10^{-3}$ | $2,06 \times 10^{-4}$ | $1,91 \times 10^{-5}$ |
| 6 | 0,6 | $9,04 \times 10^{-3}$ | $2,45 \times 10^{-4}$ | $2,60 \times 10^{-8}$ |
| 7 | 0,7 | $8,69 \times 10^{-3}$ | $2,67 \times 10^{-4}$ | $2,70 \times 10^{-5}$ |
| 8 | 0,8 | $7,94 \times 10^{-3}$ | $2,71 \times 10^{-4}$ | $5,96 \times 10^{-5}$ |
| 9 | 0,9 | $6,93 \times 10^{-3}$ | $2,59 \times 10^{-4}$ | $9,48 \times 10^{-5}$ |
| 10 | 1,0 | $5,77 \times 10^{-3}$ | $2,35 \times 10^{-4}$ | $1,30 \times 10^{-4}$ |

7.2.2 Métodos de quarta ordem

O mesmo desenvolvimento, visto anteriormente, pode ser utilizado para obter métodos de Runge-Kutta de ordem mais elevada. No caso de quarta ordem, obtém-se um sistema não linear com 11 equações e 13 incógnitas. Um dos métodos mais comuns desta ordem é o método clássico de Runge-Kutta, com as suas constantes sendo apresentadas na Tabela 7.6.

Tabela 7.6 Constantes do método clássico de Runge-Kutta de quarta ordem.

| | |
|-----|-------------|
| 0 | |
| 1/2 | 1/2 |
| 1/2 | 0 1/2 |
| 1 | 0 0 1 |
| 1/6 | 1/3 1/3 1/6 |

A Figura 7.2 mostra um algoritmo do método de Runge-Kutta de ordem 4. Os parâmetros de entrada são o limite inferior a , o limite superior b , o número de subintervalos m e o valor inicial y_0 . A função derivada $y' = f(x, y)$ deve ser especificada de acordo com a linguagem de programação escolhida. Os parâmetros de saída são as abscissas $VetX$ e as soluções $VetY$ do PVI.

Algoritmo RK4

{ Objetivo: Resolver um PVI pelo método de Runge-Kutta de ordem 4 }

parâmetros de entrada a, b, m, y_0

{ limite inferior, limite superior, número de subintervalos e valor inicial }

parâmetros de saída $VetX, VetY$

{ abscissas e solução do PVI }

$h \leftarrow (b - a)/m; xt \leftarrow a; yt \leftarrow y_0; VetX(1) \leftarrow xt; VetY(1) \leftarrow yt$

escreva $0, xt, yt$

para $i \leftarrow 1$ até m faça

$x \leftarrow xt; y \leftarrow yt; k1 \leftarrow f(x, y) \quad \{ avaliar f(x, y) \}$

$x \leftarrow xt + h/2; y \leftarrow yt + h/2 * k1; k2 \leftarrow f(x, y) \quad \{ avaliar f(x, y) \}$

$y \leftarrow yt + h/2 * k2; k3 \leftarrow f(x, y) \quad \{ avaliar f(x, y) \}$

$x \leftarrow xt + h; y \leftarrow yt + h * k3; k4 \leftarrow f(x, y) \quad \{ avaliar f(x, y) \}$

$xt \leftarrow a + i * h; yt \leftarrow yt + h/6 * (k1 + 2 * (k2 + k3) + k4)$

escreva i, xt, yt

$VetX(i + 1) \leftarrow xt; VetY(i + 1) \leftarrow yt$

fimpara

fimalgoritmo

Figura 7.2 Método de Runge-Kutta de ordem 4 para a solução de PVI.

Exemplo 7.3 Calcular a solução do PVI do Exemplo 7.1

$$y' = x - 2y + 1, \text{ com } y(0) = 1,$$

no intervalo $[0, 1]$, com $m = 10$ subintervalos, pelo método de Runge-Kutta de ordem 4 mostrado na Figura 7.2.

```
% Os parametros de entrada
a = 0
b = 1
m = 10
y0 = 1
% fornecem os resultados
Metodo de Runge-Kutta - ordem 4
    i      x        y
  0  0.00000  1.00000
  1  0.10000  0.91405
  2  0.20000  0.85274
  3  0.30000  0.81161
  4  0.40000  0.78700
  5  0.50000  0.77591
  6  0.60000  0.77590
  7  0.70000  0.78495
  8  0.80000  0.80143
  9  0.90000  0.82398
 10 1.00000  0.85150
```

A Tabela 7.7(a) apresenta a diferença entre o valor aproximado y_i dado pelo método de Runge-Kutta de ordem 4 com $m = 10$ subintervalos e a solução exata $y(x_i)$ dada por (7.6). A Tabela 7.7(b) mostra a comparação quando são utilizados $m = 100$ subintervalos.

Tabela 7.7 Comparação da solução pelo método de Runge-Kutta.

| i | x_i | y_i | $ y_i - y(x_i) $ | i | x_i | y_i | $ y_i - y(x_i) $ |
|-----|-------|---------|-----------------------|-----|-------|---------|------------------------|
| 0 | 0,0 | 1,00000 | 0 | 0 | 0,0 | 1,00000 | 0 |
| 1 | 0,1 | 0,9141 | $1,94 \times 10^{-6}$ | 10 | 0,1 | 0,9140 | $1,66 \times 10^{-10}$ |
| 2 | 0,2 | 0,8527 | $3,17 \times 10^{-6}$ | 20 | 0,2 | 0,8527 | $2,73 \times 10^{-10}$ |
| 3 | 0,3 | 0,8116 | $3,89 \times 10^{-6}$ | 30 | 0,3 | 0,8116 | $3,35 \times 10^{-10}$ |
| 4 | 0,4 | 0,7870 | $4,25 \times 10^{-6}$ | 40 | 0,4 | 0,7870 | $3,66 \times 10^{-10}$ |
| 5 | 0,5 | 0,7759 | $4,35 \times 10^{-6}$ | 50 | 0,5 | 0,7759 | $3,74 \times 10^{-10}$ |
| 6 | 0,6 | 0,7759 | $4,27 \times 10^{-6}$ | 60 | 0,6 | 0,7759 | $3,68 \times 10^{-10}$ |
| 7 | 0,7 | 0,7850 | $4,08 \times 10^{-6}$ | 70 | 0,7 | 0,7849 | $3,51 \times 10^{-10}$ |
| 8 | 0,8 | 0,8014 | $3,82 \times 10^{-6}$ | 80 | 0,8 | 0,8014 | $3,28 \times 10^{-10}$ |
| 9 | 0,9 | 0,8240 | $3,52 \times 10^{-6}$ | 90 | 0,9 | 0,8240 | $3,03 \times 10^{-10}$ |
| 10 | 1,0 | 0,8515 | $3,20 \times 10^{-6}$ | 100 | 1,0 | 0,8515 | $2,75 \times 10^{-10}$ |

(a) $m = 10 \rightarrow h = 0,1$.

(b) $m = 100 \rightarrow h = 0,01$.

7.2.3 Método de Dormand-Prince

Um modo de verificar se um método de Runge-Kutta produz valores dentro da exatidão desejada é recalculando o valor de y_{i+1} no final de cada intervalo, utilizando o passo h dividido ao meio. O valor será aceito se houver apenas uma pequena diferença entre os dois resultados; caso contrário, h deve ser dividido ao meio até que a exatidão desejada seja alcançada. O problema é que esta estratégia pode requerer um grande esforço computacional.

Um processo proposto por E. Fehlberg, no final da década de 1960, utiliza dois métodos de ordens diferentes, um de ordem 4 e outro de ordem 5, e compara os valores de y_{i+1} obtidos nos dois casos. O método de Runge-Kutta-Fehlberg é considerado um método de ordem 4.

No início da década de 1980, J. R. Dormand e P. J. Prince propuseram um método similar, porém de ordem 5, cujas constantes são mostradas na Tabela 7.8 [31, página 186]. Notar que, neste método, as constantes $a_{7i} = b_i$, $i = 1, 2, \dots, 6$. Nesta tabela, foi acrescentada uma linha e_i contendo os coeficientes para calcular os erros globais, que são as diferenças entre y_{i+1} obtido pelo processo de ordem 5 e o de ordem 4.

Tabela 7.8 Constantes do método de Dormand-Prince.

| | | | | | | | |
|-------|------------|-------------|------------|----------|---------------|--------|-------|
| 0 | | | | | | | |
| 1/5 | 1/5 | | | | | | |
| 3/10 | 3/40 | 9/40 | | | | | |
| 4/5 | 44/45 | -56/15 | 32/9 | | | | |
| 8/9 | 19372/6561 | -25360/2187 | 64448/6561 | -212/729 | | | |
| 1 | 9017/3168 | -355/33 | 46732/5247 | 49/176 | -5103/18656 | | |
| 1 | 35/384 | 0 | 500/1113 | 125/192 | -2187/6784 | 11/84 | |
| | 35/384 | 0 | 500/1113 | 125/192 | -2187/6784 | 11/84 | 0 |
| e_i | 71/57600 | 0 | -71/16695 | 71/1920 | -17253/339200 | 22/525 | -1/40 |

Um algoritmo do método de Dormand-Prince é apresentado na Figura 7.3. Os parâmetros de entrada são o limite inferior a , o limite superior b , o número de subintervalos m e o valor inicial y_0 . A função derivada $y' = f(x, y)$ deve ser especificada de acordo com a linguagem de programação adotada. Os parâmetros de saída são as abscissas $VetX$, as soluções $VetY$ do PVI e os erros globais EG .

Exemplo 7.4 Calcular a solução do PVI do Exemplo 7.1

$$y' = x - 2y + 1, \text{ com } y(0) = 1,$$

no intervalo $[0, 1]$, com $m = 10$ subintervalos, pelo método de Dormand-Prince apresentado na Figura 7.3.

```

Algoritmo DOPRI(5,4)
{ Objetivo: Resolver um PVI pelo método de Dormand-Prince }
parâmetros de entrada a, b, m, y0
{ limite inferior, limite superior, número de subintervalos e valor inicial }
parâmetros de saída VetX, VetY, EG
{ abscissas, solução do PVI e erro global }
{ parâmetros do método }
a21 ← 1/5; a31 ← 3/40; a32 ← 9/40
a41 ← 44/45; a42 ← -56/15; a43 ← 32/9
a51 ← 19372/6561; a52 ← -25360/2187; a53 ← 64448/6561; a54 ← -212/729
a61 ← 9017/3168; a62 ← -355/38; a63 ← 46732/5247; a64 ← 49/176
a65 ← -5103/18656; a71 ← 35/384; a73 ← 500/1113; a74 ← 125/192
a75 ← -2187/6734; a76 ← 11/84
c2 ← 1/5; c3 ← 3/10; c4 ← 4/5; c5 ← 8/9; c6 ← 1; c7 ← 1
e1 ← 71/57600; e3 ← -71/16695; e4 ← 71/1920; e5 ← -17253/339200
e6 ← 22/525; e7 ← -1/40
h ← (b - a)/m; xt ← a; yt ← y0
VetX(1) ← xt; VetY(1) ← yt; EG(1) ← 0
escreva 0, xt, yt
para i ← 1 até m faça
    xt ← xt + h; yt ← yt + a21 * k1
    k1 ← h * f(xt, yt) { avaliar f(xt, yt) }
    xt ← xt + c2 * h; yt ← yt + a31 * k1 + a32 * k2
    k2 ← h * f(xt, yt) { avaliar f(xt, yt) }
    xt ← xt + c3 * h; yt ← yt + a31 * k1 + a32 * k2 + a41 * k3
    k3 ← h * f(xt, yt) { avaliar f(xt, yt) }
    xt ← xt + c4 * h; yt ← yt + a41 * k1 + a42 * k2 + a43 * k3
    k4 ← h * f(xt, yt) { avaliar f(xt, yt) }
    xt ← xt + c5 * h; yt ← yt + a51 * k1 + a52 * k2 + a53 * k3 + a54 * k4
    k5 ← h * f(xt, yt) { avaliar f(xt, yt) }
    xt ← xt + c6 * h; yt ← yt + a61 * k1 + a62 * k2 + a63 * k3 + a64 * k4 + a65 * k5
    k6 ← h * f(xt, yt) { avaliar f(xt, yt) }
    xt ← xt + c7 * h; yt ← yt + a71 * k1 + a73 * k3 + a74 * k4 + a75 * k5 + a76 * k6
    k7 ← h * f(xt, yt) { avaliar f(xt, yt) }
    xt ← a + i * h
    yt ← yt + a71 * k1 + a73 * k3 + a74 * k4 + a75 * k5 + a76 * k6
    ErrGlobal ← e1 * k1 + e3 * k3 + e4 * k4 + e5 * k5 + e6 * k6 + e7 * k7
    VetX(i + 1) ← xt; VetY(i + 1) ← yt; EG(i + 1) ← ErrGlobal
    escreva i, xt, yt, ErrGlobal
fim para
finalgoritmo

```

Figura 7.3 Método de Dormand-Prince para a solução de PVI.

% Os parametros de entrada

a = 0
b = 1
m = 10
y0 = 1

% produzem os resultados

| | i | x | y | Erro |
|--|----|---------|---------|-----------|
| | 0 | 0.00000 | 1.00000 | |
| | 1 | 0.10000 | 0.91405 | 2.100e-07 |
| | 2 | 0.20000 | 0.85274 | 1.719e-07 |
| | 3 | 0.30000 | 0.81161 | 1.408e-07 |
| | 4 | 0.40000 | 0.78700 | 1.153e-07 |
| | 5 | 0.50000 | 0.77591 | 9.436e-08 |
| | 6 | 0.60000 | 0.77590 | 7.725e-08 |
| | 7 | 0.70000 | 0.78495 | 6.325e-08 |
| | 8 | 0.80000 | 0.80142 | 5.179e-08 |
| | 9 | 0.90000 | 0.82397 | 4.240e-08 |
| | 10 | 1.00000 | 0.85150 | 3.471e-08 |

A diferença entre o valor aproximado y_i dado pelo método de Dormand-Prince com $m = 10$ subintervalos e a solução exata $y(x_i)$ dada por (7.6) é mostrada na Tabela 7.9(a). A comparação para $m = 100$ subintervalos é vista na Tabela 7.9(b). É evidente a qualidade dos resultados obtidos pelo método de Dormand-Prince.

Tabela 7.9 Comparação da solução pelo método de Dormand-Prince.

| i | x_i | y_i | $ y_i - y(x_i) $ | i | x_i | y_i | $ y_i - y(x_i) $ |
|----|-------|--------|-----------------------|-----|-------|--------|------------------------|
| 0 | 0,0 | 1,0000 | 0 | 0 | 0,0 | 1,0000 | 0 |
| 1 | 0,1 | 0,9140 | 1,52×10 ⁻⁸ | 10 | 0,1 | 0,9140 | 1,13×10 ⁻¹³ |
| 2 | 0,2 | 0,8527 | 2,49×10 ⁻⁸ | 20 | 0,2 | 0,8527 | 1,86×10 ⁻¹³ |
| 3 | 0,3 | 0,8116 | 3,05×10 ⁻⁸ | 30 | 0,3 | 0,8116 | 2,27×10 ⁻¹³ |
| 4 | 0,4 | 0,7870 | 3,33×10 ⁻⁸ | 40 | 0,4 | 0,7870 | 2,48×10 ⁻¹³ |
| 5 | 0,5 | 0,7759 | 3,41×10 ⁻⁸ | 50 | 0,5 | 0,7759 | 2,54×10 ⁻¹³ |
| 6 | 0,6 | 0,7759 | 3,35×10 ⁻⁸ | 60 | 0,6 | 0,7759 | 2,49×10 ⁻¹³ |
| 7 | 0,7 | 0,7849 | 3,20×10 ⁻⁸ | 70 | 0,7 | 0,7849 | 2,38×10 ⁻¹³ |
| 8 | 0,8 | 0,8014 | 3,00×10 ⁻⁸ | 80 | 0,8 | 0,8014 | 2,23×10 ⁻¹³ |
| 9 | 0,9 | 0,8240 | 2,76×10 ⁻⁸ | 90 | 0,9 | 0,8240 | 2,05×10 ⁻¹³ |
| 10 | 1,0 | 0,8515 | 2,51×10 ⁻⁸ | 100 | 1,0 | 0,8515 | 1,87×10 ⁻¹³ |

(a) $m = 10$ subintervalos.(b) $m = 100$ subintervalos.

7.3 Métodos de Adams

Uma outra classe de métodos para resolver o problema de valor inicial são os chamados métodos lineares de passo múltiplo (ver Definição 7.2). Um método de passo k pode ser escrito na forma

$$\alpha_k y_{i+k} + \alpha_{k-1} y_{i+k-1} + \dots + \alpha_0 y_i = h(\beta_k f_{i+k} + \beta_{k-1} f_{i+k-1} + \dots + \beta_0 f_i), \quad (7.12)$$

onde α e β são constantes específicas de um método particular, sujeitas às condições $\alpha_k = 1$ e $|\alpha_0| + |\beta_0| \neq 0$ e sendo $f_i = f(x_i, y_i)$. Quando $\beta_k = 0$, o método é dito explícito e para $\beta_k \neq 0$ ele é dito implícito.

Dentre os métodos definidos por (7.12), existem os métodos de Adams. Se eles forem explícitos, são chamados métodos de Adams-Bashforth³, e se forem implícitos, são conhecidos como métodos de Adams-Moulton⁴. Os métodos de Adams podem ser obtidos por intermédio da integração do PVI (7.2)

$$y_{i+1} - y_i = \int_{x_i}^{x_{i+1}} f(t, y(t)) dt.$$

A função integrando $f(x, y(x))$ pode ser aproximada por um polinômio interpolador $P(x)$ que passa pelos pontos de coordenadas $(x_j, f(x_j, y_j))$. Logo,

$$y_{i+1} = y_i + \int_{x_i}^{x_{i+1}} P(x) dx. \quad (7.13)$$

7.3.1 Métodos de passo dois

Seja o polinômio de Lagrange de grau 1, que passa pelos pontos de coordenadas (x_0, f_0) e (x_1, f_1) , o qual por (3.5) na página 129, é

$$P_1(x) = f_0 \frac{x - x_1}{x_0 - x_1} + f_1 \frac{x - x_0}{x_1 - x_0}. \quad (7.14)$$

O valor de $f_0 = f(x_0, y_0)$ é obtido a partir de y_0 que é a condição inicial. Já o valor de y_1 para $f_1 = f(x_1, y_1)$ tem que ser calculado utilizando um método de passo simples. Fazendo

$$u = \frac{x - x_0}{h} \rightarrow x - x_0 = hu \text{ e } x - x_1 = x - x_0 - h = h(u - 1).$$

Substituindo as expressões acima em (7.14), tem-se

$$P_1(x) = f_1 \frac{hu}{h} + f_0 \frac{h(u-1)}{-h} \rightarrow P_1(x) = f_1 u + f_0(1-u).$$

Deste modo, (7.13) torna-se

$$y_2 = y_1 + \int_{x_1}^{x_2} P_1(x) dx.$$

Fazendo mudança de variável de $x \rightarrow u$ e sendo $dx = hdu$,

³J. C. Adams e F. Bashforth propuseram, em 1883, um método para resolver um problema de ação capilar.

⁴F. R. Moulton melhorou o método de Adams para calcular trajetórias balísticas na Primeira Guerra Mundial.

$$\begin{aligned} y_2 &= y_1 + \int_1^2 [f_1 u + f_0(1-u)] h du, \\ &= y_1 + h \left[f_1 \left(\frac{u^2}{2} \right) + f_0 \left(u - \frac{u^2}{2} \right) \right] \Big|_1^2, \\ &= y_1 + h \left(\frac{3}{2} f_1 - \frac{1}{2} f_0 \right), \\ y_2 &= y_1 + \frac{h}{2}(3f_1 - f_0). \end{aligned}$$

A expressão acima pode ser generalizada obtendo-se a fórmula explícita de Adams-Bashforth de passo $k = 2$

$$y_{i+1} = y_i + \frac{h}{2}(3f_i - f_{i-1}). \quad (7.15)$$

Este método explícito foi obtido pela integração do polinômio no intervalo $[x_1, x_2]$, sendo $P(x)$ determinado a partir dos pontos em $[x_0, x_1]$. Esta extrapolação sabidamente não produz bons resultados. Se, por outro lado, o polinômio for construído usando pontos no intervalo $[x_0, x_2]$, pode-se conseguir um método que produza resultados mais exatos. Para obter a fórmula implícita, considere o polinômio interpolador de Lagrange de grau 2 que passa pelos pontos de coordenadas (x_0, f_0) , (x_1, f_1) e (x_2, f_2) , tendo a forma

$$P_2(x) = f_0 \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + f_1 \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} + f_2 \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}.$$

Novamente, definindo uma variável auxiliar

$$u = \frac{x - x_0}{h} \rightarrow x - x_0 = hu, x - x_1 = h(u-1) \text{ e } x - x_2 = h(u-2),$$

tem-se que

$$P_2(x) = f_2 \frac{u(u-1)}{2} - f_1 u(u-2) + f_0 \frac{(u-1)(u-2)}{2}. \quad (7.16)$$

Substituindo (7.16) em (7.13), tem-se

$$y_2 = y_1 + \int_{x_1}^{x_2} P_2(x) dx.$$

Fazendo mudança de variável de $x \rightarrow u$,

$$\begin{aligned} y_2 &= y_1 + \int_1^2 \left[f_2 \frac{u(u-1)}{2} - f_1 u(u-2) + f_0 \frac{(u-1)(u-2)}{2} \right] h du, \\ &= y_1 + h \left[f_2 \left(\frac{u^3}{6} - \frac{u^2}{4} \right) - f_1 \left(\frac{u^3}{3} - u^2 \right) + f_0 \left(\frac{u^3}{6} - \frac{3u^2}{4} + u \right) \right] \Big|_1^2, \\ y_2 &= y_1 + \frac{h}{12}(5f_2 + 8f_1 - f_0). \end{aligned}$$

Generalizando, tem-se a fórmula implícita de Adams-Moulton de passo $k = 2$

$$y_{i+1} = y_i + \frac{h}{12}(5f_{i+1} + 8f_i - f_{i-1}). \quad (7.17)$$

É importante notar que o valor de $f_{i+1} = f(x_{i+1}, y_{i+1})$ é necessário para obter o próprio y_{i+1} . No entanto, o valor de y_{i+1} obtido por (7.15) pode ser usado em (7.17) para avaliar f_{i+1} e calcular um valor melhor de y_{i+1} . Deste modo, as fórmulas implícitas necessitam ser utilizadas em conjunto com uma explícita, formando um método do tipo preditor-corretor, onde (7.15) é preditor e (7.17) é corretor.

Exemplo 7.5 Calcular a solução do PVI do Exemplo 7.1

$$y' = x - 2y + 1, \text{ com } y(0) = 1,$$

no intervalo $[0, 1]$, com $m = 10$ e $m = 100$ subintervalos, utilizando o método preditor-corretor de passo dois, dado por (7.15) e (7.17).

A Tabela 7.10(a) mostra a diferença entre o valor aproximado y_i dado pelo método com $m = 10$ subintervalos e a solução exata $y(x_i)$ dada por (7.6). A mesma comparação usando $m = 100$ subintervalos é apresentada na Tabela 7.10(b). Nestas tabelas, os valores de f_i foram calculados usando o método de Dormand-Prince mostrado na Figura 7.3.

Tabela 7.10 Comparação da solução pelo método preditor-corretor de passo dois.

| i | x_i | y_i | $ y_i - y(x_i) $ | i | x_i | y_i | $ y_i - y(x_i) $ |
|-----|-------|--------|-----------------------|-----|-------|--------|-----------------------|
| 0 | 0,00 | 1,0000 | 0 | 0 | 0,00 | 1,0000 | 0 |
| 1 | 0,10 | 0,9140 | $1,52 \times 10^{-8}$ | 10 | 0,10 | 0,9140 | $1,19 \times 10^{-7}$ |
| 2 | 0,20 | 0,8526 | $1,35 \times 10^{-4}$ | 20 | 0,20 | 0,8527 | $2,06 \times 10^{-7}$ |
| 3 | 0,30 | 0,8114 | $2,19 \times 10^{-4}$ | 30 | 0,30 | 0,8116 | $2,58 \times 10^{-7}$ |
| 4 | 0,40 | 0,7867 | $2,68 \times 10^{-4}$ | 40 | 0,40 | 0,7870 | $2,84 \times 10^{-7}$ |
| 5 | 0,50 | 0,7756 | $2,92 \times 10^{-4}$ | 50 | 0,50 | 0,7759 | $2,92 \times 10^{-7}$ |
| 6 | 0,60 | 0,7756 | $2,99 \times 10^{-4}$ | 60 | 0,60 | 0,7759 | $2,88 \times 10^{-7}$ |
| 7 | 0,70 | 0,7847 | $2,94 \times 10^{-4}$ | 70 | 0,70 | 0,7849 | $2,75 \times 10^{-7}$ |
| 8 | 0,80 | 0,8011 | $2,80 \times 10^{-4}$ | 80 | 0,80 | 0,8014 | $2,58 \times 10^{-7}$ |
| 9 | 0,90 | 0,8237 | $2,62 \times 10^{-4}$ | 90 | 0,90 | 0,8240 | $2,38 \times 10^{-7}$ |
| 10 | 1,00 | 0,8513 | $2,41 \times 10^{-4}$ | 100 | 1,00 | 0,8515 | $2,17 \times 10^{-7}$ |

(a) $m = 10$ subintervalos.

(b) $m = 100$ subintervalos.

7.3.2 Métodos de passo três

O método explícito de Adams-Bashforth de passo três é obtido pela integração do polinômio interpolador de Lagrange de grau 2, dado por (7.16), no intervalo $[x_2, x_3]$. Consequentemente, (7.13) torna-se

$$\begin{aligned} y_3 &= y_2 + \int_{x_2}^{x_3} P_2(x) dx, \\ &= y_2 + \int_2^3 \left[f_2 \frac{u(u-1)}{2} - f_1 u(u-2) + f_0 \frac{(u-1)(u-2)}{2} \right] h du, \\ &= y_2 + h \left[f_2 \left(\frac{u^3}{6} - \frac{u^2}{4} \right) - f_1 \left(\frac{u^3}{3} - u^2 \right) + f_0 \left(\frac{u^3}{6} - \frac{3u^2}{4} + u \right) \right] \Big|_2^3, \\ y_3 &= y_2 + \frac{h}{12}(23f_2 - 16f_1 + 5f_0). \end{aligned}$$

Nesta equação $f_0 = (x_0, y_0)$ é avaliada a partir da condição inicial y_0 e os valores de f_1 e f_2 são obtidos aplicando um método de passo simples. Generalizando, obtém-se a fórmula explícita de Adams-Bashforth de passo $k = 3$

$$y_{i+1} = y_i + \frac{h}{12}(23f_i - 16f_{i-1} + 5f_{i-2}). \quad (7.18)$$

Como toda fórmula explícita, (7.18) é obtida por meio de uma extrapolação porque a integração é no intervalo $[x_i, x_{i+1}]$, enquanto o polinômio foi construído a partir de pontos em $[x_{i-2}, x_i]$. Usando o mesmo raciocínio mostrado na seção anterior, tem-se o método implícito de Adams-Moulton de passo $k = 3$

$$y_{i+1} = y_i + \frac{h}{24}(9f_{i+1} + 19f_i - 5f_{i-1} + f_{i-2}). \quad (7.19)$$

7.3.3 Método de Adams-Bashforth-Moulton de quarta ordem

Um dos métodos mais populares de passo múltiplo é o de Adams-Bashforth-Moulton de quarta ordem, e o preditor, explícito de passo $k = 4$, é dado por

$$y_{i+1} = y_i + \frac{h}{24}(55f_i - 59f_{i-1} + 37f_{i-2} - 9f_{i-3}), \quad (7.20)$$

enquanto o corretor, implícito de passo $k = 3$, é dado por (7.19). O corretor deve ser aplicado mais de uma vez para melhorar ainda mais o resultado.

A Figura 7.4 apresenta um algoritmo do método de Adams-Bashforth-Moulton de ordem quatro. Os parâmetros de entrada são o limite inferior a , o limite superior b , o número de subintervalos m e o valor inicial y_0 . A função derivada $y' = f(x, y)$ deve ser especificada de acordo com a linguagem de programação escolhida. Os parâmetros de saída são as abscissas $VetX$, as soluções $VetY$ do PVI e os erros $Erro$.

Neste algoritmo, os valores de f_1 , f_2 e f_3 são calculados pelo método de Dormand-Prince de passo simples apresentado na Figura 7.3. A estimativa do erro cometido no cálculo do valor corrigido [4] é dado por

$$Erro \approx \frac{19}{270} |y_{\text{corretor}} - y_{\text{preditor}}|.$$

```

Algoritmo ABM4
{ Objetivo: Resolver PVI pelo método de Adams-Bashforth-Moulton }
{ de ordem 4 }
parâmetros de entrada  $a, b, m, y_0$ 
{ limite inferior, limite superior, número de subintervalos e valor inicial }
parâmetros de saída  $VetX, VetY, Erro$ 
{ abscissas, solução do PVI e erro }
 $h \leftarrow (b - a)/m$ 
 $[VetX, VetY, Erro] \leftarrow DOPRI(a, a + 3 * h, 3, y_0)$  (ver Figura 7.3)
{ parâmetros de saída de DOPRI retornam em VetX, VetY, Erro }
para  $i \leftarrow 1$  até 4 faça
    escreva  $i = 1, VetX(i), VetY(i), Erro(i)$ 
fim para
para  $i \leftarrow 4$  até  $m$  faça
     $x \leftarrow VetX(i - 3); y \leftarrow VetY(i - 3); f0 \leftarrow f(x, y)$  { avaliar  $f(x, y)$  }
     $x \leftarrow VetX(i - 2); y \leftarrow VetY(i - 2); f1 \leftarrow f(x, y)$  { avaliar  $f(x, y)$  }
     $x \leftarrow VetX(i - 1); y \leftarrow VetY(i - 1); f2 \leftarrow f(x, y)$  { avaliar  $f(x, y)$  }
     $x \leftarrow VetX(i); y \leftarrow VetY(i); f3 \leftarrow f(x, y)$  { avaliar  $f(x, y)$  }
     $Y_{pre} \leftarrow h * (55 * f3 - 59 * f2 + 37 * f1 - 9 * f0) / 24 + VetY(i)$ 
     $VetY(i + 1) \leftarrow Y_{pre}; VetX(i + 1) \leftarrow a + i * h; x \leftarrow VetX(i + 1)$ 
    para  $j \leftarrow 1$  até 2 faça
         $y \leftarrow VetY(i + 1); f4 \leftarrow f(x, y)$  { avaliar  $f(x, y)$  }
         $Y_{cor} \leftarrow h * (9 * f4 + 19 * f3 - 5 * f2 + f1) / 24 + VetY(i)$ 
         $VetY(i + 1) \leftarrow Y_{cor}$ 
    fim para
     $Erro \leftarrow \text{abs}(Y_{cor} - Y_{pre}) * 19 / 270$ 
    escreva  $i, VetX(i + 1), VetY(i + 1), Erro$ 
fim para
finalgoritmo

```

Figura 7.4 Método de Adams-Bashforth-Moulton para a solução de PVI.

(Ver significado da função abs na Tabela 1.1, na página 6.)

Exemplo 7.6 Calcular a solução do PVI do Exemplo 7.1

$$y' = x - 2y + 1, \text{ com } y(0) = 1,$$

no intervalo $[0, 1]$, com $m = 10$ subintervalos, utilizando o algoritmo descrito na Figura 7.4.

```

% Os parâmetros de entrada
a = 0
b = 1
m = 10
y0 = 1
% produzem os resultados

```

Método de Adams-Bashforth-Moulton - ordem 4

| i | x | y | Erro |
|----|---------|---------|-------------|
| 0 | 0.00000 | 1.00000 | 0.00000e+00 |
| 1 | 0.10000 | 0.91405 | 2.10000e-07 |
| 2 | 0.20000 | 0.85274 | 1.71933e-07 |
| 3 | 0.30000 | 0.81161 | 1.40767e-07 |
| 4 | 0.40000 | 0.78699 | 4.23161e-06 |
| 5 | 0.50000 | 0.77590 | 3.51703e-06 |
| 6 | 0.60000 | 0.77589 | 2.82201e-06 |
| 7 | 0.70000 | 0.78494 | 2.33307e-06 |
| 8 | 0.80000 | 0.80142 | 1.90865e-06 |
| 9 | 0.90000 | 0.82397 | 1.56299e-06 |
| 10 | 1.00000 | 0.85150 | 1.27961e-06 |

Uma comparação, entre o resultado exato, dado por (7.6), e aqueles obtidos pelo método preditor-corretor de Adams-Bashforth-Moulton de ordem 4, é exibida na Tabela 7.11.

Tabela 7.11 Comparaçāo da solução pelo método preditor-corretor de ordem 4.

| i | x_i | y_i | $ y_i - y(x_i) $ | i | x_i | y_i | $ y_i - y(x_i) $ | i | $ y_i - y(x_i) $ |
|----|-------|--------|-----------------------|-----|-------|------------------------|------------------|------------------------|------------------|
| 0 | 0,00 | 1,0000 | 0 | 0 | 0,00 | 0 | 0 | 0 | 0 |
| 1 | 0,10 | 0,9140 | $1,52 \times 10^{-8}$ | 10 | 0,10 | $3,69 \times 10^{-10}$ | 100 | $5,02 \times 10^{-14}$ | |
| 2 | 0,20 | 0,8527 | $2,49 \times 10^{-8}$ | 20 | 0,20 | $7,33 \times 10^{-10}$ | 200 | $8,39 \times 10^{-14}$ | |
| 3 | 0,30 | 0,8116 | $3,05 \times 10^{-8}$ | 30 | 0,30 | $9,53 \times 10^{-10}$ | 300 | $1,03 \times 10^{-13}$ | |
| 4 | 0,40 | 0,7870 | $3,07 \times 10^{-6}$ | 40 | 0,40 | $1,07 \times 10^{-9}$ | 400 | $1,14 \times 10^{-13}$ | |
| 5 | 0,50 | 0,7759 | $4,94 \times 10^{-6}$ | 50 | 0,50 | $1,11 \times 10^{-9}$ | 500 | $1,16 \times 10^{-13}$ | |
| 6 | 0,60 | 0,7759 | $6,07 \times 10^{-6}$ | 60 | 0,60 | $1,10 \times 10^{-9}$ | 600 | $1,15 \times 10^{-13}$ | |
| 7 | 0,70 | 0,7849 | $6,62 \times 10^{-6}$ | 70 | 0,70 | $1,06 \times 10^{-9}$ | 700 | $1,10 \times 10^{-13}$ | |
| 8 | 0,80 | 0,8014 | $6,77 \times 10^{-6}$ | 80 | 0,80 | $9,99 \times 10^{-10}$ | 800 | $1,03 \times 10^{-13}$ | |
| 9 | 0,90 | 0,8240 | $6,65 \times 10^{-6}$ | 90 | 0,90 | $9,25 \times 10^{-10}$ | 900 | $9,43 \times 10^{-14}$ | |
| 10 | 1,00 | 0,8515 | $6,35 \times 10^{-6}$ | 100 | 1,00 | $8,44 \times 10^{-10}$ | 1000 | $8,56 \times 10^{-14}$ | |

(a) 10 subintervalos.

(b) 100 subintervalos. (c) 1000 subintervalos. ■

7.4 Comparação de métodos para EDO

Nesta seção, é apresentada uma comparação entre os métodos de passo simples (Runge-Kutta) e passo múltiplo (Adams) [3, 4]. Além disso, são apresentados cinco PVI para serem resolvidos pelos métodos de Euler, Dormand-Prince e Adams-Bashforth-Moulton.

7.4.1 Métodos de Runge-Kutta

Vantagens:

1. São auto-iniciáveis, ou seja, não dependem do auxílio de outros métodos.
2. É fácil fazer a alteração do incremento h , de modo que ele possa ser aumentado para reduzir o esforço computacional.

Desvantagens:

1. O número de vezes que a função $f(x, y)$ necessita ser avaliada, por passo, é elevada.
2. Para limitar o erro de discretização é necessário escolher um h pequeno, o que pode causar um aumento do erro de arredondamento.

7.4.2 Métodos de Adams

Vantagens:

1. O número de vezes que $f(x, y)$ é avaliada, a cada iteração i , é pequeno, uma vez nas fórmulas explícitas e $i + 1$ vezes nas implícitas.
2. As fórmulas são simples, podendo ser utilizadas até mesmo com uma calculadora.

Desvantagens:

1. Não são auto-iniciáveis, dependendo de um outro método.
2. A mudança do incremento h é mais difícil de ser feita.

7.4.3 Comparação de métodos para EDO

Com o intuito de ilustrar, numericamente, o desempenho de alguns métodos apresentados neste capítulo, serão mostrados, a seguir, os resultados dos métodos de Euler (Figura 7.1), Dormand-Prince (Figura 7.3) e Adams-Bashforth-Moulton (Figura 7.4), resolvendo os cinco PVI abaixo. A solução exata do j -ésimo PVI é dada pela expressão de $y_j(x)$.

$$\begin{aligned}f_1(x, y) &= -2x^2y^2, \quad y(0) = 2, \quad x \in [0, 2], \quad y_1(x) = \frac{6}{4x^3 + 3}, \\f_2(x, y) &= 3x^2y, \quad y(1) = 1, \quad x \in [1, 2], \quad y_2(x) = e^{x^3 - 1}, \\f_3(x, y) &= -2xy^3, \quad y(0) = 1, \quad x \in [0, 5], \quad y_3(x) = \frac{1}{\sqrt{2x^2 + 1}}, \\f_4(x, y) &= \cos(x)y, \quad y(0) = 1, \quad x \in [0, 10], \quad y_4(x) = e^{\sin(x)}, \\f_5(x, y) &= \sin(x) - y, \quad y(0) = 0, \quad x \in [0, \pi], \quad y_5(x) = \frac{e^{-x} + \sin(x) - \cos(x)}{2}.\end{aligned}$$

As Tabelas 7.12–7.16 compilam os resultados obtidos pelos três métodos resolvendo os cinco PVI. Nelas, Erro significa a maior diferença, em valor absoluto, entre a solução numérica e a solução exata e t_{rel} é a razão entre o tempo gasto pelo método e o tempo gasto pelo método de Euler. Apesar de o método de Dormand-Prince (DOPRI) demandar um pouco mais de tempo que o de Adams-Bashforth-Moulton (ABM), os seus resultados são bem melhores para estes cinco exemplos.

Tabela 7.12 Comparação de métodos para $f_1(x, y) = -2x^2y^2$.

| $m = 10$ | | | $m = 100$ | | | $m = 1000$ | | |
|----------|-----------------------|-----------|-----------|------------------------|-----------|------------|------------------------|-----------|
| Método | Erro | t_{rel} | Método | Erro | t_{rel} | Método | Erro | t_{rel} |
| Euler | $1,43 \times 10^{-1}$ | 1,0 | Euler | $1,26 \times 10^{-2}$ | 1,0 | Euler | $1,24 \times 10^{-3}$ | 1,0 |
| DOPRI | $3,51 \times 10^{-5}$ | 4,7 | DOPRI | $7,26 \times 10^{-11}$ | 6,4 | DOPRI | $3,33 \times 10^{-15}$ | 6,1 |
| ABM4 | $2,48 \times 10^{-3}$ | 4,3 | ABM4 | $3,62 \times 10^{-7}$ | 5,3 | ABM4 | $3,75 \times 10^{-11}$ | 4,9 |

Tabela 7.13 Comparação de métodos para $f_2(x, y) = 3x^2y$.

| $m = 10$ | | | $m = 100$ | | | $m = 1000$ | | |
|----------|-----------------------|-----------|-----------|-----------------------|-----------|------------|------------------------|-----------|
| Método | Erro | t_{rel} | Método | Erro | t_{rel} | Método | Erro | t_{rel} |
| Euler | $9,59 \times 10^2$ | 1,0 | Euler | $2,89 \times 10^2$ | 1,0 | Euler | $3,48 \times 10^1$ | 1,0 |
| DOPRI | $1,54 \times 10^{-1}$ | 7,0 | DOPRI | $1,18 \times 10^{-5}$ | 6,4 | DOPRI | $1,34 \times 10^{-10}$ | 6,0 |
| ABM4 | $4,96 \times 10^1$ | 6,5 | ABM4 | $2,82 \times 10^{-2}$ | 5,2 | ABM4 | $3,17 \times 10^{-6}$ | 4,8 |

Tabela 7.14 Comparação de métodos para $f_3(x, y) = -2xy^3$.

| $m = 10$ | | | $m = 100$ | | | $m = 1000$ | | |
|----------|-----------------------|-----------|-----------|------------------------|-----------|------------|------------------------|-----------|
| Método | Erro | t_{rel} | Método | Erro | t_{rel} | Método | Erro | t_{rel} |
| Euler | $1,84 \times 10^{-1}$ | 1,0 | Euler | $1,05 \times 10^{-2}$ | 1,0 | Euler | $9,97 \times 10^{-4}$ | 1,0 |
| DOPRI | $1,51 \times 10^{-4}$ | 6,2 | DOPRI | $1,99 \times 10^{-10}$ | 6,4 | DOPRI | $3,00 \times 10^{-15}$ | 6,1 |
| ABM4 | $3,99 \times 10^{-3}$ | 5,6 | ABM4 | $4,89 \times 10^{-6}$ | 5,3 | ABM4 | $6,23 \times 10^{-10}$ | 4,9 |

Tabela 7.15 Comparação de métodos para $f_4(x, y) = \cos(x)y$.

| $m = 10$ | | | $m = 100$ | | | $m = 1000$ | | |
|----------|-----------------------|-----------|-----------|-----------------------|-----------|------------|------------------------|-----------|
| Método | Erro | t_{rel} | Método | Erro | t_{rel} | Método | Erro | t_{rel} |
| Euler | $2,66 \times 10^0$ | 1,0 | Euler | $3,90 \times 10^{-1}$ | 1,0 | Euler | $4,15 \times 10^{-2}$ | 1,0 |
| DOPRI | $7,25 \times 10^{-4}$ | 6,8 | DOPRI | $1,02 \times 10^{-8}$ | 6,6 | DOPRI | $1,06 \times 10^{-13}$ | 6,0 |
| ABM4 | $5,65 \times 10^{-1}$ | 6,0 | ABM4 | $4,82 \times 10^{-5}$ | 5,3 | ABM4 | $3,65 \times 10^{-9}$ | 4,6 |

Tabela 7.16 Comparação de métodos para $f_5(x, y) = \sin(x) - y$.

| $m = 10$ | | | $m = 100$ | | | $m = 1000$ | | |
|----------|-----------------------|-----------|-----------|------------------------|-----------|------------|------------------------|-----------|
| Método | Erro | t_{rel} | Método | Erro | t_{rel} | Método | Erro | t_{rel} |
| Euler | $7,73 \times 10^{-2}$ | 1,0 | Euler | $7,18 \times 10^{-3}$ | 1,0 | Euler | $7,13 \times 10^{-4}$ | 1,0 |
| DOPRI | $4,90 \times 10^{-7}$ | 7,0 | DOPRI | $4,05 \times 10^{-12}$ | 6,6 | DOPRI | $1,22 \times 10^{-15}$ | 6,1 |
| ABM4 | $5,63 \times 10^{-5}$ | 6,0 | ABM4 | $8,72 \times 10^{-9}$ | 5,3 | ABM4 | $8,75 \times 10^{-13}$ | 4,6 |

7.5 Sistemas de equações diferenciais ordinárias

Na modelagem de um problema real, é muito comum o uso de sistemas de EDO. Isso ocorre, principalmente, porque uma equação diferencial de ordem $n > 1$ pode ser resolvida por meio de um sistema de ordem n . Um sistema de p equações diferenciais ordinárias com p

incógnitas apresenta a forma

$$\begin{aligned}y'_1 &= f_1(x, y_1, \dots, y_p), \\y'_2 &= f_2(x, y_1, \dots, y_p), \\&\dots \\y'_p &= f_p(x, y_1, \dots, y_p),\end{aligned}$$

sendo f_i e $y_i(a) = \eta_i$, $i = 1, 2, \dots, p$, as funções dadas do problema e as condições iniciais, respectivamente.

7.5.1 Algoritmo de Runge-Kutta para sistema de ordem dois

A Figura 7.5 apresenta uma adaptação do algoritmo do método de Runge-Kutta de ordem 4, mostrado na Figura 7.2, para resolver um sistema de EDO de ordem 2. Observar que ordem tem significado diferente para os dois métodos. Os parâmetros de entrada são o limite inferior a , o limite superior b , o número de subintervalos m e os valores iniciais y_{10} e y_{20} . As funções derivadas $y'_1 = f_1(x, y_1, y_2)$ e $y'_2 = f_2(x, y_1, y_2)$ devem ser especificadas de acordo com a linguagem de programação escolhida. Os parâmetros de saída são as abscissas $VetX$ e as soluções do PVI $VetY1$ e $VetY2$.

Exemplo 7.7 Resolver o sistema de EDO

$$\begin{aligned}y'_1 &= y_1 + y_2 + 3x, \\y'_2 &= 2y_1 - y_2 - x,\end{aligned}$$

com $y_1(0) = 0$ e $y_2(0) = -1$ no intervalo $[0, 2]$ com 10 subintervalos, utilizando o algoritmo da Figura 7.5.

```
% Os parametros de entrada
a = 0
b = 2
m = 10
y10 = 0
y20 = -1
% produzem os resultados
Metodo RK4 para sistema de ordem 2
i      x          y1          y2
0    0.00000  0.00000  -1.00000
1    0.20000 -0.14073  -0.86747
2    0.40000 -0.16119  -0.82388
3    0.60000 -0.04768  -0.80741
4    0.80000  0.22970  -0.75950
5    1.00000  0.72072  -0.61782
6    1.20000  1.50106  -0.30864
7    1.40000  2.68142  0.26206
8    1.60000  4.42101  1.22000
9    1.80000  6.94680  2.73780
10   2.00000 10.58102  5.05594
```

```
Algoritmo RK4sis2
{ Objetivo: Resolver sistema de EDO pelo método de Runge-Kutta }
{ de ordem 4 }
parâmetros de entrada a, b, m, y10, y20
{ limite inferior, limite superior, número de subintervalos e valores iniciais }
parâmetros de saída VetX, VetY1, VetY2
{ abscissas e soluções do PVI }
h ← (b - a)/m; xt ← a; y1t ← y10; y2t ← y20
VetX(1) ← xt; VetY1(1) ← y1t; VetY2(1) ← y2t
escreva 0, xt, y1t, y2t
para i ← 1 até m faça
  x ← xt; y1 ← y1t; y2 ← y2t
  k11 ← f1(x, y1, y2) { avaliar f1(x, y1, y2) }
  k12 ← f2(x, y1, y2) { avaliar f2(x, y1, y2) }
  x ← xt + h/2; y1 ← y1t + h/2 * k11; y2 ← y2t + h/2 * k12
  k21 ← f1(x, y1, y2) { avaliar f1(x, y1, y2) }
  k22 ← f2(x, y1, y2) { avaliar f2(x, y1, y2) }
  y1 ← y1t + h/2 * k21; y2 ← y2t + h/2 * k22
  k31 ← f1(x, y1, y2) { avaliar f1(x, y1, y2) }
  k32 ← f2(x, y1, y2) { avaliar f2(x, y1, y2) }
  x ← xt + h; y1 ← y1t + h * k31; y2 ← y2t + h * k32
  k41 ← f1(x, y1, y2) { avaliar f1(x, y1, y2) }
  k42 ← f2(x, y1, y2) { avaliar f2(x, y1, y2) }
  xt ← a + i * h
  y1t ← y1t + h/6 * ((k11 + 2 * (k21 + k31)) + k41)
  y2t ← y2t + h/6 * ((k12 + 2 * (k22 + k32)) + k42)
  escreva i, xt, y1t, y2t
  VetX(i+1) ← xt; VetY1(i+1) ← y1t; VetY2(i+1) ← y2t
fim para
fimalgoritmo
```

Figura 7.5 Método de Runge-Kutta de ordem 4 para sistema de ordem 2.

7.5.2 Equações diferenciais de segunda ordem

Uma equação diferencial ordinária de ordem $n > 1$ pode ser reduzida a um sistema de EDO de primeira ordem com n equações por meio de uma simples mudança de variáveis. Por exemplo, o PVI de segunda ordem

$$y'' = f(x, y, y') \text{ com } y(a) = \eta_1 \text{ e } y'(a) = \eta_2$$

é equivalente ao sistema de equações de primeira ordem

$$\begin{aligned}y'_1 &= y_2, \\y'_2 &= f(x, y_1, y_2),\end{aligned}$$

com $y_1(a) = \eta_1$ e $y_2(a) = \eta_2$, se forem feitas as mudanças de variáveis $y_1 = y$ e $y_2 = y'$.

Exemplo 7.8 Resolver o PVI de segunda ordem

$$y'' = y' + 2y - x^2,$$

com $y(0) = 1$ e $y'(0) = 0$, intervalo $[0, 1]$ com 10 subintervalos.

Fazendo as mudanças de variáveis $y_1 = y$ e $y_2 = y'$, tem-se o seguinte sistema de ordem dois de EDO de primeira ordem

$$y'_1 = y_2,$$

$$y'_2 = y_2 + 2y_1 - x^2,$$

com $y_1(0) = 1$ e $y_2(0) = 0$. Usando o algoritmo da Figura 7.5, obtém-se os resultados

% Os parametros de entrada

a = 0

b = 1

m = 10

y10 = 1

y20 = 0

% produzem os resultados

Metodo RK4 para sistema de ordem 2

| i | x | y1 | y2 |
|----|---------|---------|---------|
| 0 | 0.00000 | 1.00000 | 0.00000 |
| 1 | 0.10000 | 1.01035 | 0.21070 |
| 2 | 0.20000 | 1.04295 | 0.44591 |
| 3 | 0.30000 | 1.10053 | 0.71105 |
| 4 | 0.40000 | 1.18638 | 1.01276 |
| 5 | 0.50000 | 1.30456 | 1.35912 |
| 6 | 0.60000 | 1.46002 | 1.76003 |
| 7 | 0.70000 | 1.65878 | 2.22756 |
| 8 | 0.80000 | 1.90823 | 2.77646 |
| 9 | 0.90000 | 2.21738 | 3.42475 |
| 10 | 1.00000 | 2.59721 | 4.19443 |

A Tabela 7.17 compila os resultados obtidos pelo algoritmo da Figura 7.5, comparados com o valor exato dado por

$$y(x) = \frac{1}{4}(e^{2x} + 2x^2 - 2x + 3).$$

Exemplo 7.9 Resolver o PVI de segunda ordem

$$y'' = 4y' - 5y + x - 2,$$

com $y(0) = 1$ e $y'(0) = -1$, $x \in [0, 2]$ usando 8 subintervalos.

Com as mudanças de variáveis $y_1 = y$ e $y_2 = y'$, tem-se o sistema

$$y'_1 = y_2,$$

$$y'_2 = 4y_2 - 5y_1 + x - 2,$$

com $y_1(0) = 1$ e $y_2(0) = -1$. Utilizando o algoritmo da Figura 7.5, obtém-se os resultados

Tabela 7.17 Comparaçao da solução pelo algoritmo RK4sis2.

| i | x_i | y_i | $ y_i - y(x_i) $ | i | x_i | y_i | $ y_i - y(x_i) $ | i | $ y_i - y(x_i) $ |
|----|-------|---------|-----------------------|-----|-------|------------------------|------------------------|------|------------------------|
| 0 | 0,0 | 1,00000 | 0 | 0 | 0,0 | 0 | 0 | 0 | 0 |
| 1 | 0,1 | 1,0103 | $8,98 \times 10^{-7}$ | 10 | 0,1 | $1,04 \times 10^{-10}$ | $1,13 \times 10^{-14}$ | 100 | $1,13 \times 10^{-14}$ |
| 2 | 0,2 | 1,0430 | $2,17 \times 10^{-6}$ | 20 | 0,2 | $2,51 \times 10^{-10}$ | $2,69 \times 10^{-14}$ | 200 | $4,82 \times 10^{-14}$ |
| 3 | 0,3 | 1,1005 | $3,93 \times 10^{-6}$ | 30 | 0,3 | $4,53 \times 10^{-10}$ | $7,53 \times 10^{-14}$ | 300 | $1,13 \times 10^{-13}$ |
| 4 | 0,4 | 1,1864 | $6,32 \times 10^{-6}$ | 40 | 0,4 | $7,29 \times 10^{-10}$ | $1,63 \times 10^{-13}$ | 400 | $2,29 \times 10^{-13}$ |
| 5 | 0,5 | 1,3046 | $9,54 \times 10^{-6}$ | 50 | 0,5 | $1,10 \times 10^{-9}$ | $3,15 \times 10^{-13}$ | 500 | $4,29 \times 10^{-13}$ |
| 6 | 0,6 | 1,4600 | $1,38 \times 10^{-5}$ | 60 | 0,6 | $1,59 \times 10^{-9}$ | $5,76 \times 10^{-13}$ | 600 | $1,13 \times 10^{-13}$ |
| 7 | 0,7 | 1,6588 | $1,95 \times 10^{-5}$ | 70 | 0,7 | $2,25 \times 10^{-9}$ | $1,13 \times 10^{-13}$ | 700 | $1,13 \times 10^{-13}$ |
| 8 | 0,8 | 1,9082 | $2,69 \times 10^{-5}$ | 80 | 0,8 | $3,10 \times 10^{-9}$ | $1,13 \times 10^{-13}$ | 800 | $1,13 \times 10^{-13}$ |
| 9 | 0,9 | 2,2174 | $3,66 \times 10^{-5}$ | 90 | 0,9 | $4,22 \times 10^{-9}$ | $1,13 \times 10^{-13}$ | 900 | $1,13 \times 10^{-13}$ |
| 10 | 1,0 | 2,5972 | $4,93 \times 10^{-5}$ | 100 | 1,0 | $5,68 \times 10^{-9}$ | $1,13 \times 10^{-13}$ | 1000 | $1,13 \times 10^{-13}$ |

(a) 10 subintervalos.

(b) 100 subintervalos.

(c) 1000 subintervalos.

% Os parametros de entrada

a = 0

b = 1

m = 8

y10 = 1

y20 = -1

% produzem os resultados

Metodo RK4 para sistema de ordem 2

| i | x | y1 | y2 |
|---|---------|------------|------------|
| 0 | 0.00000 | 1.00000 | -1.00000 |
| 1 | 0.25000 | 0.29150 | -5.22233 |
| 2 | 0.50000 | -1.97300 | -13.86486 |
| 3 | 0.75000 | -7.25688 | -30.00220 |
| 4 | 1.00000 | -17.95859 | -58.07121 |
| 5 | 1.25000 | -37.76370 | -103.89015 |
| 6 | 1.50000 | -71.92819 | -173.98777 |
| 7 | 1.75000 | -127.22682 | -273.40703 |
| 8 | 2.00000 | -211.01753 | -400.51082 |

7.6 Exemplos de aplicação

Com a finalidade de ilustrar a utilização de equações diferenciais ordinárias na solução de problemas reais, serão apresentados dois estudos de caso: um sobre controle de poluição e outro sobre deflexão de uma viga metálica.

7.6.1 Controle de poluição

Definição do problema

Dado um volume de controle definido em torno de uma cidade que se quer fazer a análise da emissão de monóxido de carbono (CO), deseja-se calcular a concentração de CO neste local, a cada 10 minutos, no período de 8:00 às 10:00 h.

Modelagem matemática

A taxa de variação da concentração de CO em função das taxas nas quais o CO entra e sai do volume de ar do local que está sendo analisado é dada por

$$\frac{dC}{dt} = \frac{R}{V} - (K_v + K_q)C,$$

onde C é a concentração atual de CO (em kg/m^3), t é o tempo (em min), R é a taxa na qual o CO está entrando no volume de controle a partir de um agente poluidor (em kg/min) e K_v e K_q são constantes que determinam a taxa na qual o vento e as reações químicas removem o CO da unidade de controle (em min^{-1}).

Solução numérica

Utilizando os valores $C_0 = 4 \times 10^{-3} \text{ kg/m}^3$, $R = 1,5 \times 10^4 \text{ kg/min}$, $V = 5 \times 10^8 \text{ m}^3$, $K_v = 1,5 \times 10^{-3} \text{ min}^{-1}$ e $K_q = 3 \times 10^{-4} \text{ min}^{-1}$, o PVI torna-se

$$\begin{aligned}\frac{dC}{dt} &= \frac{1,5 \times 10^4}{5 \times 10^8} - (1,5 \times 10^{-3} + 3 \times 10^{-4})C \sim \\ \frac{dC}{dt} &= 3 \times 10^{-5} - 1,8 \times 10^{-3}C, \text{ com } C_0 = 4 \times 10^{-3}.\end{aligned}$$

O algoritmo do método de Dormand-Prince, mostrado na Figura 7.3, deve ser modificado de modo que o passo h seja igual a 10 min. Considerando que $t \in [8, 10]$, então tem-se $m = 12$ subintervalos. Apesar de utilizar $h = 10$, os valores das abscissas são apresentados na forma decimal: $0,16667 = 1/6$ hora = 10 min. Os resultados obtidos são

```
% Os valores de entrada
a = 8
b = 10
m = 12
y0 = 4e-3
% produzem os resultados
    Método de Dormand-Prince
    i   x       y      Erro
    0  8.00000  0.00400
    1  8.16667  0.00423 -1.949e-14
    2  8.33333  0.00445 -1.914e-14
    3  8.50000  0.00467 -1.880e-14
    4  8.66667  0.00488 -1.846e-14
    5  8.83333  0.00509 -1.813e-14
    6  9.00000  0.00530 -1.781e-14
    7  9.16667  0.00550 -1.749e-14
    8  9.33333  0.00570 -1.718e-14
    9  9.50000  0.00589 -1.687e-14
    10 9.66667  0.00609 -1.657e-14
    11 9.83333  0.00628 -1.628e-14
    12 10.00000 0.00646 -1.599e-14
```

Análise dos resultados

A Figura 7.6 mostra o aumento da concentração de monóxido de carbono à medida que o tempo decorre.

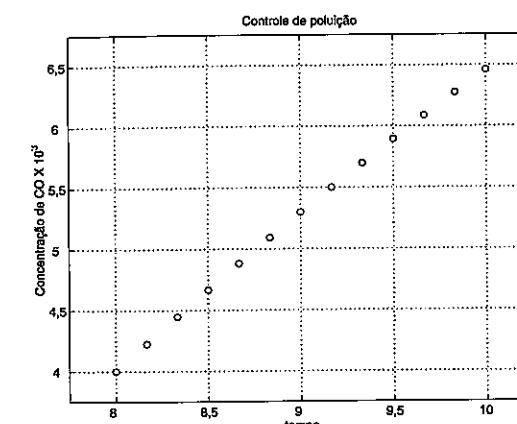


Figura 7.6 Concentração de monóxido de carbono em função do tempo.

7.6.2 Deflexão de viga

Definição do problema

Seja uma viga metálica de comprimento $L = 200 \text{ cm}$ engastada em uma de suas extremidades e sujeita a uma carga $P = 15 \text{ N}$ na extremidade livre. Calcular a deflexão da viga quando a carga se move da parte engastada até a extremidade livre.

Modelagem matemática

As deflexões elásticas de uma viga metálica satisfazem à equação diferencial de segunda ordem

$$y'' = \frac{P}{EI}(1 + (y')^2)^{1.5}(L - x),$$

onde E é o módulo de Young (dependente do material) e I é o momento de inércia da seção reta da viga. Como a viga está presa em $x = 0$, então as condições iniciais são $y(0) = 0$ e $y'(0) = 0$.

Considerando uma viga de aço estrutural com $E = 2 \times 10^5 \text{ N} \cdot \text{cm}^{-2}$, $I = 25 \text{ cm}^4$ e fazendo as mudanças de variáveis $y_1 = y$ e $y_2 = y'_1$, obtém-se o sistema de EDO

$$\begin{aligned}y'_1 &= y_2, \\ y'_2 &= \frac{15}{2 \times 10^5 \times 25}(1 + (y_2)^2)^{1.5}(200 - x) = 3 \times 10^{-6}(1 + (y_2)^2)^{1.5}(200 - x),\end{aligned}$$

com $y_1(0) = 0$ e $y_2(0) = 0$.

Solução numérica

Utilizando o método de Runge-Kutta para o sistema de ordem dois, dado na Figura 7.5, no intervalo $[0, 200]$ e com $m = 20$ subintervalos, obtém-se os resultados

```
% Os valores de entrada
a = 0
b = 200
m = 20
y10 = 0
y20 = 0
% produzem os resultados
Metodo RK4 para sistema de ordem 2
i      x        y1        y2
0  0.00000  0.00000  0.00000
1  10.00000  0.02950  0.00585
2  20.00000  0.11600  0.01140
3  30.00000  0.25652  0.01665
4  40.00000  0.44805  0.02161
5  50.00000  0.68762  0.02626
6  60.00000  0.97224  0.03061
7  70.00000  1.29891  0.03467
8  80.00000  1.66466  0.03843
9  90.00000  2.06648  0.04189
10 100.00000 2.50139  0.04505
11 110.00000 2.96640  0.04790
12 120.00000 3.45849  0.05046
13 130.00000 3.97468  0.05272
14 140.00000 4.51195  0.05468
15 150.00000 5.06731  0.05634
16 160.00000 5.63773  0.05770
17 170.00000 6.22022  0.05875
18 180.00000 6.81175  0.05951
19 190.00000 7.40932  0.05996
20 200.00000 8.00990  0.06011
```

Análise dos resultados

A Figura 7.7 mostra a deflexão da viga em função da distância em que a carga é colocada. Em um estudo mais detalhado, deve-se levar em conta que, a partir de uma certa carga, este modelo elástico não será mais válido, visto que a deflexão estará na faixa plástica.

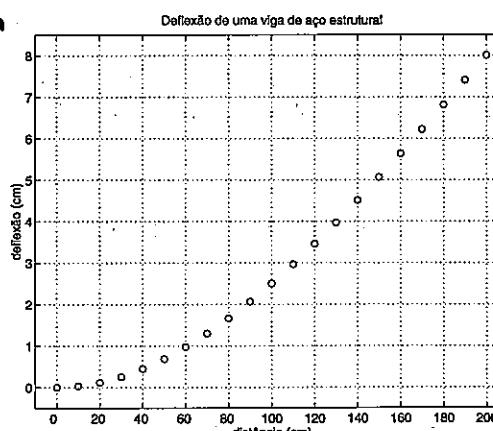


Figura 7.7 Deflexão de uma viga metálica em função da distância da carga.

7.7 Exercícios

Seção 7.1

Resolver os problemas de valor inicial abaixo utilizando o método de Euler com o número de subintervalos m indicado

$$7.1. y' = \sqrt{x}, y(0) = 0, x \in [0, 2] \text{ e } m = 5.$$

$$7.2. y' = x^2 + y^2, y(1) = 0, x \in [1, 2] \text{ e } m = 8.$$

$$7.3. y' = xy, y(0) = 1, x \in [0, 1] \text{ e } m = 10.$$

7.4. Implementar o método de Euler mostrado na Figura 7.1 utilizando qualquer linguagem.

7.5. Resolver os Exercícios 7.1–7.3, usando o programa do Exercício 7.4.

Seção 7.2

7.6. Utilizando qualquer linguagem de programação, implementar o método de Runge-Kutta de ordem quatro apresentado na Figura 7.2.

7.7. Implementar o método de Dormand-Prince exibido na Figura 7.3.

Resolver os três PVI abaixo usando os algoritmos de Runge-Kutta e Dormand-Prince implementados nos Exercícios 7.6 e 7.7. Comparar os resultados com o valor exato dado por $y(x)$

$$7.8. y' = -\sin(x)y, y(0) = -1, x \in [0, \pi], m = 10 \text{ e } y(x) = -e^{\cos(x)-1}.$$

$$7.9. y' = (\sqrt{x} + 1)y, y(1) = 1, x \in [1, 3], m = 20 \text{ e } y(x) = e^{2x^{1.5}/3+x-5/3}.$$

$$7.10. y' = x^2 - 3x - 1 - y, y(1) = 1, x \in [1, 2], m = 50 \text{ e } y(x) = e^{1-x} + x^2 - 5x + 4.$$

Seção 7.3

7.11. Implementar em uma linguagem de programação o método preditor-corretor de Adams-

Bashforth-Moulton de ordem quatro mostrado na Figura 7.4.

Resolver os quatro PVI dados abaixo utilizando o algoritmo de Adams-Bashforth-Moulton implementado no Exercício 7.11. Comparar os resultados com o valor exato dado por $y(x)$

$$7.12. y' = x \cos(x) + y, y(0) = \pi, x \in [0, \pi/2], m = 20 \text{ e } y(x) = \pi e^x + \frac{\sin(x)(x+1) - \cos(x)x}{2}.$$

$$7.13. y' = (e^x x - 1)y, y(1) = -1, x \in [1, 2], m = 100 \text{ e } y(x) = -e^{(e^x-1)(x-1)}.$$

$$7.14. y' = e^x x^2 + y, y(0) = -1, x \in [0, 1], m = 100 \text{ e } y(x) = e^x \left(\frac{x^3}{3} - 1 \right).$$

$$7.15. y' = 5x^3 + 2x^2 + x - 1 - y, y(0) = 1, x \in [0, 2], m = 200 \text{ e } y(x) = 29e^{-x} + 5x^3 - 13x^2 + 27x - 28.$$

Seção 7.4

Comparar o valor exato dos PVI abaixo dado por $y(x)$ com os valores obtidos pelos métodos Dormand-Prince e Adams-Bashforth-Moulton de ordem quatro

$$7.16. y' = \sin(x) \cos(x) - y, y(0) = 0, x \in [0, \pi], m = 100 \text{ e } y(x) = \frac{e^{-x} - \cos(2x)}{5} + \frac{\sin(2x)}{10}.$$

$$7.17. y' = x - \cos(x) + y, y(0) = -1, x \in [0, \pi], m = 100 \text{ e } y(x) = \frac{\cos(x) - \sin(x) - e^x}{2} - x - 1.$$

$$7.18. y' = xy(x+1), y(0) = 1, x \in [0, 1], m = 100 \text{ e } y(x) = e^{x^2(2x+3)/6}.$$

$$7.19. y' = x^2 + 3x - 5, y(0) = 0, x \in [0, 2], m = 100 \text{ e } y(x) = \frac{x^3}{3} + \frac{3x^2}{2} - 5x.$$

7.20. $y' = e^{2x+1} + 3x^2 + y$, $y(-1) = 1/e$, $x \in [-1, 1]$, $m = 100$ e

$$y(x) = e^{2x+1} + 3e^{x+1} - 3x^2 - 6x - 6.$$

Seção 7.5

7.21. Implementar o método de Runge-Kutta de ordem quatro para resolver o sistema de EDO de ordem dois mostrado na Figura 7.5.

Resolver os PVI abaixo utilizando o algoritmo de Runge-Kutta para sistema de EDO implementado no Exercício 7.21

7.22. $y'_1 = -y_1 + y_2$, $y'_2 = -y_1 - y_2$, $y_1(0) = 0$, $y_2(0) = 1$, intervalo $[0, 1]$ e $m = 100$.

7.23. $y'_1 = y_1 + 3y_2$, $y'_2 = -y_1 + y_2$, $y_1(0) = 1$, $y_2(0) = 0$, intervalo $[0, 2]$ e $m = 100$.

7.24. $y'' = y' + 2y - x \operatorname{sen}(x)$, $y(0) = 1$, $y'(0) = -1$, $x \in [0, 1]$ e $m = 100$.

7.25. $y'' = -y' + 2y - x^2 + 1$, $y(1) = -1$, $y'(1) = 1$, $x \in [1, 2]$ e $m = 100$.

Gerais

7.26. Usando qualquer linguagem de programação, implementar o método de Euler modificado dado por (7.10).

7.27. Implementar o método de Euler melhorado dado por (7.11).

7.28. Elaborar um algoritmo para resolver sistema de EDO de ordem três baseado no método de Runge-Kutta de ordem quatro.

7.29. Usando uma linguagem de programação qualquer, implementar o algoritmo do Exercício 7.28.

7.30. Resolver o sistema de EDO de ordem três dado abaixo usando o programa implementado no Exercício 7.29

$$\begin{aligned} y'_1 &= y_2 + y_3, \\ y'_2 &= -y_1 - y_3, \\ y'_3 &= y_1 - y_2, \\ \text{com } y_1(0) &= 1, y_2(0) = 1, y_3(0) = 1, \\ &\text{no intervalo } [0, 1] \text{ e } m = 100. \end{aligned}$$

Apêndice A

Linguagem FORTRAN

O FORTRAN, acrônimo de FORmula TRANslation, surgiu no final da década de 1950, sendo a primeira linguagem de programação de alto nível. O fato de esta linguagem vir modernizando-se durante décadas faz com que seja ainda a linguagem de programação mais utilizada em aplicações técnicas e científicas.

Neste texto, serão apresentados apenas os comandos necessários para implementar, na linguagem FORTRAN, os algoritmos escritos na notação proposta na Seção 1.2. O manual do usuário e/ou um livro texto [16] devem ser consultados para maiores informações.

A.1 Estrutura de um programa FORTRAN

O FORTRAN ainda conserva conceitos do tempo em que se programava em cartões perfurados de 80 colunas, consideradas uma linha de programa. Os comandos devem respeitar a seguinte disposição: as colunas de 1 a 5 devem conter o número do comando; a coluna 6 contendo um caractere diferente de zero ou branco indica que esta linha é uma continuação da linha anterior; as colunas de 7 a 72 devem conter as declarações e os comandos; e as colunas de 73 a 80 são ignoradas pelo compilador, pois elas eram usadas para numerar os cartões! Uma estrutura típica de um programa FORTRAN é

```
program <nome>
<declarações de variáveis>
<comandos>
end
subroutine <nome> ( <lista_de_parâmetros> )
<declarações de variáveis>
<comandos>
end
```

```
function <nome> ( <lista_de_parâmetros> )
<declarações de variáveis>
<comandos>
end
```

onde `program` é a palavra-chave para definir o início físico do programa, `subroutine` e `function` são palavras-chave usadas para especificar o início de subprogramas, `<nome>` são identificadores do programa e subprogramas, `<declarações de variáveis>` especifica nomes e tipos para as variáveis que são usadas no escopo do programa e dos subprogramas, `<comandos>` são as estruturas de controle e de atribuição usadas para implementar os algoritmos e `<lista_de_parâmetros>` contém as variáveis utilizadas para troca de informações entre o programa e os subprogramas. O comando `end` determina o fim físico do programa e dos subprogramas. O uso de subprogramas será visto na Seção A.8.

A.2 Variáveis e comentários

As variáveis são representadas por identificadores que são cadeias de caracteres alfanuméricos começando com uma letra. As variáveis devem ser declaradas e podem ser de vários tipos, entre os quais `integer` para variáveis sem ponto flutuante, `real` para variáveis de ponto flutuante com 4 bytes, `real*8` (ou `double precision`) para variáveis de ponto flutuante com 8 bytes, `complex` para variáveis de ponto flutuante complexas, `logical` para variáveis lógicas e `character` para variáveis literais.

Se as variáveis não forem declaradas, serão assumidos os seguintes tipos: `integer` se os nomes começarem com as letras `i`, `j`, `k`, `l`, `m` e `n`; e do tipo `real` se começarem com as demais letras. Estas declarações implícitas devem ser evitadas, pois constitui uma boa norma de programação declarar todas as variáveis.

No caso de vetores e matrizes, as suas dimensões máximas devem ser especificadas quando da sua declaração. Ao declarar uma variável, pode-se atribuir-lhe um valor inicial, colocando-o entre barras (/ /).

Exemplo A.1 Sejam as variáveis e suas declarações

| |
|-----------------------|
| <i>Tempo</i> |
| <i>v_i</i> |
| <i>M_{ij}</i> |

```
real*8 Tempo / 3.2d-3 /
integer v(3) / 1, 3, 4 /
real M(5,8)
```

A variável `Tempo` contém valores de precisão dupla (8 bytes), sendo o valor inicial igual a $3,2 \times 10^{-3}$, a variável `v` é um vetor com 3 posições contendo valores inteiros (sem casas decimais) com `v(1)=1`, `v(2)=3` e `v(3)=4`, e `M` é uma matriz de dimensão 5×8 contendo valores reais de precisão simples (4 bytes). Uma constante de precisão dupla deve conter o caractere `d`, e `8.5d55` significa $8,5 \times 10^{55}$.

Um comentário é um texto inserido em qualquer parte do algoritmo para aumentar a sua clareza. Se for colocado o caractere `c` ou `C` ou `*` na coluna 1, então o texto a seguir será considerado um comentário.

Exemplo A.2 Seja o texto de comentário

{ Cálculo da raiz }

c Calculo da raiz

A.3 Expressões e comando de atribuição

São aceitos três tipos de expressões: aritméticas, lógicas e literais.

Expressões aritméticas

Uma expressão aritmética é construída com operadores e operandos aritméticos. Os operadores podem ser de adição (+), subtração (-), multiplicação (*) e divisão (/). Os operandos podem ser constantes, variáveis e funções aritméticas. Usualmente, uma expressão é usada em um comando de atribuição e, durante a execução do comando, a expressão é avaliada e o resultado atribuído a uma variável. Um comando de atribuição tem o formato

`<variável> = <expressão_aritmética>`,

sendo o operador de atribuição o símbolo formado pelo caractere (=).

Exemplo A.3 Sejam um trecho de algoritmo e sua implementação

```
velocidade←deslocamento/tempo
ângulo←cos(2+x)
delta←raiz(b2-4*a*c)
```

```
real a, angulo, b, c, delta, x
real*8 deslocamento, tempo, velocidade
velocidade = deslocamento/tempo
angulo = cos(2.+x)
delta = sqrt(b**2-4.*a*c)
```

A potenciação é indicada por **, isto é, b^2 é implementado por meio da expressão `b**2` e a^{i+1} por `a**(i+1)`. A Tabela A.1 apresenta algumas funções matemáticas do FORTRAN. Dependendo do tipo de argumento da função e do resultado, os nomes das funções devem ser precedidos por uma outra letra. Consulte o manual do usuário para obter a lista completa das funções.

Expressões lógicas

Expressão lógica é aquela cujos operadores são lógicos e cujos operandos são relações e/ou variáveis do tipo lógico. O resultado de uma relação ou de uma expressão lógica é verdadeiro (.true.) ou falso (.false.). Os operadores relacionais do FORTRAN são mostrados na Tabela A.2.

Os operadores lógicos permitem a combinação ou negação das relações lógicas. Os operadores lógicos do FORTRAN são listados na Tabela A.3.

Tabela A.1 Algumas funções matemáticas do FORTRAN.

| Função | Descrição | Função | Descrição |
|-----------------|--|--------|--|
| Trigonométricas | | | |
| sin | seno | asin | arco seno |
| cos | co-seno | acos | arco co-seno |
| tan | tangente | atan | arco tangente |
| Exponenciais | | | |
| exp | exponencial | sqrt | raiz quadrada |
| alog | logaritmo natural | alog10 | logaritmo decimal |
| tanh | tangente hiperbólica | atanh | arco tangente hiperbólica |
| sinh | seno hiperbólico | cosh | co-seno hiperbólico |
| Complexas | | | |
| abs | módulo | conjg | complexo conjugado |
| aimag | parte imaginária do complexo | real | parte real do complexo |
| Numéricas | | | |
| ifix | converte real para inteiro truncando a parte decimal | float | converte inteiro para real |
| dble | converte para precisão dupla | cmplx | converte para complexo |
| char | converte inteiro para caractere | sign | sinal ($\text{sign}(x,y)$ é definida como $ x $ se $y \geq 0$ e $- x $ se $y < 0$) |
| abs | valor absoluto | mod | resto de divisão |
| amax1 | maior valor | amin1 | menor valor |

Tabela A.2 Operadores relacionais do FORTRAN.

| Operador relacional | Descrição |
|---------------------|------------------|
| .gt. | maior que |
| .ge. | maior ou igual a |
| .lt. | menor que |
| .le. | menor ou igual a |
| .eq. | igual a |
| .ne. | diferente de |

Tabela A.3 Operadores lógicos do FORTRAN.

| Operador lógico | Descrição | Uso |
|-----------------|-----------|-----------|
| .and. | e | conjunção |
| .or. | ou | disjunção |
| .not. | não | negação |

Exemplo A.4 Sejam um trecho de algoritmo e a sua implementação

```
a ← 10
t ← verdadeiro
a = 10
t = .true.
w = (a.eq.5).and.t
```

```
integer a
logical t, w
a = 10
t = .true.
w = (a.eq.5).and.t
```

Em uma expressão lógica mais complexa, as relações devem ser colocadas entre parênteses, por exemplo, $(a.ge.b).and.(c.lt.0).or..not(d.ne.0)$. As expressões lógicas podem ser usadas tanto para expressar condições em estruturas condicionais ou em estruturas de repetição quanto em comandos de atribuição do tipo

`<variável_lógica> = <expressão_lógica>.`

Expressões literais

Uma variável literal contém caracteres em vez de um número. Os caracteres devem ser delimitados por apóstrofos ' ' para serem atribuídos a uma variável literal.

Exemplo A.5 Atribuição de uma variável literal

```
mensagem ← "matriz singular"
```

```
character*15 mensagem
mensagem = 'matriz singular'
```

Duas cadeias de caracteres podem ser concatenadas por intermédio do operador literal //, por exemplo, 'numero' // 'complexo' resultando em 'numero complexo'. A função len(r) determina o comprimento da cadeia de caracteres r, por exemplo, len('texto') resulta em 5. A função index(r,s) retorna o índice do primeiro caractere da cadeia r igual à subcadeia s, ou seja, index('abcd','bc') tem como resultado 2.

A.4 Comandos de entrada e saída

Leitura

O comando read faz a leitura de dados formatados em arquivo e sua sintaxe simplificada é

```
read(<unidade_entrada>,<nf>) <lista_de_variáveis>
<nf> format(<formato>)
```

sendo `<unidade_entrada>` o número da unidade de entrada (console, disco, fita magnética etc.) na qual será feita a leitura dos dados da `<lista_de_variáveis>` com a especificação de formato número `<nf>`. Este número é definido pelo programador e não pode ser repetido em outro format. Se `<unidade_entrada>` contiver o caractere *, então os valores serão lidos pelo console. A cadeia de caracteres `<formato>`, contendo caracteres alfanuméricos e/ou especificações de conversão, determina a forma de exibição da `<lista_de_variáveis>`. Algumas especificações de conversão são mostradas na Tabela A.4. Deve haver uma especificação de formato correspondente a cada variável da lista.

Tabela A.4 Alguns formatos de exibição do FORTRAN.

| Formato | Especificação |
|---------|--|
| in | usado para valores inteiros, sendo <i>n</i> o tamanho do campo; |
| fn.d | notação na forma [-]888.88, sendo <i>n</i> o tamanho do campo (número total de caracteres) e <i>d</i> o número de dígitos decimais; |
| en.d | notação na forma [-]0.888E±88, sendo <i>n</i> o tamanho do campo (número total de caracteres) e <i>d</i> o número de dígitos decimais; |
| an | caracteres em um campo de tamanho <i>n</i> ; |
| ln | valor lógico T (verdadeiro) ou F (falso) em um campo de tamanho <i>n</i> ; |
| nx | coloca <i>n</i> espaços em branco entre dois formatos; |
| / | começa a leitura/impressão em uma linha nova. |

Exemplo A.6 O trecho de programa

```
read(*,15) Indice, Raiz
15 format(i3,2x,f10.0)
```

faz a leitura da variável Indice do tipo integer em um campo de 3 posições e de Raiz do tipo real em um campo de 10 posições com 2 espaços em branco entre os dois campos

123^1234.56789

onde o caractere ^ representa um espaço em branco¹. Como o ponto decimal é definido no campo de leitura, o número de decimais do formato f pode conter o valor 0.

Exibição

O comando print * é utilizado para exibir os valores de variáveis dentro do programa.

Exemplo A.7 Exibir as variáveis m e n

escreva m, n print *, m, n

No entanto, o comando write tem mais recursos. Ele grava dados formatados em um arquivo, sendo sua sintaxe simplificada

```
write(<unidade_saída>,<nf>) <lista_de_variáveis>
<nf> format(<formato>)
```

onde <unidade_saída> é o número da unidade de saída (tela, disco, fita magnética etc.) na qual será feita a gravação dos valores contidos na <lista_de_variáveis> com a especificação de formato número <nf> definido pelo programador. Caso a <unidade_saída> contenha o caractere *, então os valores serão exibidos na tela. Uma especificação de formato deve corresponder a cada variável da lista.

¹O caractere não consta da entrada, sendo colocado no texto apenas para representar os espaços em branco.

Exemplo A.8 O trecho de programa

```
a = 123.456789
b = -12345.6789
c = .true.
write(*,16) a, b, c
16 format('a = ',f6.2,' b = ',e10.3,' c = ',12)
```

produz os resultados

a^=123.46^b^=-0.123E+05^c^=^T

onde o caractere ^ representa um espaço em branco².

Uma seqüência de caracteres colocada entre apóstrofos é exibida como tal. Quando for necessário ter o caractere (' mostrado, basta usá-lo duas vezes.

Exemplo A.9 Exibição do caractere '

```
write(*,26)
26 format('a matriz e'' singular')
```

produz

a matriz e' singular

Uso de arquivo

O comando open associa um arquivo a uma unidade de entrada ou saída. Sua sintaxe simplificada é

```
open(<unidade>, file=<nome_arquivo>, status=<estado>)
```

onde <unidade> é o número da unidade de entrada ou saída que está sendo associada ao arquivo identificado pela variável literal <nome_arquivo>. A outra variável literal <estado> informa se o arquivo já existe e está sendo aberto para leitura (status='old') ou um arquivo novo será gravado (status='new').

O comando close desconecta um arquivo de uma unidade de entrada ou saída. Sua sintaxe simplificada é

```
close(<unidade>, status=<estado>)
```

sendo <unidade> o número da unidade de entrada ou saída que está sendo desconectada do arquivo. A variável literal <estado> informa se o arquivo deve ser conservado após a execução do close (status='keep') ou se o arquivo pode ser apagado (status='delete').

²O caractere não é impresso, ele foi colocado no texto apenas para representar os espaços em branco.

Exemplo A.10 Seja o algoritmo para conversão de graus Fahrenheit para Celsius

```
Algoritmo Converte_grau
{ Objetivo: Converter grau Fahrenheit para Celsius }
leia Fahrenheit
Celsius ← (Fahrenheit - 32) * 5/9
escreva Fahrenheit, Celsius
fimalgoritmo
```

e o arquivo fahrenheit.dat contendo o valor de grau Fahrenheit.

212.0

O programa faz a leitura do grau no arquivo fahrenheit.dat

```
program Converte_grau
c   Objetivo: converter grau Fahrenheit para Celsius
real Celsius, Fahrenheit
open(1,file='fahrenheit.dat',status='old')
open(2,file='celsius.sai',status='new')
read(1,11) Fahrenheit
Celsius = (Fahrenheit - 32.) * 5. / 9.
write(2,12) Fahrenheit, Celsius
close(2,status='keep')
stop
11 format(f5.0)
12 format(f8.3,' Fahrenheit = ',f8.3,' Celsius')
end
```

e grava o arquivo celsius.sai, cujo conteúdo é

212.000 Fahrenheit = 100.000 Celsius

A.5 Estruturas condicionais

Estrutura condicional simples

É implementada por meio da estrutura if--endif

```
se <condição> então
  <comandos>
fimse
```

```
if ( <condição> ) then
  <comandos>
endif
```

onde if, then e endif são palavras-chave, <condição> é uma expressão lógica que se for verdadeira faz com que a lista <comandos> seja executada.

Exemplo A.11 Implementar o algoritmo para calcular o logaritmo decimal de um número positivo.

```
Algoritmo Logaritmo_decimal
{ Objetivo: Calcular logaritmo decimal }
leia x
se x ≥ 0 então
  LogDec ← log10(x)
  escreva x, LogDec
fimse
fimalgoritmo
```

```
c   program Logaritmo_decimal
      Objetivo: calcular logaritmo decimal
      real LogDec, x
      read *, x
      if ( x.gt.0.0) then
        LogDec = alog10(x)
        print *, x, LogDec
      endif
      stop
    end
```

Estrutura condicional composta

A sua implementação é feita pela estrutura if--else--endif

```
se <condição> então
  <comandos_1>
senão
  <comandos_2>
fimse
```

```
if ( <condição> ) then
  <comandos1>
else
  <comandos2>
endif
```

sendo if, then, else e endif palavras-chave, <condição> uma expressão lógica que caso seja verdadeira faz com que a seqüência <comandos1> seja executada e a seqüência <comandos2> não seja executada. Se o resultado de <condição> for falso, então somente a lista <comandos2> será executada.

Exemplo A.12 Implementar um algoritmo para avaliar a função modular $f(x) = |2x|$.

```
Algoritmo Função_modular
{ Objetivo: Avaliar uma função modular }
leia x
se x ≥ 0 então
  fx ← 2 * x
senão
  fx ← -2 * x
fimse
escreva x, fx
fimalgoritmo
```

```

program Funcao_modular
    Objetivo: avaliar uma função modular
    real fx, x
    read *, x
    if ( x.ge.0.0 ) then
        fx = 2.0 * x
    else
        fx = -2.0 * x
    endif
    print *, x, fx
    stop
end

```

A.6 Estruturas de repetição

Número indefinido de repetições

Utiliza-se o comando do while com um comando go to

```

repita
    <comandos_1>
    se <condição> então
        interrompa
    fimse
    <comandos_2>
fimrepita
<comandos_3>

```

```

do while (.true.)
    <comandos1>
    if ( <condição> ) then
        go to <rótulo>
    endif
    <comandos2>
enddo
<rótulo> continue
<comandos3>

```

onde do while, go to, continue e enddo são palavras-chaves e <rótulo> é um número de comando. O comando go to <rótulo> faz com que a execução seja transferida para o comando neutro <rótulo> continue. As listas <comandos1> e <comandos2> serão repetidas até que a expressão lógica <condição> torne verdadeiro. Quando isso ocorrer, a repetição será interrompida (<comandos2> não será executada) e a lista <comandos3> passa a ser executada.

Exemplo A.13 Implementar o algoritmo para determinar o maior número de ponto flutuante que somado a 1 seja igual a 1.

```

Algoritmo Epsilon
{ Objetivo: Determinar a precisão da máquina }
Epsilon ← 1
repita
    Epsilon ← Epsilon/2
    se Epsilon + 1 = 1 então
        interrompa
    fimse
fimrepita
escreva Epsilon
finalgoritmo

```

```

program precisao
    Objetivo: determinar a precisão da máquina
    real Epsilon
    Epsilon = 1.0
    do while (.true.)
        Epsilon = Epsilon / 2.0
        if ( Epsilon + 1.0.eq.1.0 ) then
            go to 10
        endif
    enddo
10 continue
print *, Epsilon
stop
end

```

Número definido de repetições

É implementada pela estrutura do--enddo

```

para <controle> ← <valor_inicial> até <valor_final> passo <delta> faça
    <comandos>
fimpara

```

```

do <controle> = <valor_inicial>, <valor_final>, <passo>
    <comandos>
enddo

```

com do e enddo sendo palavras-chaves. Inicialmente, é atribuído à variável <controle> o valor de <valor_inicial> e verificado se ele é maior do que o <valor_final>. Se for maior, a estrutura do--enddo não é executada. Se for menor ou igual, então os <comandos> serão executados e a variável <controle> será incrementada com o valor de <delta>. Novamente, é verificado se a variável <controle> é maior do que o <valor_final>, se não for maior, então os <comandos> serão executados e assim sucessivamente. Se <passo> for igual a 1, então ele pode ser omitido do comando do.

Exemplo A.14 Implementar o algoritmo para mostrar que a soma dos n primeiros números ímpares é igual ao quadrado de n .

```

Algoritmo Primeiros_impar
{ Objetivo: Verificar propriedade dos números ímpares }
leia n
Soma ← 0
para i ← 1 até 2 * n - 1 passo 2 faça
    Soma ← Soma + i
fimpara
escreva Soma, n2
finalgoritmo

```

```

program Primeiros_impares
c      Objetivo: verificar propriedade dos numeros impares
integer i, n
real Soma
read *, n
Soma = 0.0
do i = 1, 2*n-1, 2
    Soma = Soma + i
enddo
print *, Soma, n**2
stop
end

```

A.7 Falha no programa

No caso de inconsistência nos parâmetros de entrada ou de uma operação que possa causar uma operação inválida, utiliza-se o comando `return` para abandonar a execução de uma `subroutine` ou `function`

abandone **return**

Se o teste de consistência for feito no programa principal, utiliza-se o comando `stop` em vez do `return`.

A.8 Subprogramas

Sub-rotina

Um algoritmo pode ser implementado como uma sub-rotina (`subroutine`), sendo a forma geral para a declaração

```

subroutine <nome>(<lista_de_parâmetros>
<declarações de variáveis locais>
<comandos>
end

```

sendo `subroutine` uma palavra-chave que indica o início de uma declaração de sub-rotina, `<nome>` o identificador da sub-rotina e `<lista_de_parâmetros>`, quando houver, declara a lista de variáveis que são usadas para a troca de informações entre o programa e os subprogramas. As `<declarações de variáveis locais>` declaram os nomes e tipos das variáveis locais, cujo escopo é encontrado somente dentro da sub-rotina. Os `<comandos>` são estruturas que implementam os algoritmos. O comando `end` indica o final físico da `subroutine` e transfere a execução para o comando imediatamente seguinte à chamada do subprograma. Opcionalmente, para retornar de um subprograma pode-se utilizar o comando `return`, o qual não precisa vir, imediatamente, antes do `end`. Um subprograma pode ser chamado tanto pelo programa principal quanto por um outro subprograma. Os parâmetros de uma `subroutine` são de entrada e/ou saída.

Exemplo A.15 Forma para declaração de parâmetros

Algoritmo raiz
parâmetros de entrada a, b, c
parâmetros de saída x, y

```

subroutine raiz(a,b,c,x,y)
integer a,b,x
real c,y

```

Tanto os parâmetros de entrada quanto os de saída são passados por referência, isto é, o endereço do parâmetro é passado para o subprograma e, portanto, qualquer alteração que for feita, neste parâmetro, dentro do subprograma será retornada ao programa que chamou.

O comando `call` tem que ser utilizado para evocar uma `subroutine` dentro do programa ou do subprograma, sendo seu formato

```
call <nome_da_sub-rotina>(<lista_de_parâmetros>)
```

onde `<nome_da_sub-rotina>` é o mesmo identificador usado quando da declaração do subprograma e a `<lista_de_parâmetros>` é uma lista contendo os parâmetros, cujos valores são usados pela sub-rotina chamada. Os parâmetros passados para a sub-rotina devem ser iguais em número e tipo daqueles declarados no subprograma.

No comando de leitura foi usado o chamado do implícito, que faz a leitura dos `n` elementos do vetor `X` dispostos em uma linha.

Função

Em vários algoritmos apresentados, freqüentemente, é necessário avaliar funções do tipo $y = f(x)$. Isto pode ser implementado por meio de uma `function`

```

<tipo> function <nome_da_função> (<parâmetros_de_entrada>)
<declarações de variáveis locais>
<comandos>
<nome_da_função> = <resultado>
end

```

onde `function` é uma palavra-chave para indicar o início de uma função, `<nome_da_função>` é o nome escolhido da `function` usado para retornar o resultado, `<tipo>` é o tipo do resultado retornado e `<parâmetros_de_entrada>` é a declaração da lista de parâmetros, cujos valores serão usados para o cálculo da função. A `<declarações de variáveis locais>` declara os nomes e tipos das variáveis que serão usadas somente no escopo da `function`. Os `<comandos>` são as estruturas de controle e atribuição utilizadas para implementar o algoritmo da função. O comando `<nome_da_função> = <resultado>` deve aparecer na implementação para que o resultado dos cálculos seja atribuído ao nome da função e retornado ao programa que a evocou. O comando `end` indica o final físico da `function` e transfere a execução para o comando que chamou a função. Opcionalmente, para retornar de uma

function pode-se utilizar o comando `return`, o qual não precisa vir, imediatamente, antes do `end`. Uma `function` deve sempre ser chamada a partir de uma expressão aritmética que apareça do lado direito de um comando de atribuição

```
<variável> = <nome_da_função>(<parâmetros_de_entrada>).
```

Passagem de parâmetros

Uma forma alternativa de passar parâmetros entre programa e subprogramas é por meio do comando `common`, cuja sintaxe é

```
common /<rótulo1>/ <variáveis1>, /<rótulo2>/ <variáveis2>, ...
```

onde `common` é uma palavra-chave, `<rótulo>`, opcional, é o identificador de uma área de memória em comum e `<variáveis>` são os nomes das variáveis, separados por vírgula, que compartilham o mesmo espaço de memória. Na realidade, as variáveis no `common`, ao compartilharem o mesmo espaço de memória, permitem economizar memória do computador e o tempo de transferência de argumentos entre módulos do programa. Por outro lado, o seu uso pode dificultar a depuração do programa.

Exemplo A.16 Seja o trecho de programa

```
program comun
integer Código, Tipo
real Media, Raiz
logical Verifica
common /bloco1/ Código, Raiz, Verifica, /bloco2/ Tipo, Media
...
end
c
subroutine sub1
integer Código
real Raiz
logical Verifica
common /bloco1/ Código, Raiz, Verifica
...
end
c
subroutine sub2
integer Tipo
real Media, Raiz
common /bloco2/ Tipo, Media
...
end
```

O bloco em comum `bloco1` permite a passagem dos parâmetros `Código`, `Raiz` e `Verifica` entre o programa `comun` e a sub-rotina `sub1`. Por sua vez, a área `bloco2` faz o compartilhamento das variáveis `Tipo` e `Media` entre o programa e a sub-rotina `sub2`.

A.9 Exemplos de programas

Exemplo A.17 Decompor a matriz do Exemplo 2.37 usando um programa implementado a partir do algoritmo da decomposição de Cholesky mostrado na Figura 2.7. Os dados da matriz estão no arquivo `matriz.dat` e os resultados devem ser gravados no arquivo `cholesky.sai`.

```
program usa_cholesky
c
integer CondErro, n, nmax
c      define ordem maxima da matriz
parameter (nmax=100)
real*8 A(nmax,nmax), Det
open(1, file='matriz.dat', status='old')
open(3, file='cholesky.sai', status='new')
read(1,11) n
do i = 1, n
    read(1,21) (A(i,j), j = 1, n)
enddo
c      chama subrotina para fazer a decomposicao de Cholesky
call cholesky(n, nmax, A, Det, CondErro)
do i = 1, n
    write(3,13) (A(i,j), j = 1, i)
enddo
write(3,23) Det, CondErro
close(3, status='keep')
11 format(i3)
13 format(10f8.4)
21 format(10f5.0)
23 format('Det = ',f10.5,5x,'CondErro = ',i1)
end
c
subroutine cholesky(n, nmax, A, Det, CondErro)
c
c      Objetivo: fazer a decomposicao de Cholesky de uma matriz A,
c      simetrica e definida positiva.
c      parametros de entrada: n, nmax, A
c      ordem, ordem maxima e matriz
c      parametros de saida: A, Det, CondErro
c      fator L, determinante e condicao de erro
c
integer CondErro, n, nmax
real*8 A(nmax,n), Det
c      variaveis locais
integer i, j, k
real*8 r, Soma, t
c
CondErro = 0
Det = 1.0d0
do j = 1, n
    Soma = 0.0d0
    do k = 1, j-1
```

```

        Soma = Soma + A(j,k)**2
    enddo
    t = A(j,j) - Soma
    if ( t.gt.0.0d0) then
        A(j,j) = dsqrt(t)
        r = 1.0d0 / A(j,j)
        Det = Det * t
    else
        CondErro = 1
        print *, 'a matriz nao e'' definida positiva'
    endif
    do i = j+1, n
        Soma = 0.0d0
        do k = 1, j-1
            Soma = Soma + A(i,k) * A(j,k)
        enddo
        A(i,j) = ( A(i,j) - Soma ) * r
    enddo
enddo
end

```

A partir do arquivo matriz.dat

```

4
9.   6.  -3.   3.
6.  20.   2.  22.
-3.   2.   6.   2.
3.  22.   2.  28.

```

é gerado o arquivo cholesky.sai

```

3.0000
2.0000 4.0000
-1.0000 1.0000 2.0000
1.0000 5.0000 -1.0000 1.0000
Det = 576.00000 CondErro = 0

```

O comando parameter é usado para dar um valor inicial a uma variável, a qual não pode ter o seu valor alterado. Deve ser observado que, se uma matriz é passada como parâmetro, então o número de linhas definido em sua declaração deve ser idêntico ao número de linhas definido no subprograma.

Exemplo A.18 Calcular a taxa de juros do Plano 1 do Exemplo de aplicação 6.7.1 utilizando o método pégaso descrito na Figura 6.15. Os dados de entrada estão no arquivo juros.dat e os resultados devem ser mostrados na tela.

```

program calcula_juros
integer CondErro, Iter, IterMax, Prazo
real*8 A, Avista, B, Entrada, Fator, Juros, Mensalidade, Toler
common /funcao_juros/ Fator, Prazo
open(1, file='juros.dat', status='old')
c

```

```

read(1,11) Avista, Entrada, Mensalidade, Prazo
read(1,11) A, B, Toler, IterMax
Fator = (Avista - Entrada) / Mensalidade
call pegaso(A,B,Toler,IterMax,Juros,Iter,CondErro)
write(*,16) Juros, CondErro
11 format(3f10.0,i3)
16 format('Juros = ',f9.5,' CondErro = ',i1)
end

c
subroutine pegaso(A, B, Toler, IterMax, Raiz, Iter, CondErro)
c      parametros de entrada: A, B, Toler, IterMax
c      limite inferior, limite superior, tolerancia e
c      numero maximo de iteracoes
c      parametros de saida: Raiz, Iter, CondErro
c      raiz, numero de iteracoes gastas e condicao de erro
c
integer CondErro, Iter, IterMax
real*8 A, B, Toler, Raiz
c      variaveis locais
real*8 DeltaX, Fa, Fb, Funcao, Fx, X

Fa = Funcao(A)
Fb = Funcao(B)
X = B
Fx = Fb
Iter = 0
write(*,16)
do while (.true.)
    DeltaX = -Fx / (Fb - Fa) * (B - A)
    X = X + DeltaX
    Fx = Funcao(X)
    write(*,26) Iter, A, Fa, B, Fb, X, Fx, DeltaX
    if ( (dabs(DeltaX).le.Toler.and.dabs(Fx).le.Toler) .or.
j     Iter.ge.IterMax ) then
        go to 10
    endif
    if (Fx*Fb.lt.0.0d0) then
        A = B
        Fa = Fb
    else
        Fa = Fa * Fb / (Fb + Fx)
    endif
    B = X
    Fb = Fx
    Iter = Iter + 1
enddo
10 continue
Raiz = X
c      teste de convergencia
if( dabs(DeltaX).le.Toler.and.dabs(Fx).le.Toler) then
    CondErro = 0
else
    CondErro = 1

```

```

        endif
16 format(15x,'Calculo de raiz de equacao pelo metodo pegaso','/iter'
         g,4x,'a',9x,'Fa',8x,'b',9x,'Fb',8x,'x',8x,'Fx',10x,'Delta_x')
26 format(i3,5f10.5,2e12.3)
end
c      Definicao da funcao f(x)
real*8 function Funcao(x)
integer Prazo
real*8 Fator, x
common /funcao_juros/ Fator, Prazo
Funcao = (1.0d0-(1.0d0+x)**(-Prazo))/x - Fator
return
end

```

A partir do arquivo juros.dat

| | | | |
|-------|------|--------|-----|
| 1100. | 100. | 224.58 | 6 |
| 0.05 | 0.1 | 1d-5 | 100 |

são obtidos os resultados

| Calculo de raiz de equacao pelo metodo pegaso | | | | | | | |
|---|---------|---------|---------|----------|---------|------------|------------|
| iter | a | Fa | b | Fb | x | Fx | Delta_x |
| 0 | 0.05000 | 0.62294 | 0.10000 | -0.09750 | 0.09323 | -0.976E-02 | -0.677E-02 |
| 1 | 0.05000 | 0.56626 | 0.09323 | -0.00976 | 0.09250 | -0.937E-04 | -0.732E-03 |
| 2 | 0.05000 | 0.56088 | 0.09250 | -0.00009 | 0.09249 | 0.138E-06 | -0.710E-05 |
| Juros = 0.09249 CondErro = 0 | | | | | | | |

Apêndice B

Linguagem Pascal

A linguagem Pascal foi proposta por Niklaus Wirth, em 1968, com base na linguagem Algol-60, com o primeiro compilador tendo ficado operacional em 1970 [27]. Segundo o seu autor, a linguagem Pascal tem dois objetivos principais. O primeiro é tornar disponível uma linguagem adequada para ensinar programação como uma disciplina sistemática, baseada em certos conceitos fundamentais, que são clara e naturalmente refletidos pela linguagem. O segundo é desenvolver implementações desta linguagem que sejam confiáveis e eficientes nos computadores disponíveis.

O manual do usuário ou um livro texto [17] deve ser consultado para maiores informações porque serão apresentados apenas os comandos necessários para implementar, na linguagem Pascal, os algoritmos escritos na notação proposta na Seção 1.2.

B.1 Estrutura de um programa Pascal

A estrutura típica de um programa Pascal é

```

program <nome>;
  <declarações de variáveis globais>;
  <declarações de procedimentos>;
  <declarações de funções>;
begin
  <comandos>;
end.

```

onde **program** é uma palavra-chave que identifica o início físico do programa, **<nome>** é o nome escolhido para identificar o programa, **<declarações de variáveis globais>** especifica nomes e tipos para as variáveis que são usadas no escopo do programa, **<declarações de procedimentos>** e as **<declarações de funções>** são usadas para a construção de sub-

programas do tipo procedure e function, conforme será descrito na Seção B.8, begin e end delimitam o escopo do programa ou de uma lista de comandos e <comandos> são as estruturas de controle e de atribuição usadas para implementar os algoritmos.

Todas as declarações e comandos terminam com o caractere (;), exceto o último end do programa que termina com (.) para sinalizar o fim do texto do programa Pascal.

B.2 Variáveis e comentários

As variáveis são representadas por identificadores que são cadeias de caracteres alfanuméricicos iniciadas por uma letra. As variáveis têm que ser declaradas e podem ser de vários tipos, entre os quais integer para variáveis sem ponto flutuante, real para variáveis de ponto flutuante com 4 bytes, double para variáveis de ponto flutuante com 8 bytes, boolean para variáveis lógicas e char para as variáveis literais. As declarações de variáveis têm o seguinte formato:

```
var
  <lista_de_variaveis>:<tipo_1>;
  ...
  <lista_de_variaveis>:<tipo_n>;
```

onde var é uma palavra-chave que inicia a declaração, <lista_de_variaveis> é uma lista contendo uma ou mais variáveis separadas por (,) que sejam do mesmo tipo e <tipo_i> são os tipos permitidos em Pascal, como integer, real, double, boolean e char.

No caso de vetores e matrizes, as suas dimensões devem ser especificadas quando da sua declaração.

Exemplo B.1 Sejam as variáveis e suas declarações

```
Tempo
v
M
```

```
var
  Tempo: double;
  v: array [1..3] of integer;
  M: array [1..5,1..8] of real;
```

Pelas declarações, a variável Tempo contém constante numérica de ponto flutuante com 8 bytes, a variável v é um vetor com 3 posições contendo valores inteiros e M é uma matriz de dimensão 5 × 8 contendo valores de ponto flutuante com 4 bytes.

É possível a criação de novos tipos, além daqueles predefinidos por meio da declaração

```
type <identificador_tipo> = <descrição>;
```

sendo type uma palavra-chave que inicia a declaração do novo tipo, <identificador_tipo> o nome que identificará o novo tipo e <descrição>, que faz a descrição do novo tipo. Por exemplo, a matriz M do exemplo acima pode ser definida por

```
type
  matriz = array [1..5,1..8] of real;
var
  M: matriz;
```

Um comentário é um texto inserido em qualquer parte do algoritmo para aumentar a sua clareza, e qualquer texto delimitado por { e } é considerado um comentário.

Exemplo B.2 Seja o texto de comentário

{ Cálculo da raiz }

{ Calculo da raiz }

B.3 Expressões e comando de atribuição

São aceitos três tipos de expressões: aritméticas, lógicas e literais.

Expressões aritméticas

Uma expressão aritmética é construída com operadores e operandos aritméticos. Os operadores podem ser de adição (+), subtração (-), multiplicação (*) e divisão (/). Os operandos podem ser constantes, variáveis e funções aritméticas. Geralmente, uma expressão é usada em um comando de atribuição e, durante a execução do comando, a expressão é avaliada e o resultado atribuído a uma variável. Um comando de atribuição tem o formato

```
<variável> := <expressão_aritmética>;
```

sendo o operador de atribuição o símbolo formado pelos dois caracteres (:=).

Exemplo B.3 Sejam um trecho de algoritmo e sua implementação

```
velocidade←desloca/tempo
ângulo←cos(2+x)
delta←raiz2((b2-4*a*c))
velocidade := desloca/tempo;
ângulo := cos(2+x);
delta := sqrt(b*b-4*a*c);
```

Infelizmente, a linguagem Pascal não possui o operador de potenciação. Por isso, b^2 é implementado por meio da expressão $b*b$ (ou $sqr(b)$) e a^{i+1} por $exp((i+1)*ln(a))$.

A Tabela B.1 apresenta algumas funções matemáticas do Pascal. Para obter a lista completa das funções, consulte o manual do usuário.

Tabela B.1 Algumas funções matemáticas do Pascal.

| Função | Descrição | Função | Descrição |
|-----------------|--|--------|---------------------------------------|
| Trigonométricas | | | |
| sin | seno | cos | co-seno |
| arctan | arco tangente | | |
| Exponenciais | | | |
| exp | exponencial | ln | logaritmo natural |
| sqrt | raiz quadrada | sqr | quadrado |
| Numéricas | | | |
| trunc | converte real para inteiro truncando a parte decimal | round | arredonda para o inteiro mais próximo |
| abs | valor absoluto | | |

Expressões lógicas

Expressão lógica é aquela cujos operadores são lógicos e cujos operandos são relações e/ou variáveis do tipo lógico. O resultado de uma relação ou de uma expressão lógica é verdadeiro (true) ou falso (false). Os operadores relacionais do Pascal são mostrados na Tabela B.2.

Tabela B.2 Operadores relacionais do Pascal.

| Operador relacional | Descrição |
|---------------------|------------------|
| > | maior que |
| >= | maior ou igual a |
| < | menor que |
| <= | menor ou igual a |
| = | igual a |
| <> | diferente de |

Os operadores lógicos permitem a combinação ou a negação das relações lógicas e estão listados na Tabela B.3.

Tabela B.3 Operadores lógicos do Pascal.

| Operador lógico | Descrição | Uso |
|-----------------|-----------|-----------|
| and | e | conjunção |
| or | ou | disjunção |
| not | não | negação |

Exemplo B.4 Sejam um trecho de algoritmo e a sua implementação

```
a ← 10
t ← verdadeiro
w ← (a=5) e t
```

```
var
  a: integer;
  t, w: boolean;
...
a := 10;
t := true;
w := (a=5) and t;
```

Em uma expressão lógica mais complexa, as relações devem ser colocadas entre parênteses, por exemplo, $(a >= b)$ and $(c < 0)$ or not($d <> 0$). As expressões lógicas podem ser usadas tanto para expressar condições em estruturas condicionais ou em estruturas de repetição quanto em comandos de atribuição do tipo

`<variável_lógica> := <expressão_lógica>;`

Expressões literais

Uma variável literal contém um caractere em vez de um número. O caractere deve ser delimitado por apóstrofos ' ' para ser atribuído a uma variável literal. Uma cadeia de caracteres pode ser atribuída a uma variável com subscrito, onde cada posição contém um caractere, porém a variável deve ser declarada de um modo especial.

Exemplo B.5 Atribuição de uma variável literal

```
mensagem ← "matriz singular"
```

```
var mensagem: packed array[1..15] of char;
...
mensagem := 'matriz singular';
```

Duas cadeias de caracteres podem ser concatenadas por intermédio da função literal concat, por exemplo, concat('numero', 'complexo') resulta em 'numero complexo'. A função length(r) determina o comprimento da cadeia de caracteres r, ou seja, length('texto') resulta em 5. A função pos(s,r) retorna a posição do primeiro caractere da subcadeia s igual a cadeia r, por exemplo, pos('bc', 'abcd') tem como resultado 2.

B.4 Comandos de entrada e saída

Leitura

Os comandos read e readln fazem a leitura de dados em uma unidade de entrada, sendo suas sintaxes

```
read(<unidade_entrada>, <lista_de_variáveis>);
readln(<unidade_entrada>, <lista_de_variáveis>);
```

onde `<unidade_entrada>` é o nome interno da unidade de entrada na qual os dados serão lidos, se omitido então é assumido o teclado e `<lista_de_variáveis>` são nomes de variáveis do tipo `integer`, `real`, `double`, `boolean` ou `char` que deverão ser lidas. A diferença entre os comandos `read` e `readln` é que o último faz uma mudança de linha na unidade de entrada após a leitura dos valores.

Os campos correspondentes às variáveis numéricas da `<lista_de_variáveis>` devem conter caracteres que representam números inteiros e/ou de ponto flutuante. Estes números devem ser separados por pelo menos um espaço em branco. Os campos de variáveis literais ocupam exatamente os espaços com os quais foram declarados, sendo permitido o caractere correspondente ao espaço em branco.

Exemplo B.6 O trecho de programa

```
var
  Indice: integer;
  Raiz: real;
  Titulo: packed array [1..30] of char;
  ...
  readln(Indice,Raiz,Titulo);
```

faz a leitura no teclado da variável `Indice` do tipo `integer`, de `Raiz` do tipo `real` e da cadeia de caracteres `Titulo`.

`123^1234.56789^Cadeia de caracteres`

onde o caractere `^` representa um espaço em branco¹. ■

Exibição

Os comandos `write` e `writeln` exibem os dados em uma unidade de saída e apresentam as sintaxes

```
write(<unidade_saida>, <lista_de_variáveis>);
writeln(<unidade_saida>, <lista_de_variáveis>);
```

sendo `<unidade_saida>` o nome interno da unidade de saída na qual os dados serão gravados, se omitido então é assumida a tela e `<lista_de_variáveis>` são nomes de variáveis do tipo `integer`, `real`, `boolean` ou `char` que deverão ser escritas.

A diferença entre esses dois comandos é que `writeln` provoca uma mudança de linha na unidade de saída após a escrita dos valores.

¹O caractere não consta da entrada, sendo colocado no texto apenas para representar os espaços em branco.

Exemplo B.7 O trecho de programa

```
var
  Indice: integer;
  Raiz: real;
  Teste: boolean;
  Titulo: packed array [1..10] of char;
  ...
  Indice := 45;
  Raiz := -12345.6789;
  Teste := true;
  Titulo := 'Algoritmos';
  writeln(Indice,Raiz,Teste,Titulo);
```

produz os resultados na tela

`45-1.234567890000000e+04TrueAlgoritmos` ■

Deve ser notado que o valor de cada variável foi impresso ocupando um campo, isto é, um determinado número de colunas ou de caracteres, cujo tamanho vai depender do compilador utilizado.

O tamanho do campo de saída de uma variável pode ser controlado, usando-se para isto a notação `<variável>:<n>`, onde `<n>` é o número mínimo de caracteres a ser escrito. Se `<n>` definir um campo de tamanho insuficiente para conter o valor, então será destinado mais espaço. Por outro lado, se `<n>` for demasiado, o espaço excedente será completado com espaços em branco à esquerda.

Para variáveis do tipo `real`, é disponível o formato `<variável>:<n>:<d>`, sendo `<d>` o número de dígitos decimais. Neste caso, o valor de `<n>` deve ser dimensionado, considerando-se o espaço ocupado por `<d>`, pelo ponto decimal e pelo eventual sinal negativo.

Exemplo B.8 Substituindo no trecho de programa acima, o comando `writeln` por

`writeln(Indice:5,Raiz:15:5,Teste:6,Titulo:12);`

exibe os resultados na tela

`^^45^^-12345.67890^^True^^Algoritmos`

onde o caractere `^` representa um espaço em branco². ■

Uma seqüência de caracteres colocada entre apóstrofos é impressa como tal. Quando for necessário ter o caractere `('')` exibido, deve-se usá-lo duas vezes.

²O caractere não é impresso, ele foi colocado no texto apenas para representar os espaços em branco.

Exemplo B.9 Exibição do caractere '
writeln('a matriz e' singular');
produz

a matriz e' singular

Uso de arquivo

Tanto os comandos `read` e `readln` quanto os `write` e `writeln` podem ser usados para leitura e gravação de arquivos em outros dispositivos de entrada e saída. A linguagem Pascal possibilita o uso de arquivos do tipo binário e do tipo texto, sendo este último o único a ser abordado nesta seção. Os arquivos do tipo texto tem que ser declarados como variáveis do programa, na forma

```
var
  <lista_de_arquivos>: text;
```

onde `var` é uma palavra-chave usada para declarações, `<lista_de_arquivos>` são os identificadores internos associados aos arquivos e `text` é uma palavra-chave para caracterizar os arquivos do tipo texto.

O comando `assign` associa um arquivo interno ao programa a um arquivo físico, externo ao programa. Sua sintaxe é

```
assign(<nome_arquivo>, <nome_externo>);
```

onde `<nome_arquivo>` é o nome do arquivo usado pelo programa e `<nome_externo>` é o nome externo do arquivo, tal como reconhecido pelo sistema operacional. Antes de utilizar um arquivo, é preciso abri-lo com um dos três comandos

`reset(<nome_arquivo>);` usado para abrir o arquivo `<nome_arquivo>`, já existente, somente para leitura, a partir de seu início;

`rewrite(<nome_arquivo>);` utilizado para criar o arquivo `<nome_arquivo>` ou, caso ele exista, para apagá-lo e criar um novo arquivo, somente para escrita a partir de seu início;

`append(<nome_arquivo>);` abre o arquivo `<nome_arquivo>`, já existente, exclusivamente para escrita, a partir do último caractere já gravado.

O comando `close` desconecta um arquivo interno ao programa de um arquivo físico, externo ao programa. Sua sintaxe é

```
close(<nome_arquivo>);
```

sendo `<nome_arquivo>` o nome do arquivo utilizado pelo programa. Este comando não é necessário quando o arquivo for usado apenas para leitura.

Exemplo B.10 Seja o algoritmo para conversão de graus Fahrenheit para Celsius

| |
|---|
| Algoritmo Converte_grau { Objetivo: Converter grau Fahrenheit para Celsius } leia Fahrenheit Celsius ← (Fahrenheit - 32) * 5/9 escreva Fahrenheit, Celsius finalgoritmo |
|---|

e o arquivo `fahrenheit.dat` contendo o valor de grau Fahrenheit.

212.0

O programa faz a leitura do grau no arquivo `fahrenheit.dat`

```
program Converte_grau;
  { Objetivo: converter grau Fahrenheit para Celsius }
var Celsius, Fahrenheit: real;
  entrada, saida: text;
begin
  { associar o nome interno 'entrada' ao nome externo 'fahrenheit.dat' }
  assign(entrada, 'fahrenheit.dat');
  { abrir o arquivo somente para leitura }
  reset(entrada);
  { associar o nome interno 'saida' ao nome externo 'celsius.sai' }
  assign(saida, 'celsius.sai');
  { criar um arquivo novo ou se ja' existir entao apagar e recriar }
  rewrite(saida);
  readln(entrada, Fahrenheit);
  Celsius := (Fahrenheit - 32) * 5 / 9;
  writeln(saida, Fahrenheit:8:3, ' Fahrenheit = ', Celsius:8:3, ' Celsius');
  { fechar o arquivo }
  close(saida);
end.
```

e grava o arquivo `celsius.sai` com o conteúdo

212.000 Fahrenheit = 100.000 Celsius

B.5 Estruturas condicionais

Estrutura condicional simples

A sua implementação é feita por meio da estrutura `if--then`

```

    se <condição> então
      <comandos>
    fimse
  
```

```

if <condição> then
begin
  <comandos>;
end;
  
```

onde if, then, begin e end são palavras-chave, <condição> é uma expressão lógica que se for verdadeira faz com que a lista <comandos> seja executada. Se a lista de comandos for composta por mais de um comando, então ela deve estar delimitada por begin e end.

Exemplo B.11 Implementar o algoritmo para calcular o logaritmo decimal de um número positivo.

```

Algoritmo Logaritmo_decimal
{ Objetivo: Calcular logaritmo decimal }
leia x
se x > 0 então
  LogDec ← log10(x)
  escreva x, LogDec
fimse
finalgoritmo
  
```

```

program Logaritmo_decimal;
{ Objetivo: calcular logaritmo decimal }
var
  LogDec, x: real;
begin
  readln( x );
  if x > 0 then
    begin
      LogDec := ln(x) / ln(10);
      writeln( x:6:2, LogDec:8:5 );
    end
  end.
  
```

Estrutura condicional composta

É implementada pela estrutura if--then--else

```

    se <condição> então
      <comandos_1>
    senão
      <comandos_2>
    fimse
  
```

```

if <condição> then
begin
  <comandos1>;
end
else;
begin
  <comandos1>;
end;
  
```

sendo if, then, else, begin e end palavras-chave, <condição> uma expressão lógica que caso seja verdadeira faz com que a seqüência <comandos1> seja executada e a seqüência <comandos2> não seja executada. Se o resultado de <condição> for falso, então somente a lista <comandos2> será executada. É importante observar que não se pode usar o caractere (;) antes do else.

Exemplo B.12 Implementar um algoritmo para avaliar a função modular $f(x) = |2x|$.

```

Algoritmo Função_modular
{ Objetivo: Avaliar uma função modular }
leia x
se x ≥ 0 então
  fx ← 2 * x
senão
  fx ← -2 * x
fimse
escreva x, fx
finalgoritmo
  
```

```

program Funcao_modular;
{ Objetivo: avaliar uma função modular }
var
  fx, x: real;
begin
  readln( x );
  if x ≥ 0 then
    fx := 2 * x
  else
    fx := -2 * x;
  writeln( x:7:2, fx:7:2 );
end.
  
```

B.6 Estruturas de repetição

Número indefinido de repetições

A linguagem Pascal não dispõe de um modo simples para implementar a estrutura de repetição com um número indefinido de repetições na sua forma genérica, que é com interrupção no interior. Uma maneira de implementá-la é por intermédio da estrutura while--do

```

repita
  <comandos_1>
  se <condição> então
    interrompa
  fimse
  <comandos_2>
fimrepita
<comandos_3>
  
```

```

<comandos1>;
while not <condição> do
begin
  <comandos2>;
  <comandos1>;
end;
<comandos3>;
  
```

Exemplo B.13 Implementar o algoritmo para determinar o maior número de ponto flutuante que somado a 1 seja igual a 1.

```
Algoritmo Epsilon
{ Objetivo: Determinar a precisão da máquina }
Epsilon ← 1
repita
  Epsilon ← Epsilon / 2
  se Epsilon + 1 = 1 então
    interrompa
  fimse
fimrepita
escreva Epsilon
finalgoritmo
```

```
program precisao;
{ Objetivo: determinar a precisao da maquina }
var
  Epsilon: real;
begin
  Epsilon := 1;
  Epsilon := Epsilon / 2;
  while Epsilon + 1 <> 1 do
    Epsilon := Epsilon / 2;
  writeln( Epsilon );
end.
```

Número definido de repetições

Quando o passo for igual a 1, a implementação pode ser feita pelo comando for--do

```
para <controle> ← <valor-inicial> até <valor-final> passo 1 faça
  <comandos>
fimpara
```

```
for <controle> := <valor_inicial> to <valor_final> do
begin
  <comandos>;
end;
```

Quando o passo for -1 , utiliza-se `downto` em vez de `to`. No entanto, o Pascal não admite um passo diferente de 1 ou -1 . Por este motivo, a estrutura `para--faça` pode ser implementada por meio de um `for--do` adaptado

```
para <controle> ← <valor-inicial> até <valor-final> passo <delta> faça
  <comandos>
fimpara
```

```
for <controle> := <valor_inicial> + (<controle> - 1) * <delta> to ((<valor_final>-<valor_inicial>) div <delta>) + 1 do
begin
  <controle> := <valor_inicial> + (<controle> - 1) * <delta>;
  <comandos>;
end;
```

com `for`, `to` e `do` sendo palavras-chaves e `div` é o operador de divisão inteira.

Exemplo B.14 Implementar o algoritmo para mostrar que a soma dos n primeiros números ímpares é igual ao quadrado de n .

```
Algoritmo Primeiros_imparés
{ Objetivo: Verificar propriedade dos números ímpares }
leia n
Soma ← 0
para i ← 1 até 2 * n - 1 passo 2 faça
  Soma ← Soma + i
fimpara
escreva Soma, n2
finalgoritmo
```

```
program Primeiros_imparés;
{ Objetivo: verificar propriedade dos numeros impares }
var
  i, j, n: integer;
  Soma: real;
begin
  readln( n );
  Soma := 0;
  for j := 1 to ((2*n-1 - 1) div 2) + 1 do
    begin
      i := 1 + (j - 1) * 2;
      Soma := Soma + i;
    end;
  writeln( Soma:8:2, n**2:8:2 );
end.
```

B.7 Falha no programa

No caso de inconsistência nos parâmetros de entrada ou de uma operação que possa causar uma operação inválida, utiliza-se o comando `goto <rótulo>` para transferir a execução para o final de uma procedure ou function (ver Seção B.8). O `<rótulo>` deve ser declarado como label.

**escreva "grau menor que 1"
abandone**

```
procedure ...  
var  
...  
label retorna;  
begin  
...  
writeln('grau menor que 1');  
goto retorna;  
...  
retorna: { saída }  
end;
```

Alguns compiladores disponibilizam formas mais simples de abandonar um subprograma imediatamente, em caso de falha, como os comandos `return` ou `exit`.

B.8 Subprogramas

Procedimento

Um algoritmo pode ser implementado na forma de um procedimento (`procedure`), sendo a sua declaração na forma

```
procedure <nome>(<lista_de_parâmetros>);  
<declarações de variáveis locais>;  
begin  
  <comandos>;  
end;
```

sendo `procedure` uma palavra-chave que indica o início de uma declaração de procedimento, `<nome>` o nome dado ao procedimento e `<lista_de_parâmetros>`, quando houver, declara a lista de parâmetros do procedimento. A `<declarações de variáveis locais>` declara os nomes e tipos das variáveis locais, ou seja, aquelas cujo escopo ocorre somente dentro do procedimento. Entre o bloco `begin--end` são declarados os `<comandos>` que descrevem o algoritmo do procedimento. Todas as declarações e comandos terminam com o caractere `(;)`.

Uma procedure pode, ou não, possuir parâmetros. Eles podem ser parâmetros de entrada e/ou saída. Os parâmetros devem ser declarados na lista de parâmetros do procedimento.

Exemplo B.15 Forma para declaração de parâmetros

Algoritmo raiz
parâmetros de entrada `a, b, c`
parâmetros de saída `x, y`

```
procedure raiz (a, b: integer; c: real;  
var x: integer; var y: real);
```

Os parâmetros de entrada são declarados sem a palavra-chave `var` e os parâmetros de saída possuem a palavra `var` na declaração. O uso do `var` indica que os parâmetros são passados por referência, isto é, o endereço do parâmetro é passado para o procedimento e, portanto, qualquer alteração que for feita neste parâmetro dentro do procedimento será retornada ao programa que chamou.

Em contraste, se a declaração não possuir o `var`, então a passagem é por valor, ou seja, apenas o valor do parâmetro é passado para o procedimento e, se for alterado, esta alteração não será retornada ao programa que chamou o procedimento.

Para evocar uma procedure a partir de um programa, ou de outra procedure, usa-se um comando da forma

```
<nome_do_procedimento>(<lista_de_parâmetros>);
```

onde `<nome_do_procedimento>` é o mesmo nome usado na declaração do procedimento e `<lista_de_parâmetros>` é uma lista contendo os parâmetros, cujos valores são usados pelo procedimento chamado.

Função

Como visto em vários algoritmos apresentados, freqüentemente, é necessário avaliar funções do tipo $y = f(x)$. Isto pode ser implementado por meio de uma `function` cuja declaração tem o formato geral

```
function <nome_da_função>(<parâmetros_de_entrada>):<tipo>;  
<declarações de variáveis locais>;  
begin  
  <comandos>;  
  <nome_da_função> := <resultado>;  
end;
```

onde `function` é uma palavra-chave para indicar o início da função, `<nome_da_função>` é o nome escolhido, sendo usado para retornar o resultado, e `<parâmetros_de_entrada>` é a declaração da lista de parâmetros, cujos valores serão usados para o cálculo da função. Normalmente, estes parâmetros são passados por valor, ou seja, sem o uso do `var`. O `<tipo>` declara o tipo de resultado que será retornado pelo nome da `function`, por exemplo, `integer`, `real` etc. A `<declarações de variáveis locais>` declara os nomes e tipos das

variáveis que serão usadas somente no escopo da function. As palavras-chaves begin e end delimitam o escopo de comandos e também podem ser usadas para a construção de outros blocos de comandos. Os <comandos> são as estruturas de controle e atribuição utilizadas para implementar o algoritmo da função. O comando

```
<nome_da_função> := <resultado>;
```

deve aparecer na implementação para que o resultado dos cálculos seja atribuído ao nome da função e retornado ao programa que a evocou.

Uma function deve sempre ser chamada a partir de uma expressão aritmética que apareça do lado direito de um comando de atribuição

```
<variável> := <nome_da_função>(<parâmetros_de_entrada>);
```

B.9 Exemplos de programas

Exemplo B.16 Decompor a matriz do Exemplo 2.37 usando um programa implementado a partir do algoritmo da decomposição de Cholesky mostrado na Figura 2.7. Os dados da matriz estão no arquivo matriz.dat e os resultados devem ser gravados no arquivo cholesky.sai.

```
program usa_cholesky;
  { Objetivo: usar o procedimento cholesky }

const nmax = 100;
type Matriz = array [1..nmax,1..nmax] of double;
var
  CondErro, n: integer;
  Det: double;
  A: matriz;

{ procedimento cholesky }
procedure cholesky(n: integer; var A: matriz; var Det: double;
  var CondErro: integer);
{
  Objetivo: fazer a decomposicao de Cholesky de uma matriz A,
  simetrica e definida positiva.
  parametros de entrada: n, A
  ordem e matriz
  parametros de saida: A, Det, CondErro
  fator L escrito sobre A, determinante e condicao de erro
}
var
  i, j, k: integer;
  r, Soma, t: double;
begin
  CondErro := 0;
```

```
Det := 1;
for j := 1 to n do begin
  Soma := 0;
  for k := 1 to j-1 do
    Soma := Soma + sqr(A[j,k]);
  t := A[j,j] - Soma;
  if t > 0 then
    begin
      A[j,j] := sqrt(t);
      r := 1 / A[j,j];
      Det := Det * t;
    end
  else
    begin
      CondErro := 1;
      writeln('a matriz nao e'' definida positiva');
    end;
  for i := j+1 to n do begin
    Soma := 0;
    for k := 1 to j-1 do
      Soma := Soma + A[i,k] * A[j,k];
    A[i,j] := (A[i,j] - Soma) * r;
  end;
end; { procedimento cholesky }

{ modulo principal }
var
  i, j: integer;
  ler, sai: text;
begin
  assign(ler, 'matriz.dat');
  reset(ler);
  assign(sai, 'cholesky.sai');
  rewrite(sai);
  readln(ler, n);
  for i := 1 to n do begin
    for j := 1 to n do
      read(ler, A[i,j]);
  end;
{   chama procedimento para fazer a decomposicao de Cholesky }
  cholesky(n, A, Det, CondErro);
  for i := 1 to n do begin
    for j := 1 to i do
      write(sai, A[i,j]:8:4);
    writeln(sai);
  end;
  write(sai, 'Det = ', Det:10:5, ':5, 'CondErro = ', CondErro:1);
end.
```

A partir do arquivo matriz.dat

```

4
9. 6. -3. 3.
6. 20. 2. 22.
-3. 2. 6. 2.
3. 22. 2. 28.

```

é gerado o arquivo cholesky.sai

```

3.0000
2.0000 4.0000
-1.0000 1.0000 2.0000
1.0000 5.0000 -1.0000 1.0000
Det = 576.00000 CondErro = 0

```

O comando const é usado para dar um valor inicial à uma variável, a qual não pode ter o seu valor alterado.

Exemplo B.17 Calcular a taxa de juros do Plano 1 do Exemplo de aplicação 6.7.1 utilizando o método pégaso descrito na Figura 6.15. Os dados de entrada estão no arquivo juros.dat e os resultados devem ser mostrados na tela.

```

program calcula_juros;
  { Objetivo: calcular a taxa de juros }
var
  CondErro, Iter, IterMax, Prazo: integer;
  A, Avista, B, Entrada, Fator, Juros, Mensalidade, Toler: double;

  { definicao da funcao f(x) }
function Funcao(x:double): double;
begin
  Funcao := (1-exp(-ln(1+x)*Prazo))/x - Fator;
end;

  { procedimento pegaso }
procedure pegaso(A, B, Toler: double; IterMax: integer; var Raiz: double;
  var Iter, CondErro: integer);
{
  parametros de entrada: A, B, Toler, IterMax
  limite inferior, limite superior, tolerancia e
  numero maximo de iteracoes
  parametros de saida: Raiz, Iter, CondErro
  raiz, numero de iteracoes gastas e condicao de erro
}
var
  DeltaX, Fa, Fb, Fx, X: double;
begin
  Fa := Funcao(A);
  Fb := Funcao(B);
  X := B;
  Fx := Fb;
  Iter := 0;

```

```

writeln(' :15,'Calculo de raiz de equacao pelo metodo: pegaso');
write('iter', ' :4,'a', ' :9,'Fa', ' :8,'b', ' :9,'Fb', ' :8);    E   S   rest
writeln('x', ' :8,'Fx', ' :10,'Delta_x');
DeltaX := -Fx / (Fb - Fa) * (B - A);
X := X + DeltaX;
Fx := Funcao(X);
write(Iter:3, A:10:5, Fa:10:5, B:10:5, Fb:10:5);
writeln(X:10:5, ' :2, Fx:10, ' :2, DeltaX:10);
while (((abs(DeltaX) > Toler) or (abs(Fx) > Toler)) and (Iter < IterMax)) do
begin
  if Fx*Fb < 0 then
    begin
      A := B;
      Fa := Fb;
    end
  else
    Fa := Fa * Fb / (Fb + Fx);
  B := X;
  Fb := Fx;
  Iter := Iter + 1;
  DeltaX := -Fx / (Fb - Fa) * (B - A);
  X := X + DeltaX;
  Fx := Funcao(X);
  write(Iter:3, A:10:5, Fa:10:5, B:10:5, Fb:10:5);
  writeln(X:10:5, ' :2, Fx:10, ' :2, DeltaX:10);
end;
Raiz := X;
{ teste de convergencia }
if ((abs(DeltaX) <= Toler) and (abs(Fx) <= Toler)) then
  CondErro := 0
else
  CondErro := 1;
end; { procedimento pegaso }

{ modulo principal }
var
  ler: text;
begin
  assign(ler, 'juros.dat');
  reset(ler);
  readln(ler, Avista, Entrada, Mensalidade, Prazo);
  readln(ler, A, B, Toler, IterMax);
  Fator := (Avista - Entrada) / Mensalidade;
  { chama procedimento para calcular o zero da funcao }
  pegaso(A, B, Toler, IterMax, Juros, Iter, CondErro);
  writeln('Juros = ', Juros:9:5, ' CondErro = ', CondErro:1);
end.

```

A partir do arquivo juros.dat

```

1100.    100.    224.58     6
0.05     0.1     1e-5     100

```

são obtidos os resultados

```

Calculo de raiz de equacao pelo metodo pegaso
iter   a      Fa      b      Fb      x      Fx      Delta_x
0    0.05000  0.62294  0.10000 -0.09750  0.09323 -9.758e-03 -6.766e-03
1    0.05000  0.56626  0.09323 -0.00976  0.09250 -9.373e-05 -7.324e-04
2    0.05000  0.56088  0.09250 -0.00009  0.09249  1.384e-07 -7.101e-06
Juros =  0.09249  CondErro = 0

```

Apêndice C

Linguagem MATLAB

MATLAB (R), acrônimo de MATrix LABoratory [32], é um sistema interativo e linguagem de programação para computação numérica e visualização para as áreas técnicas e científicas. MATLAB é marca registrada da The MathWorks, Inc. (<http://www.mathworks.com>).

Será mostrado, a seguir, como implementar os algoritmos descritos na notação proposta na Seção 1.2 na linguagem de programação do MATLAB. O manual deve ser consultado para mais informações.

C.1 Estrutura de um programa MATLAB

O MATLAB é usado interativamente, ou seja, dado um comando, este é interpretado fornecendo-se os resultados solicitados. Uma lista desses comandos pode ser escrita em um arquivo para uma execução posterior. Esse arquivo tem que ter a extensão .m para ser reconhecido pelo MATLAB. Um programa deve conter comandos da linguagem e chamadas de subprograma function, que será visto na Seção C.8.

Um arquivo tem que conter apenas um programa; caso ele chame alguma function, esta tem que estar em um outro arquivo. Todos os valores numéricos são considerados de ponto flutuante de 8 bytes, não sendo possível fazer as declarações de variáveis. Já os valores literais ocupam 2 bytes cada.

C.2 Variáveis e comentários

As variáveis são representadas por identificadores compostos por cadeias de caracteres alfanuméricos. Os elementos de vetores e matrizes são referenciados por subscritos ou índices.

Exemplo C.1 Sejam as variáveis e suas declarações

| | |
|-----------------------|----------------|
| <i>Tempo</i> | Tempo |
| <i>v_i</i> | <i>v(i)</i> |
| <i>M_{ij}</i> | <i>M(i, j)</i> |

Um comentário é um texto inserido em qualquer parte do algoritmo para aumentar a sua clareza. Um texto precedido pelo caractere % será considerado um comentário.

Exemplo C.2 Seja o texto de comentário

| | |
|---------------------|-------------------|
| { Cálculo da raiz } | % Calculo da raiz |
|---------------------|-------------------|

C.3 Expressões e comando de atribuição

São aceitas três tipos de expressões: aritméticas, lógicas e literais.

Expressões aritméticas

Uma expressão aritmética é composta por operadores e operandos aritméticos. Os operadores são de adição (+), subtração (-), multiplicação (*) e divisão (/). Os operandos podem ser constantes, variáveis e funções aritméticas. Normalmente, uma expressão é utilizada em um comando de atribuição e, durante a execução do comando, a expressão é avaliada e o resultado, atribuído a uma variável. O comando de atribuição tem o formato

<variável> = <expressão_aritmética>

onde o símbolo formado pelo caractere (=) é o operador de atribuição.

Exemplo C.3 Sejam um trecho de algoritmo e sua implementação

| |
|---|
| $\text{velocidade} \leftarrow \text{deslocamento}/\text{tempo}$ $\text{ângulo} \leftarrow \cos(2 + x)$ $\text{delta} \leftarrow \text{raiz}_2(b^2 - 4 * a * c)$ |
|---|

```
velocidade = deslocamento/tempo
ângulo = cos(2+x);
delta = sqrt(b^2-4*a*c)
```

A potenciação é representada pelo caractere (^), sendo b^2 implementado como b^2 e a^{i+1} por $a^{(i+1)}$. O caractere (;) faz com que o valor da variável não seja exibido após a expressão lhe ter sido atribuída. A Tabela C.1 apresenta algumas funções matemáticas do MATLAB. Para obter mais informações sobre as funções do MATLAB, usar o comando help.

Expressões lógicas

Expressão lógica é aquela cujos operadores são lógicos e cujos operandos são relações e/ou variáveis do tipo lógico. O resultado de uma relação ou de uma expressão lógica é verdadeiro

Tabela C.1 Algumas funções matemáticas elementares do MATLAB.

| Função | Descrição | Função | Descrição |
|-----------------|----------------------------------|--------|---------------------------------|
| Trigonométricas | | | |
| acos | arco co-seno | cos | co-seno |
| acot | arco co-tangente | cot | co-tangente |
| acsc | arco co-secante | csc | co-secante |
| asec | arco secante | sec | secante |
| asin | arco seno | sin | seno |
| atan | arco tangente | tan | tangente |
| Exponenciais | | | |
| exp | exponencial | log10 | logaritmo decimal |
| log | logaritmo natural | sqrt | raiz quadrada |
| acosh | arco co-seno hiperbólico | cosh | co-seno hiperbólico |
| acoth | arco co-tangente hiperbólica | coth | co-tangente hiperbólica |
| acsch | arco co-secante hiperbólica | csch | co-secante hiperbólica |
| asech | arco secante hiperbólica | sech | secante hiperbólica |
| asinh | arco seno hiperbólico | sinh | seno hiperbólico |
| atanh | arco tangente hiperbólica | tanh | tangente hiperbólica |
| Complexas | | | |
| abs | valor absoluto | imag | parte imaginária do complexo |
| angle | ângulo de fase | real | parte real do complexo |
| conj | complexo conjugado | | |
| Numéricas | | | |
| ceil | arredonda em direção a $+\infty$ | lcm | mínimo múltiplo comum |
| fix | arredonda em direção a 0 | rem | resto de divisão |
| floor | arredonda em direção a $-\infty$ | round | arredonda em direção ao inteiro |
| gcd | máximo divisor comum | sign | mais próximo |
| | | | sinal |

ou falso; contudo, no MATLAB, o resultado é numérico, e 1 significa verdadeiro e 0 significa falso. Os operadores relacionais do MATLAB são mostrados na Tabela C.2. Notar que (=) é usado para atribuição de um valor a uma variável, enquanto (==) é usado para comparação de igualdade.

Tabela C.2 Operadores relacionais do MATLAB.

| Operador relacional | Descrição |
|---------------------|------------------|
| > | maior que |
| \geq | maior ou igual a |
| < | menor que |
| \leq | menor ou igual a |
| \neq | igual a |
| $\sim=$ | diferente de |

Os operadores lógicos permitem a combinação ou negação das relações lógicas e são listados na Tabela C.3.

Tabela C.3 Operadores lógicos do MATLAB.

| Operador lógico | Descrição | Uso |
|-----------------|-----------|-----------|
| & | e | conjunção |
| | ou | disjunção |
| ~ | não | negação |

Exemplo C.4 Sejam um trecho de algoritmo e a sua implementação

```
a ← 10
t ← verdadeiro
w ← (a == 5) & t
```

```
a = 10
t = 1
w = (a == 5) & t
```

Em uma expressão lógica mais complexa, as relações devem ser colocadas entre parênteses, por exemplo, $(a \geq b) \& (c < 0) | \sim(d == 0)$. As expressões lógicas podem ser usadas tanto para expressar condições em estruturas condicionais ou em estruturas de repetição quanto em comandos de atribuição do tipo

`<variável_lógica> = <expressão_lógica>.`

Expressões literais

Uma variável literal contém uma cadeia de caracteres em vez de um número. A cadeia de caracteres deve ser delimitada por apóstrofos ' ' para ser atribuída a uma variável literal.

Exemplo C.5 Atribuição de uma variável literal

```
mensagem ← "matriz singular"
```

```
mensagem = 'matriz singular'
```

As cadeias de caracteres podem ser concatenadas por meio da função literal `strcat`, como, por exemplo, `strcat('numero', 'complexo')` resulta em 'numero complexo'. A função `length(r)` determina o comprimento da cadeia de caracteres `r`, ou seja, `length('texto')` resulta em 5. A função `findstr(r,s)` retorna o índice do primeiro caractere da cadeia `r` igual à subcadeia `s`, assim, `findstr('abcd', 'bc')` tem como resultado 2.

C.4 Comandos de entrada e saída

Leitura

A leitura de dados pelo teclado é feita pelo comando `input`

Exemplo C.6 Ler as variáveis *Indice* e *Raiz* pelo teclado

leia Indice, Raiz

```
Indice = input('entre com Indice:')
Raiz = input('valor de Raiz:')
```

Quando o comando `input` é executado, a cadeia de caracteres entre os apóstrofos é exibida na tela e a execução do programa é interrompida, aguardando a digitação de um valor. Isto feito, este valor é atribuído à variável e a execução do programa continua.

O comando `fscanf` efetua a leitura de dados formatados em um arquivo. Sua sintaxe é

```
[<variável>, <tamanho>] = fscanf(<fid>, <formato>, <elementos>)
```

onde `<fid>` é o identificador associado ao arquivo no qual está sendo feita a leitura dos dados escritos no formato especificado na cadeia de caracteres `<formato>`. Os dados são convertidos segundo o `<formato>` e atribuídos à `<variável>`. As especificações de conversão são mostradas na Tabela C.5 e o parâmetro `<elementos>` é descrito na Tabela C.4. A variável `<tamanho>` retorna o número de elementos que foram lidos com sucesso.

Tabela C.4 Especificação de elementos para leitura no MATLAB.

| elementos | Especificação |
|--------------------|---|
| <code>n</code> | Lê até <code>n</code> valores em um vetor coluna. |
| <code>inf</code> | Lê até o fim do arquivo, resultando em um vetor coluna contendo o mesmo número de elementos do arquivo. |
| <code>[m,n]</code> | Lê os valores suficientes para preencher uma matriz de dimensão <code>m × n</code> , preenchendo esta matriz por coluna; <code>n</code> pode ser igual à <code>inf</code> mas não pode ser igual à <code>m</code> . |

Quando o MATLAB estiver lendo um arquivo, ele tentará combinar os dados no arquivo com a forma especificada em `<formato>`. Se a combinação ocorrer então os dados serão atribuídos por coluna à `<variável>`. No entanto, se somente uma combinação parcial ocorrer, então apenas os dados combinados com o `<formato>` serão atribuídos à `<variável>` e a operação de leitura será interrompida.

Exibição

O comando `disp` é utilizado para exibir o valor de variáveis dentro do programa

Exemplo C.7 Exibir as variáveis *m* e *n*

escreva m, n

disp([m, n])

O comando `fprintf` grava dados formatados em uma unidade de saída e sua sintaxe é

```
<tamanho> = fprintf(<fid>, <formatos>, <lista_variáveis>)
```

sendo **<tamanho>**, opcional, o número de *bytes* escritos, **<fid>** o identificador associado a uma unidade de saída na qual será feita a exibição dos valores contidos na **<lista_variáveis>** com os formatos especificados na cadeia de caracteres **<formatos>**, a qual deve conter caracteres alfanuméricos e/ou especificações de conversão.

Essas especificações de conversão são delimitadas pelo caractere % e uma das letras i, e, f, g ou s, de acordo com a Tabela C.5. Se **<fid>** for omitido do comando **fprintf** ou se **<fid> = 1**, então os valores serão exibidos na tela.

Tabela C.5 Formatos de exibição do MATLAB.

| Formato | Especificação |
|---------|--|
| %ni | usado para valores inteiros, sendo <i>n</i> o tamanho do campo de exibição; |
| %n.df | notação na forma [-]888.88, sendo <i>n</i> o tamanho do campo (número total de caracteres exibidos) e <i>d</i> o número de dígitos decimais; |
| %n.de | notação na forma [-]8.88e±88, sendo <i>n</i> o tamanho do campo (número total de caracteres exibidos) e <i>d</i> o número de dígitos decimais; |
| %n.dg | equivalente a %n.de ou %n.df, dependendo de qual formato for mais curto; além disso, os zeros insignificantes não são exibidos; |
| %ns | exibe caracteres em um campo de tamanho <i>n</i> . |

Exemplo C.8 O trecho de programa

```
a = 123.456789;
b = -12345.6789;
c = 1;
fprintf('a = %6.2f b = %10.3e c = %2i\n', a, b, c)
```

produz os resultados

```
a=123.46 b=-1.235e+04 c=1
```

onde o caractere ^ representa um espaço em branco¹.

Uma seqüência de caracteres colocada entre apóstrofos é exibida como ela é. Quando for necessário ter o caractere (') mostrado, basta usá-lo duas vezes.

Exemplo C.9 Exibição do caractere '

```
fprintf('a matriz e'' singular\n')
```

produz

¹O caractere não é impresso, ele foi colocado no texto apenas para representar os espaços em branco.

a matriz e' singular

Uso de arquivo

A transferência de dados entre o espaço de trabalho e algum dispositivo de entrada e saída (arquivo em disco, impressora etc.) aumenta a utilização do MATLAB visto tornar possível, por exemplo, até a troca de informações com um outro programa.

O comando **fopen** abre um arquivo ou obtém informações sobre os arquivos abertos. Sua sintaxe é

```
[<fid>, <mensagem>] = fopen(<nome_do_arquivo>, <permissão>)
```

Deste modo, o comando **fopen** associa o nome externo do arquivo **<nome_do_arquivo>** à unidade **<fid>** que será utilizada nos comandos de entrada e saída no modo especificado pela **<permissão>**. Os caracteres possíveis para **<permissão>** estão listados na Tabela C.6.

Tabela C.6 Atributos de arquivo do MATLAB.

| <permissão> | Especificação |
|-------------|--|
| 'r' | Abre o arquivo para leitura. |
| 'r+' | Abre o arquivo para leitura e escrita, mas não cria o arquivo. |
| 'w' | Abre o arquivo para escrita e caso necessário cria o arquivo. Porém, remove o conteúdo do arquivo existente. |
| 'w+' | Abre o arquivo para leitura e escrita e se necessário cria o arquivo. Todavia, remove o conteúdo do arquivo existente. |
| 'a' | Cria e abre um arquivo novo ou abre um arquivo já existente para escrita, anexando ao final do arquivo. |
| 'a+' | Cria e abre um arquivo novo ou abre um arquivo já existente para leitura e escrita, anexando ao final do arquivo. |

Se **<permissão>** for omitida, então será assumido o valor 'r'. Se não for especificado, os arquivos serão abertos em modo binário. Para abrir um arquivo texto, o caractere 't' deve ser adicionado ao caractere de permissão, como em 'wt' e 'rt+'. Similarmente, o caractere 'b' pode ser usado para reiterar que um arquivo deve ser binário.

Caso o comando **fopen** tenha sucesso ao abrir o arquivo, ele retornará o identificador de arquivo **<fid>** contendo um número inteiro maior do que 2 e o conteúdo de **<mensagem>** será vazio. O **<fid>** é usado com outras rotinas de entrada e saída para identificar o arquivo no qual as operações serão realizadas.

No entanto, se o comando **fopen** não tiver sucesso, então **<fid> = -1** e **<mensagem>** conterá uma cadeia de caracteres informando o tipo de erro ocorrido. O comando **help fopen** pode ser utilizado para obter mais informações sobre este comando.

O comando `fclose(<fid>)` fecha o arquivo previamente aberto pelo comando `fopen`, cujo identificador associado a este arquivo seja `<fid>`. O valor 0 é retornado em caso de sucesso no fechamento e -1 no caso de insucesso. Por sua vez, o comando `fclose('all')` fecha todos os arquivos abertos. Quando um arquivo é fechado, a associação entre o identificador `<fid>` e o arquivo físico `<nome_do_arquivo>` é desfeita.

O comando `fscanf` do MATLAB difere de seu homônimo da linguagem C em um aspecto muito importante: ele é *vetorizado* ao retornar um argumento matriz. Isso significa que o `<formato>` é reciclado através do arquivo até o final deste arquivo ser encontrado ou a quantidade de dados definida em `<tamanho>` tiver sido lida. Por exemplo, seja o arquivo `sqrtemp.dat`

```
>> type sqrtemp.dat % lista o arquivo sqrtemp.dat
1.00 1.00000 0.3679
1.20 1.09545 0.3012
1.40 1.18322 0.2466
1.60 1.26491 0.2019
1.80 1.34164 0.1653
2.00 1.41421 0.1353
>> ler = fopen('sqrtemp.dat','r') % abre arquivo para leitura
fid =
3
>> [A,n] = fscanf(ler,'%5f%10f%10f',[3 inf]) % leitura das 3 colunas
A =
1.0000 1.2000 1.4000 1.6000 1.8000 2.0000
1.0000 1.0955 1.1832 1.2649 1.3416 1.4142
0.3679 0.3012 0.2466 0.2019 0.1653 0.1353
n =
18
```

Se o caractere * for colocado entre o % e um caractere de conversão (d, e, f, g ou s), então o correspondente valor combinado não será armazenado na `<variável>`. O comando `frewind(<fid>)` é usado para posicionar o acesso ao primeiro registro do arquivo cujo identificador seja `<fid>`.

```
>> frewind(ler) % posiciona a leitura para o inicio do arquivo
>> [A,n] = fscanf(ler,'%5f%10f%*10f',[2 inf]) % leitura de 2 colunas
A =
1.0000 1.2000 1.4000 1.6000 1.8000 2.0000
1.0000 1.0955 1.1832 1.2649 1.3416 1.4142
n =
12
>> fclose(ler) % fecha o arquivo
ans =
0
```

Durante o processo de leitura, é importante verificar se o último registro do arquivo já foi lido. O comando `feof(<fid>)` faz esta verificação no arquivo de identificador `<fid>`. Se o último registro já foi lido, então será retornado o valor 1; caso contrário, 0 será retornado.

De modo similar ao `fscanf`, o comando `fprintf` do MATLAB difere de seu homônimo da linguagem C, ele também é *vetorizado* no caso da variável a ser exibida ser uma matriz. Neste caso, o formato é aplicado aos elementos da matriz, por coluna, até a última linha. Ele é, então, usado de modo similar, sem reinicialização, pelos demais elementos das outras colunas da matriz. O trecho de programa mostra a diferença entre os dois comandos `fprintf`, usando a matriz M

```
>> x = 1:0.5:2;
>> M = [x; sqrt(x)];
M =
1.0000 1.0000
1.5000 1.2247
2.0000 1.4142
>> fprintf('%5.2f %10.5f \n',M)
1.00 1.50000
2.00 1.00000
1.22 1.41421
>> fprintf('%5.2f %10.5f \n',M')
1.00 1.00000
1.50 1.22474
2.00 1.41421
```

Usar o comando `help iofun` para conhecer outras funções do MATLAB para leitura e gravação de dados.

Exemplo C.10 Seja o algoritmo para conversão de graus Fahrenheit para Celsius

| |
|---|
| Algoritmo Converte_grau { Objetivo: Converter grau Fahrenheit para Celsius } leia Fahrenheit Celsius ← (Fahrenheit - 32) * 5 / 9 escreva Fahrenheit, Celsius finalalgoritmo |
|---|

e o arquivo `fahrenheit.dat` contendo o valor de grau Fahrenheit.

212.0

O programa faz a leitura do grau no arquivo `fahrenheit.dat`

```
% programa Converte_grau
% Objetivo: converter grau Fahrenheit para Celsius
ler = fopen('fahrenheit.dat', 'rt');
sai = fopen('celsius.sai', 'wt');
[Fahrenheit, tam] = fscanf(ler, '%5f', inf);
Celsius = (Fahrenheit - 32) * 5 / 9;
fprintf(sai, '%8.3f Fahrenheit = %8.3f Celsius', Fahrenheit, Celsius);
fclose(sai);
```

e grava o arquivo celsius.sai, cujo conteúdo é
212.000 Fahrenheit = 100.000 Celsius

C.5 Estruturas condicionais

Estrutura condicional simples

É implementada por meio da estrutura if--end

```
se <condição> então
  <comandos>
fimse
```

```
if <condição>
  <comandos>
end
```

onde if e end são palavras-chave, <condição> é uma expressão lógica que se for verdadeira faz com que a lista <comandos> seja executada.

Exemplo C.11 Implementar o algoritmo para calcular o logaritmo decimal de um número positivo.

```
Algoritmo Logaritmo_decimal
{ Objetivo: Calcular logaritmo decimal }
leia x
se x > 0 então
  LogDec ← log10(x)
  escreva x, LogDec
fimse
finalgoritmo
```

```
% programa Logaritmo_decimal
% Objetivo: calcular logaritmo decimal
x = input('Entre com x: ');
if x > 0
  LogDec = log10(x);
  disp([x, LogDec])
end
```

Estrutura condicional composta

A sua implementação é feita pela estrutura if--else--end

```
se <condição> então
  <comandos_1>
senão
  <comandos_2>
fimse
```

```
if <condição>
  <comandos1>
else
  <comandos2>
end
```

sendo if, else e end palavras-chave, <condição> uma expressão lógica que, caso seja verdadeira, faz com que a sequência <comandos1> seja executada e a sequência <comandos2> não seja executada. Se o resultado de <condição> for falso, então somente a lista <comandos2> será executada.

Exemplo C.12 Implementar um algoritmo para avaliar a função modular $f(x) = |2x|$.

```
Algoritmo Função_modular
{ Objetivo: Avaliar uma função modular }
leia x
se x ≥ 0 então
  fx ← 2 * x
senão
  fx ← -2 * x
fimse
escreva x, fx
finalgoritmo
```

```
% programa Função_modular
% Objetivo: avaliar uma função modular
x = input('Entre com x: ');
if x >= 0
  fx = 2 * x;
else
  fx = -2 * x;
end
disp([x, fx])
```

C.6 Estruturas de repetição

Número indefinido de repetições

Utiliza-se a estrutura while--end com um break

```
repita
  <comandos_1>
  se <condição> então
    interrompa
  fimse
  <comandos_2>
fimrepita
<comandos_3>
```

```
while 1
  <comandos1>
  if <condição>
    break
  end
  <comandos2>
end
<comandos3>
```

onde while e break são palavras-chave. O comando break faz com que a execução do while seja interrompida. As listas <comandos1> e <comandos2> serão repetidas até que a expressão lógica <condição> torne verdadeiro. Quando isso ocorrer, a repetição será interrompida (<comandos2> não será executada) e a lista <comandos3> passa a ser executada.

Exemplo C.13 Implementar o algoritmo para determinar o maior número de ponto flutuante que somado a 1 seja igual a 1.

```
Algoritmo Epsilon
{ Objetivo: Determinar a precisão da máquina }
Epsilon ← 1
repita
    Epsilon ← Epsilon/2
    se Epsilon + 1 = 1 então
        interrompa
    fimse
fimrepita
escreva Epsilon
finalgoritmo
```

```
% programa preciso
% Objetivo: determinar a precisão da máquina
Epsilon = 1;
while 1
    Epsilon = Epsilon / 2;
    if Epsilon + 1 == 1
        break
    end
end
disp(Epsilon)
```

A estrutura while <expressão_lógica>--end continuará sendo repetida enquanto o valor da <expressão_lógica> for verdadeiro. Nesse exemplo, como a expressão será sempre verdadeira (1), então é o comando break que faz com que a execução da estrutura de repetição seja interrompida, transferindo a execução para o comando fora da estrutura. ■

Número definido de repetições

É implementada pela estrutura for--end, com a variável de controle sendo definida como um vetor

```
para <controle> ← <valor-inicial> até <valor-final> passo <delta> faça
    <comandos>
fimpara
```

```
for <controle> = <define_vetor>
    <comandos>
end
```

com for e end sendo palavras-chaves. A variável de controle <controle> assume todos os valores obtidos pela expressão <define_vetor> usada para gerar um vetor.

Exemplo C.14 Implementar o algoritmo para mostrar que a soma dos n primeiros números ímpares é igual ao quadrado de n .

```
Algoritmo Primeiros_imparés
{ Objetivo: Verificar propriedade dos números ímpares }
leia n
Soma ← 0
para i ← 1 até 2 * n - 1 passo 2 faça
    Soma ← Soma + i
fimpara
escreva Soma, n2
finalgoritmo
```

```
% programa Primeiros_imparés
% Objetivo: verificar propriedade dos números ímpares
n = input('Entre com n: ');
Soma = 0;
for i = 1:2:2*n-1
    Soma = Soma + i;
end
disp([Soma, n^2])
```

C.7 Falha no algoritmo

No caso de inconsistência nos parâmetros de entrada ou de uma operação que causaria uma operação inválida, utiliza-se o comando error

```
escreva "grau menor que 1"
abandone
error('grau menor que 1')
```

O comando error exibe a cadeia de caracteres entre apóstrofos e abandona imediatamente o programa ou a function.

C.8 Subprogramas

Um subprograma é definido como uma function com a sintaxe

```
function [<parâmetros_saida>] = <nome>(<parâmetros_entrada>)
    <comandos>
```

onde function é uma palavra-chave que determina o início físico do subprograma, <nome> é o identificador, <parâmetros_entrada> é uma lista contendo os parâmetros de entrada e <parâmetros_saida> especifica as variáveis que retornam da function. Os <comandos> são estruturas que implementam os algoritmos.

Exemplo C.15 Forma para declaração de parâmetros

Algoritmo raiz
 parâmetros de entrada a, b, c
 parâmetros de saída x, y

$[x, y] = \text{raiz}(a, b, c)$

O nome da função tem que ser igual ao nome do arquivo M onde ela está definida, mas sem a extensão .m, ou seja, a função *raiz* mencionada acima deve estar no arquivo *raiz.m*. Um arquivo pode conter mais que uma *function*, e a primeira delas tem que ter o nome desse arquivo. As outras *function's* só podem ser chamadas pela primeira delas, não sendo possível serem chamadas por *function's* escritas em um outro arquivo. Na realidade, várias funções do MATLAB são arquivos M. Ao contrário do programa, no qual as variáveis são globais, em uma *function* as variáveis são locais, ou seja, elas não têm acesso e nem podem criar variáveis no espaço de trabalho. Por sua vez, os parâmetros de saída (x, y no caso da *function* *raiz*) são criados no espaço de trabalho.

Em vários algoritmos descritos, há a necessidade de especificar uma função $f(x)$. No MATLAB, $f(x)$ pode ser definida por meio de uma variável literal contendo uma expressão aritmética que é, posteriormente, avaliada por meio do comando *eval*.

O comando *eval(<expressão>)* interpreta a expressão definida pela cadeia de caracteres *<expressão>*

```
>> x = 2; y = eval('3*x+sqrt(x+2)')
y =
  8
```

Para avaliar uma expressão que possui a variável *x*, esta tem que ser previamente definida.

Exemplo C.16 Considere um trecho do algoritmo da Figura 5.7, na página 227

Algoritmo Newton-Cotes
 { Objetivo: Integrar uma função pelo método de Newton-Cotes }
 parâmetros de entrada a, b, n, m
 { limite inferior, limite superior, grau do polinômio, número de subintervalos }
 parâmetros de saída *Integral, CondErro*
 { valor da integral e condição de erro, sendo }
 { *CondErro* = 0 se não houve erro de consistência dos parâmetros dados, }
 { *CondErro* = 1 se ($n < 1$ ou $n > 8$), }
 { *CondErro* = 2 se resto(m, n) ≠ 0 e }
 { *CondErro* = 3 se ambas as condições ocorrerem }
 ...
 $x \leftarrow a + i * h$
 $y \leftarrow f(x) \quad \{ \text{avaliar a função integrando em } x \}$

Para implementar este algoritmo, define-se o parâmetro de entrada extra *funcao*, que é uma variável literal especificando $f(x)$

```
function [Integral, CondErro] = newcotes(funcao, a, b, n, m)
...
x = a + i * h;
y = eval(funcao);
```

Para calcular $\int_0^1 \cos(x)e^{x+1} dx$, pela regra do 1/3 de Simpson com quatro subintervalos, basta definir esta chamada da *function* *newcotes*

```
>> [Integral, CondErro] = newcotes('cos(x)*exp(x+1)', 0, 1, 2, 4)
```

C.9 Exemplos de programas

Exemplo C.17 Decompor a matriz do Exemplo 2.37 usando um programa implementado a partir do algoritmo da decomposição de Cholesky mostrado na Figura 2.7. Os dados da matriz estão no arquivo *matriz.dat* e os resultados devem ser gravados no arquivo *cholesky.sai*. O programa

```
% programa usa_cholesky
ler = fopen('matriz.dat', 'rt');
sai = fopen('cholesky.sai', 'wt');
n = fscanf(ler, '%3i', 1);
for i = 1:n
    linha = fscanf(ler, '%5f', [1,n]);
    A(i,:) = linha;
end
% chama funcao para fazer a decomposicao de Cholesky
[A, Det, CondErro] = cholesky(A);
for i = 1:n
    for j = 1:i
        fprintf(sai, '%8.4f', A(i,j));
    end
    fprintf(sai, '\n');
end
fprintf(sai, 'Det = %10.5f      CondErro = %li\n', Det, CondErro);
fclose(sai);
```

e o subprograma

```
function [A, Det, CondErro] = cholesky(A)
n = length(A); CondErro = 0; Det = 1;
for j = 1:n
    Soma = 0;
    for k = 1:j-1
        Soma = Soma + A(j,k)^2;
```

```

end
t = A(j,j) - Soma;
if t > 0
    A(j,j) = sqrt( t ); r = 1 / A(j,j); Det = Det * t;
else
    CondErro = 1; error('a matriz nao e'' definida positiva')
end
for i = j+1:n
    Soma = 0;
    for k = 1:j-1
        Soma = Soma + A(i,k) * A(j,k);
    end
    A(i,j) = ( A(i,j) - Soma ) * r;
end

```

gravados em arquivos separados, e lendo o arquivo matriz.dat

```

4
9.   6.  -3.   3.
6.  20.   2.  22.
-3.   2.   6.   2.
3.  22.   2.  28.

```

geram o arquivo cholesky.sai

```

3.0000
2.0000  4.0000
-1.0000  1.0000  2.0000
 1.0000  5.0000 -1.0000  1.0000
Det = 576.00000  CondErro = 0

```

Como o comando fscanf faz a leitura de matriz de um modo *vetorizado*, cada linha da matriz foi lida como um vetor e atribuído à linha correspondente da matriz. ■

Exemplo C.18 Calcular a taxa de juros do Plano 1 do Exemplo de aplicação 6.7.1 utilizando o método pégaso descrito na Figura 6.15. Os dados de entrada estão no arquivo juros.dat e os resultados devem ser mostrados na tela. O programa

```

% programa calcula_juros
ler = fopen('juros.dat', 'rt');
linha = fscanf(ler, '%10f', 4);
Avista = linha(1); Entrada = linha(2);
Mensalidade = linha(3); Prazo = linha(4);
linha = fscanf(ler, '%10f', 4);
A = linha(1); B = linha(2);
Toler = linha(3); IterMax = linha(4);
Fator = (Avista - Entrada) / Mensalidade;
% define a funcao como uma cadeia de caracteres
funcao = ['(1-(1+x)^(Prazo))/x', num2str(Fator,10)];
% chama subprograma para calcular o zero da funcao
[Raiz, Iter, CondErro] = pegaso(funcao, A, B, Toler, IterMax, 0);
fprintf('Juros = %9.5f  CondErro = %i\n', Raiz, CondErro);

```

e o subprograma

```

function [Raiz,Iter,CondErro] = pegaso(funcao,a,b,Toler,IterMax,Exibe)
%PEGASO Calculo de raiz de equacao pelo metodo pegaso.
% parametros de entrada:
%   funcao: cadeia de caracteres que especifica a funcao;
%   a, b: limites inferior e superior do intervalo;
%   Toler: tolerancia no calculo da raiz;
%   IterMax: numero maximo de iteracoes;
%   Exibe: mostrar resultados, sendo
%       Exibe = 0: nao exibir e Exibe = 1: exibir.
% parametros de saida:
%   Raiz: zero da funcao;
%   Iter: numero de iteracoes gastas;
%   CondErro: condicao de erro, sendo
%       CondErro = 0: nao houve erro e
%       CondErro = 1: nao convergiu com os parametros dados.
%
x = a; Fa = eval(funcao); x = b; Fb = eval(funcao);
if Exibe ~= 0
    fprintf('          Calculo de raiz de equacao pelo metodo pegaso\n');
    fprintf('iter      a      Fa      b      Fb      x      Fx');
    fprintf('      Delta_x\n');
end
x = b; Fx = Fb; Iter = 0;
while 1
    DeltaX = -Fx / (Fb - Fa) * (b - a);
    x = x + DeltaX; Fx = eval(funcao);
    if Exibe ~= 0
        fprintf('%3i%10.5f%10.5f%10.5f%10.5f%12.3e%12.3e\n', ...
            Iter, a, Fa, b, Fb, x, Fx, DeltaX)
    end
    if ((abs(DeltaX) <= Toler & abs(Fx) <= Toler) | Iter >= IterMax), break, end
    if Fx*Fb < 0
        a = b; Fa = Fb;
    else
        Fa = Fa * Fb / (Fb + Fx);
    end
    b = x; Fb = Fx; Iter = Iter + 1;
end
Raiz = x;
if abs(DeltaX) <= Toler & abs(Fx) <= Toler
    CondErro = 0;
else
    CondErro = 1;
end

```

gravados em arquivos separados, produzem a partir do arquivo juros.dat

| | | | |
|-------|------|--------|-----|
| 1100. | 100. | 224.58 | 6 |
| 0.05 | 0.1 | 1e-5 | 100 |

os resultados

```

Calculo de raiz de equacao pelo metodo pegaso
iter   a      Fa     b      Fb     x      Fx      Delta_x
 0    0.05000  0.62294  0.10000 -0.09750  0.09323 -9.758e-03 -6.766e-03
 1    0.05000  0.56626  0.09323 -0.00976  0.09250 -9.373e-05 -7.324e-04
 2    0.05000  0.56088  0.09250 -0.00009  0.09249  1.384e-07 -7.101e-06
Juros =  0.09249  CondErro = 0

```

Apêndice D

Respostas dos exercícios

Capítulo 1: Computação numérica

Seção 1.2

1.6) $c \leq 5$ ou $j \neq 1$; $d < 0$ e $m = 3$.

Seção 1.4

1.16) $A(n) = n$.

1.17) $A(n) = n^2$.

1.18) $A(n) = n^3$.

1.19) $A(n) = n^2 + 4n$.

1.20) $A(n) = n^2 + 4n + 2$.

Capítulo 2: Sistemas lineares

Seção 2.1

$$2.1.a) A + B = \begin{bmatrix} 3 & -3 & 9 \\ 5 & 6 & 5 \\ 2 & 4 & 13 \end{bmatrix}.$$

$$2.1.b) Ax = \begin{bmatrix} -4 \\ 27 \\ 25 \end{bmatrix}.$$

$$2.1.c) AB = \begin{bmatrix} -10 & -6 & 21 \\ -1 & 44 & 35 \\ -27 & 44 & 87 \end{bmatrix}.$$

2.1.d) $x^T y = 140$.

$$2.1.e) xy^T = \begin{bmatrix} 10 & 20 & 30 \\ 20 & 40 & 60 \\ 30 & 60 & 90 \end{bmatrix}.$$

2.3.a) $\lambda_1 = 6$, $\lambda_2 = 3$,
traço(A) = 9, det(A) = 18.

2.3.b) $\lambda_1 = 6$, $\lambda_2 = 3$,
traço(A) = 9, det(A) = 18.

2.3.c) $\lambda_1 = 2 + i$, $\lambda_2 = 2 - i$,
traço(A) = 4, det(A) = 5.

2.4.a) $\|x\|_1 = 15$.

2.4.b) $\|x\|_2 = 7,4162$.

2.4.c) $\|x\|_\infty = 5$.

2.5.a) $\|A\|_1 = 19$.

2.5.b) $\|A\|_\infty = 21$.

2.5.c) $\|A\|_F = 16,8819$.

Seção 2.2

$$2.6) x = \begin{bmatrix} -0,5 \\ 0,8 \\ 0,3 \end{bmatrix}.$$

$$2.7) \quad x = \begin{bmatrix} 4 \\ 0 \\ -1 \end{bmatrix}.$$

$$2.8) \quad x = \begin{bmatrix} -2,2619 \\ -5,6667 \\ 0,5000 \\ 1,6667 \end{bmatrix}.$$

Seção 2.3

$$2.11) \quad x = \begin{bmatrix} -1,000 \\ 3,000 \\ 2,000 \end{bmatrix}, \quad r = \begin{bmatrix} 0,000 \\ 0,000 \\ 0,000 \end{bmatrix}, \\ \det(A) = -175.$$

$$2.12) \quad x = \begin{bmatrix} 7,000 \\ 1,000 \\ 6,000 \end{bmatrix}, \quad r = \begin{bmatrix} 0,000 \\ 0,000 \\ 0,000 \end{bmatrix}, \\ \det(A) = -64.$$

2.13) sem/com pivotação:

$$x = \begin{bmatrix} -4,000 \\ 0,000 \\ 3,000 \\ 1,000 \end{bmatrix}, \quad r = \begin{bmatrix} 0,000 \\ 0,000 \\ 0,000 \\ 0,000 \end{bmatrix}, \\ \det(A) = 2825.$$

2.14) sem pivotação:

$$x = \begin{bmatrix} -0,314 \\ -0,818 \\ 1,753 \\ 0,666 \end{bmatrix}, \quad r = \begin{bmatrix} -0,001 \\ 0,001 \\ 0,001 \\ 0,000 \end{bmatrix},$$

com pivotação:

$$x = \begin{bmatrix} 0,315 \\ -0,819 \\ 1,755 \\ 0,666 \end{bmatrix}, \quad r = \begin{bmatrix} 0,002 \\ -0,001 \\ 0,000 \\ -0,003 \end{bmatrix},$$

$$\det(A) = -401.$$

2.15) com pivotação:

$$x = \begin{bmatrix} 2,346 \\ 4,353 \\ -2,390 \\ -1,768 \\ 2,338 \end{bmatrix}, \quad r = \begin{bmatrix} 0,001 \\ 0,004 \\ 0,007 \\ -0,002 \\ 0,007 \end{bmatrix}, \\ \det(A) = -4990.$$

Seção 2.4

2.16.a) sem pivotação:

$$x = \begin{bmatrix} 2,7354 \\ 0,8500 \\ 1,8569 \end{bmatrix}, \quad r = \begin{bmatrix} -1 \times 10^{-4} \\ -5 \times 10^{-5} \\ 2,1963 \end{bmatrix}.$$

2.16.b) com pivotação:

$$x = \begin{bmatrix} 3,2422 \\ 0,6810 \\ 1,8569 \end{bmatrix}, \quad r = \begin{bmatrix} 0,0003 \\ 0,0003 \\ 0,0001 \end{bmatrix}, \\ \det(A) = 91,03.$$

$$2.17) \quad x = \begin{bmatrix} 1,0000 \\ 5,0000 \\ 2,0000 \end{bmatrix}, \quad r = \begin{bmatrix} 0,0000 \\ 0,0000 \\ 0,0000 \end{bmatrix}, \\ \det(A) = -45.$$

$$2.18) \quad x = \begin{bmatrix} 1,0000 \\ 2,0000 \\ 3,0000 \\ 4,0000 \end{bmatrix}, \quad r = \begin{bmatrix} 0,0000 \\ 0,0000 \\ 0,0000 \\ 0,0000 \end{bmatrix}, \\ \det(A) = -610.$$

$$2.19) \quad x = \begin{bmatrix} 17 - 3\theta \\ 4 - \theta \\ \theta \end{bmatrix}, \quad \det(A) = 0.$$

Infinitas soluções, uma para cada valor de θ .**Seção 2.5**

$$2.21) \quad x = \begin{bmatrix} -1 \\ 0 \\ 2 \end{bmatrix}, \quad r = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \\ \det(A) = 3600.$$

$$2.22) \quad x = \begin{bmatrix} 2 \\ 6 \\ -1 \\ 1 \end{bmatrix}, \quad r = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \\ \det(A) = 36.$$

$$2.23) \quad x = \begin{bmatrix} -2,2487 \\ 7,9688 \\ -0,8588 \\ 3,8150 \\ 0,2450 \end{bmatrix}, \quad r = \begin{bmatrix} 0 \\ 7,11 \times 10^{-15} \\ 7,11 \times 10^{-15} \\ 0 \\ -7,11 \times 10^{-15} \end{bmatrix}, \\ \det(A) = 1600.$$

Seção 2.7

$$2.28) \quad x^0 = \begin{bmatrix} 2,9890 \\ -2,0049 \\ 3,9997 \end{bmatrix}, \quad x^1 = \begin{bmatrix} 2,9999 \\ -2,0000 \\ 4,0000 \end{bmatrix},$$

$$x^2 = \begin{bmatrix} 3,0000 \\ -2,0000 \\ 4,0000 \end{bmatrix}.$$

$$2.29) \quad A^{-1} = \begin{bmatrix} -5,8 & -5,6 & 3,6 \\ -2,2 & -2,4 & 1,4 \\ 5,0 & 5,0 & -3,0 \end{bmatrix}.$$

Seção 2.8

$$2.31) \quad J: x^6 = \begin{bmatrix} 2,9972 \\ -1,0006 \\ 4,9982 \\ 7,0006 \end{bmatrix},$$

$$\text{GS: } x^4 = \begin{bmatrix} 3,0013 \\ -0,9999 \\ 5,0005 \\ 7,0000 \end{bmatrix}.$$

2.32) J: $\rho(J) = 6,4071 \times 10^{-6}$, GS: $\rho(S) = 2$; como $\rho(J) < 1$ e $\rho(S) > 1$, a solução só irá convergir pelo método de Jacobi;
 $x = [1 \ 2 \ 3]^T$.2.33) J: $\rho(J) = 1,1180$, GS: $\rho(S) = 0,5$; como $\rho(J) > 1$ e $\rho(S) < 1$, a solução só irá convergir pelo método de Gauss-Seidel;
 $x = [2 \ 4 \ 6]^T$.**Seção 2.9**

$$2.36) \quad \kappa_1(H_2) = 27; \quad \kappa_2(H_2) = 19,2815; \\ \kappa_\infty(H_2) = 27.$$

2.38)

$$H_4 = \begin{bmatrix} 1,0000 & 0,5000 & 0,3333 & 0,2500 \\ 0,5000 & 0,3333 & 0,2500 & 0,2000 \\ 0,3333 & 0,2500 & 0,2000 & 0,1667 \\ 0,2500 & 0,2000 & 0,1667 & 0,1429 \end{bmatrix},$$

$$H_4^{-1} = \begin{bmatrix} 16 & -120 & 240 & -140 \\ -120 & 1200 & -2700 & 1680 \\ 240 & -2700 & 6480 & -4200 \\ -140 & 1680 & -4200 & 2800 \end{bmatrix}.$$

$$2.40) \quad \|H_4\|_\infty = 2,0833; \\ \|H_4^{-1}\|_\infty = 13620; \quad \kappa_\infty(H_4) = 28375.$$

Gerais

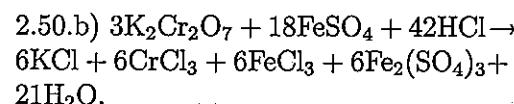
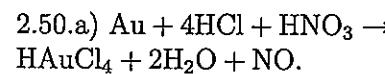
$$2.41) \quad \det(M) = x_0(x_1^2 - x_2^2) - x_1(x_0^2 - x_2^2) + x_2(x_0^2 - x_1^2).$$

$$2.42) \quad \|A\|_2 = \max(\sqrt{\lambda(A^T A)}) = 5,8339.$$

$$2.43.\text{a}) \quad D(\lambda) = \lambda^3 - 2\lambda^2 - 11\lambda + 12.$$

$$2.43.\text{b}) \quad D(\lambda) = \lambda^3 - 10\lambda^2 + 15\lambda + 49.$$

$$2.45) L = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 \\ -1 & 3 & 0 & 0 & 0 \\ 0 & 2 & 4 & 0 & 0 \\ 0 & 0 & 3 & 5 & 0 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$



Capítulo 3: Interpolação polinomial

Seção 3.1

3.1) $P_1(0,33) = 0,3237$.

3.2) $P_2(0,33) = 0,3241$.

3.3) $P_1(0,38) = 0,3706$.

3.4) $P_2(0,38) = 0,3709$.

Seção 3.2

3.6) $L_1(1,1) = 0,9785$.

3.7) $L_2(1,1) = 0,9867$.

3.8) $L_2(1,1) = 0,9867$.

3.9) $L_3(1,2) = 1,1187$.

Seção 3.3

3.11) $P_1(2,1) = 1,0749$; $P_2(2,1) = 1,0752$; $P_3(2,1) = 1,0752$.

Seção 3.4

3.17) $P_1(2,15) = 0,4415$; $P_2(2,15) = 0,4396$; $P_3(2,15) = 0,4393$.

Seção 3.5

3.21) $x = (1,1; 1,3)$.

3.22) $x = (1,1; 1,3; 1,4)$.

3.23) $x = (1,3; 1,4; 1,7; 1,8)$.

3.24) $x = (1,4; 1,7; 1,8)$.

3.25) $x = (1,7; 1,8; 2,0)$.

Seção 3.6

3.26) $P_2(3,5) = 6,5466$.

3.27) $T_2(3,5) = 0,0019$.

3.28) $P_2(3,5) = 6,5485$.

3.29) $T_2(3,5) = -7,8612 \times 10^{-4}$.

Seção 3.11

3.42) $s(1,1) = 1,0433$; $s(2,2) = 1,4870$; $s(4,3) = 2,0727$; $s(5,7) = 2,3889$; $s(8,8) = 2,9660$; $s(8,9) = 2,9830$.

3.43) $s(1,1) = 1,0466$; $s(2,2) = 1,4848$; $s(4,3) = 2,0732$; $s(5,7) = 2,3883$; $s(8,8) = 2,9663$; $s(8,9) = 2,9832$.

Gerais

3.46) $c_p = P_3(250) = 1,158$.

3.47) $\rho = P_2(25) = 13,534$.

3.48.a) $P_1(0,3) = 0,61488$; $P_2(0,3) = 0,61987$; $P_3(0,3) = 0,61894$; $P_4(0,3) = 0,61805$; $P_5(0,3) = 0,61767$; $P_6(0,3) = 0,61763$.

3.48.b) $s(0,3) = 0,61786$.

3.48.c) $s(0,3) = 0,61859$.

3.49) $z = 0,82297$.

3.50) Com ordenadas $x = [-1, 0, 5, 2]$:
 $\psi = 1,5069$.

Capítulo 4: Ajuste de curvas

Seção 4.1

4.2) $p(x) = -2,7143x + 6,4571$; $D = 94,4780$.

4.3) $p(x) = -0,7273x + 4,3273$; $D = 1,9674$.

4.4) $p(x) = -0,9061x + 4,8015$; $D = 1,4776$.

4.5) O melhor modelo é a reta de quadrados mínimos, pois apresenta o menor desvio.

Seção 4.2

4.6) $r^2 = 0,8246$.

4.7) $\sigma^2 = 0,4549$.

4.8) $r^2 = 0,7235$; $\sigma^2 = 0,3096$.

4.9) $r^2 = 0,9660$; $\sigma^2 = 9,5822 \times 10^{-13}$.

4.10) O coeficiente de determinação mede a proporção da variação total dos dados em torno da média de y , que é explicada pelo modelo de regressão.

Seção 4.3

4.11) $u = 4,2393 + 3,4000x_1 - 6,4643x_2$.

4.12) $u = -2,0177 + 11,3315x - 1,2222x^2$.

4.14) $b_0 = 4,23929$; $b_1 = 3,40000$; $b_2 = -6,46429$.

4.15) $b_0 = -2,01765$; $b_1 = 11,3315$; $b_2 = -1,22223$.

Seção 4.4

4.18) $b_0 = 4,23929$; $b_1 = 3,40000$; $b_2 = -6,46429$.

4.19) $b_0 = -2,01765$; $b_1 = 11,3315$; $b_2 = -1,22223$.

4.20) Os resultados são iguais.

Seção 4.5

4.21) $p(x) = 3x + 2$.

4.22) $p(x) = 3x + 2$.

4.23) Sim, porque a reta de quadrados mínimos de grau 1, utilizando 2 pontos, coincide com o polinômio interpolador de grau 1 que passa por estes pontos.

4.25) A interpolação deve ser utilizada quando se necessita de um valor intermediário não constante de uma tabela. A regressão deve ser usada quando se deseja estimar um parâmetro de um modelo semideterminístico e/ou prever um valor dado por este modelo.

Gerais

4.26.a) $\frac{1}{y} = \frac{b}{a} + \frac{c}{a}x$.

4.26.b) $\log_e(y) = \log_e(a) + \log_e(b)x$.

4.26.c) $\frac{1}{y} = \frac{b}{a} + \frac{1}{a}x$.

4.26.d) $\log_e\left(\frac{1}{y} - 1\right) = bx$.

4.27) $\begin{bmatrix} \sum x_{i1}x_{i1} & \sum x_{i2}x_{i1} & \sum x_{i3}x_{i1} \\ \sum x_{i1}x_{i2} & \sum x_{i2}x_{i2} & \sum x_{i3}x_{i2} \\ \sum x_{i1}x_{i3} & \sum x_{i2}x_{i3} & \sum x_{i3}x_{i3} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} \sum x_{i1}y_i \\ \sum x_{i2}y_i \\ \sum x_{i3}y_i \end{bmatrix}$.

4.28) $u = 2,8014x - 3,5908x^2 + 1,8115x^3$.

4.29) O melhor modelo é o (c), pois apresenta o maior coeficiente de determinação e a menor variância residual.

4.30) $y = ab^x$, sendo $a = 55,0700$ e $b = 0,8870$.

Capítulo 5: Integração numérica

Seção 5.1

5.1.a) $I = 0,8595$.

5.1.b) $I = 0,8438$.

5.1.c) $I = 0,8448$.

5.2.a) $P_2(x) = 4,05x^2 + 4,95x + 1$.

5.2.b) $I = 4,7$.

5.2.c) $I = 4,7$.

5.2.d) A regra de 1/3 de Simpson (item c) consiste nas etapas a e b.

5.3) $I = 6,3895$. Este valor foi obtido usando-se a regra de 1/3 de Simpson com $m = 6$.

5.5)

| m | I | m | I |
|-----|---------|-----|---------|
| 2 | 28,5061 | 4 | 28,1170 |
| 6 | 28,0920 | 8 | 28,0876 |

Seção 5.2

5.6) $I = 28,0855$.

5.7) $I = 20,5240$.

5.8) $I = 2,8284$.

Seção 5.3

5.11)

Regra do trapézio: $I = 1,8591$;
Gauss-Legendre: $I = 1,7179$;
valor analítico: $I = 1,7183$.

5.12)

1/3 de Simpson: $I = 28,1170$;
Gauss-Legendre: $I = 28,0855$;
valor analítico: $I = 28,0855$.

5.13)

3/8 de Simpson: $I = 20,5402$;
Gauss-Legendre: $I = 20,5240$;
valor analítico: $I = 20,5240$.

5.14)

Newton-Cotes: $I = 2,8284$;
Gauss-Legendre: $I = 2,8284$;
valor analítico: $I = 2,8284$.

5.15)

Newton-Cotes: $I = 41,1715$;
Gauss-Legendre: $I = 41,1711$;
valor analítico: $I = 41,1711$.

Seção 5.4

5.17) $I = 2,8284271247$.

5.18) $I = 8,4239791591 \times 10^{-1}$.

5.19) $I = 2,7761914467 \times 10^{-1}$.

5.20) $I = -4,7240071835$.

Seção 5.5

5.21) $I = 1,3364$.

5.22) $I = 11,0922$.

5.23) $I = 15,3669$.

Seção 5.6

5.26) $I = 1,7528$.

5.27) $I = 1,0310$.

5.28) $I = 11,5297$.

Seção 5.7

5.31) NC: $I = 0,5000$; GL: $I = 0,2500$.

5.32) NC: $I = 0,3333$; GL: $I = 0,2000$.

5.33) NC: $I = 1,6026$; GL: $I = -0,0134$.

5.34) NC: $I = 0,6250$; GL: $I = 0,6081$.

5.35) NC: $I = 0,9729$; GL: $I = 0,9752$.

Gerais

5.36.a) $L_1(x) = x + 1$.

5.36.b) $I = 1,5$.

5.36.c) $I = 1,5$.

5.36.d) A regra do trapézio (item c) consiste dos itens a e b.

5.39.a) $I = 22$.

5.39.b) Sim. O erro é dado por $E = \frac{-(b-a)^5}{180m^4} f^{iv}(\theta)$. A função que está sendo integrada é um polinômio de grau 3, logo $f^{iv}(x) = 0$.

5.40.a) $m = 9$.

5.40.b) $I = 12,3358$.

5.40.c) $|12,3354 - 12,3358| = 4 \times 10^{-4}$.

5.43.a) $I = 44$.

5.43.b) $I = 15,5556$.

5.43.c) valor exato = 15,5556.

5.43.d) Apenas o resultado obtido pela quadratura de Gauss-Legendre é exato, pois esta é construída de forma a ser exata para polinômios de grau menor ou igual a $2n - 1$. Newton-Cotes com $n_x = n_y = 1$ é obtida integrando-se um polinômio interpolador de grau 1, o que não é suficiente para a obtenção de um valor exato.

5.45) $I = 0,88623$.

Capítulo 6: Raízes de equações

Seção 6.1

6.1) $0,4545 \leq \xi^+ \leq 4,0000$ e $-2,7321 \leq \xi^- \leq -0,5635$;
 $n^+ = 2$ ou 0 e $n^- = 1$.

6.3) $[0,4545 \ 4,0000 \ -2,7321 \ -0,5635]$.

6.5) $\xi_1 \in [-2,74; -0,95]$ para $z = -1$ e $\xi_2 \in [0,95; 5,65]$ para $z = 1$.

Seção 6.2

6.6) $\xi = 0,9536$ em $[0, 2]$.

6.7) $\xi = -0,7594$ em $[-1, 0]$.

6.8) $\xi = 0,5994$ em $[0, 1]$.

Seção 6.3

6.11)

| método | intervalo | raiz | iter |
|--------------|-----------|--------|------|
| secante | [3, 4] | 3,0672 | 3 |
| regula falsi | [3, 4] | 3,0672 | 12 |
| pégaso | [3, 4] | 3,0672 | 5 |

6.12)

| método | intervalo | raiz | iter |
|--------------|-----------|--------|------|
| secante | [1, 2] | 1,2167 | 5 |
| regula falsi | [1, 2] | 1,2167 | 19 |
| pégaso | [1, 2] | 1,2167 | 5 |

6.13)

| método | intervalo | raiz | iter |
|--------------|-----------|---------|------|
| secante | [-2, 0] | 3,5270 | 16 |
| regula falsi | [-2, 0] | -1,6813 | 8 |
| pégaso | [-2, 0] | -1,6813 | 6 |

Seção 6.4

6.16)

| método | intervalo | raiz | iter |
|-----------|-----------|---------|------|
| Muller | [-2, 0] | -1,3133 | 4 |
| W-D-Brent | [-2, 0] | -1,3133 | 8 |

6.17)

| método | intervalo | raiz | iter |
|-----------|-----------|--------|------|
| Muller | [0, 1] | 0,6329 | 3 |
| W-D-Brent | [0, 1] | 0,6329 | 6 |

6.18)

| método | intervalo | raiz | iter |
|-----------|-----------|--------|------|
| Muller | [0, 2] | 0,9180 | 3 |
| W-D-Brent | [0, 2] | 0,9180 | 6 |

Seção 6.5

6.21)

| método | x_0 | raiz | iter |
|-------------|-------|--------|------|
| Newton | 2 | 1,2823 | 5 |
| Schröder(1) | 2 | 1,2823 | 5 |

6.22)

| método | x_0 | raiz | iter |
|-------------|-------|--------|------|
| Newton | 2 | 1,0000 | 28 |
| Schröder(2) | 2 | 1,0000 | 12 |
| Schröder(3) | 2 | 1,0000 | 4 |

6.23)

| método | x_0 | raiz | iter |
|-------------|-------|--------|------|
| Newton | 3 | 2,0000 | 18 |
| Schröder(2) | 3 | 2,0000 | 5 |

Seção 6.6

6.26)

| método | raiz | iter | erro |
|--------------|--------|------|------|
| bisseção | 0,9454 | 35 | 0 |
| secante | 0,9454 | 8 | 0 |
| regula falsi | 0,9454 | 38 | 0 |
| pégaso | 0,9454 | 7 | 0 |
| Muller | 0,9454 | 4 | 0 |
| W-D-Brent | 0,9454 | 8 | 0 |
| Newton | 0,9454 | 4 | 0 |
| Schröder(1) | 0,9454 | 4 | 0 |

6.27)

| método | raiz | iter | erro |
|--------------|--------|------|------|
| bisseção | 1,8798 | 36 | 0 |
| secante | 1,8798 | 8 | 0 |
| regula falsi | 1,8798 | 41 | 0 |
| pégaso | 1,8798 | 7 | 0 |
| Muller | 1,8798 | 5 | 0 |
| W-D-Brent | 1,8798 | 8 | 0 |
| Newton | 1,8798 | 8 | 0 |
| Schröder(1) | 1,8798 | 8 | 0 |

6.28)

| método | raiz | iter | erro |
|--------------|-------------|------|------|
| bisseção | 1,5708 | 34 | 0 |
| secante | 4,7124 | 33 | 1 |
| regula falsi | 1,5708 | 9 | 0 |
| pégaso | 1,5708 | 9 | 0 |
| Muller | ($a+bi$)* | 500 | 1 |
| W-D-Brent | 1,5708 | 12 | 0 |
| Newton | 1,5708 | 8 | 0 |
| Schröder(1) | 1,5708 | 8 | 0 |

* $a = 7,8540$ e $b = -8 \times 10^{-4}$.

6.29)

| método | raiz | iter | erro |
|--------------|--------|------|------|
| bisseção | 3,0000 | 34 | 0 |
| secante | 3,0000 | 136 | 0 |
| regula falsi | 2,5877 | 500 | 1 |
| pégaso | 3,0000 | 188 | 0 |
| Muller | 3,0109 | 500 | 1 |
| W-D-Brent | 3,0000 | 80 | 0 |
| Newton | 3,0000 | 95 | 0 |
| Schröder(5) | 3,0000 | 4 | 0 |

6.30)

| método | raiz | iter | erro |
|--------------|---------|------|------|
| bisseção | -2,0000 | 34 | 0 |
| secante | -2,0000 | 72 | 0 |
| regula falsi | -2,0396 | 500 | 1 |
| pégaso | -2,0000 | 97 | 0 |
| Muller | -2,0000 | 500 | 1 |
| W-D-Brent | -2,0000 | 93 | 0 |
| Newton | -2,0000 | 54 | 0 |
| Schröder(3) | -2,0000 | 4 | 0 |

Gerais6.34) $D(\lambda) = \lambda^3 - 10\lambda^2 + 15\lambda + 49$, com $\lambda_1 = -1,5120$; $\lambda_2 = 4,9045$; $\lambda_3 = 6,6076$.6.35) $D(\lambda) = \lambda^3 - 17\lambda^2 + 75\lambda - 91$, com $\lambda_1 = 2,0543$; $\lambda_2 = 4,0748$; $\lambda_3 = 10,8709$.6.36) $L_3(x) = \frac{1}{2}(5x^3 - 3x)$, com $\lambda_1 = -0,77460$; $\lambda_2 = 0$; $\lambda_3 = 0,77460$.6.38) $V = 0,9984$ litro \times mol $^{-1}$.

6.39) pH = 6,82.

6.40) taxa = 5,75 %.

Capítulo 7: Equações diferenciais ordinárias**Seção 7.1**7.1) $y_5 = 1,55490$.7.2) $y_8 = 3,39195$.7.3) $y_{10} = 1,54711$.**Seção 7.3**7.12) $y_{20} = 16,39798$.7.13) $y_{100} = -595,31949$.7.14) $y_{100} = -1,81219$.7.15) $y_{200} = 17,92472$.**Seção 7.4**7.16) DP e ABM: $y_{100} = -0,19136$.7.17) DP e ABM: $y_{100} = -16,21194$.7.18) DP e ABM: $y_{100} = 2,30098$.7.19) DP e ABM: $y_{100} = -1,33333$.7.20) DP e ABM: $y_{100} = 27,25271$.

Seção 7.5

7.22) $y_{1,100} = 0,30956$ e $y_{2,100} = 0,19877$;
 solução exata: $y_1(t) = e^{-t} \sin(t)$,
 $y_2(t) = e^{-t} \cos(t)$.

7.23) $y_{1,100} = -7,00810$ e $y_{2,100} = 1,35212$;
 solução exata: $y_1(t) = e^t \cos(t\sqrt{3})$,
 $y_2(t) = -\frac{\sqrt{3}}{3}e^t \sin(t\sqrt{3})$.

7.24) $y_{100} = 0,26342$; solução exata:
 $y(x) = \cos(x)\left(\frac{11}{50} - \frac{x}{10}\right) + \sin(x)\left(\frac{1}{25} + \frac{3x}{10}\right) - \frac{4e^{2x}}{75} + \frac{5e^{-x}}{6}$.

7.25) $y_{100} = -1,35942$; solução exata:
 $y(x) = -\frac{7e^{-2x+2}}{12} - \frac{5e^{x-1}}{3} + \frac{x^2+x}{2} + \frac{1}{4}$.

Gerais

7.30) Solução exata: $y_1(x) = 3 - 2e^{-x}$,
 $y_2(x) = 3 - 2e^x$, $y_3(x) = 2(e^x + e^{-x}) - 3$.

Referências Bibliográficas

- [1] M. Abramowitz e I. A. Stegun. *Handbook of Mathematical Functions*. Dover, Nova Iorque, 1972.
- [2] F. S. Acton. *Numerical Methods that Work*. Harper and Row Pub., Nova Iorque, 1970.
- [3] P. Albrecht. *Análise Numérica, um curso moderno*. Livros Técnicos e Científicos Editora, Rio de Janeiro, 1973.
- [4] L. C. Barroso, M. M. A. Barroso, F. F. Campos, filho, M. L. B. de Carvalho, e M. L. Maia. *Cálculo Numérico*. Editora HARBRA, São Paulo, 2^a edição, 1987.
- [5] S. Bocanegra, L. C. de C. Medeiros, e F. F. Campos. Algoritmos de Newton-Cotes e Gauss-Legendre para integração dupla. RT 001/2000, DCC.ICEEx.UFMG, 2000.
- [6] R. P. Brent. *Algorithms for Minimization without Derivatives*. Prentice-Hall, Englewood Cliffs, 1973.
- [7] F. F. Campos e J. S. Rollett. The exponents method for calculating equilibrium concentrations of complex species in solution. *J. Comp. Chem.*, 16(5):534–544, 1995.
- [8] T. S. Chihara. *An Introduction to Orthogonal Polynomials*. Gordon and Breach, Nova Iorque, 1978.
- [9] L. Collatz. *Functional Analysis and Numerical Mathematics*. Academic Press, Nova Iorque, 1964.
- [10] B. P. Demidovich e I. A. Maron. *Computational Mathematics*. Mir Pub., Moscou, 1976.
- [11] W. S. Dorn e D. D. McCracken. *Cálculo Numérico com estudos de casos em Fortran IV*. Editora Campus-EDUSP, São Paulo, 1981.
- [12] M. Dowell e P. Jarratt. The “pegasus” method for computing the root of an equation. *BIT*, 12:503–508, 1972.
- [13] N. R. Draper e H. Smith. *Applied Regression Analysis*. John Wiley & Sons, Nova Iorque, 3^a edição, 1998.