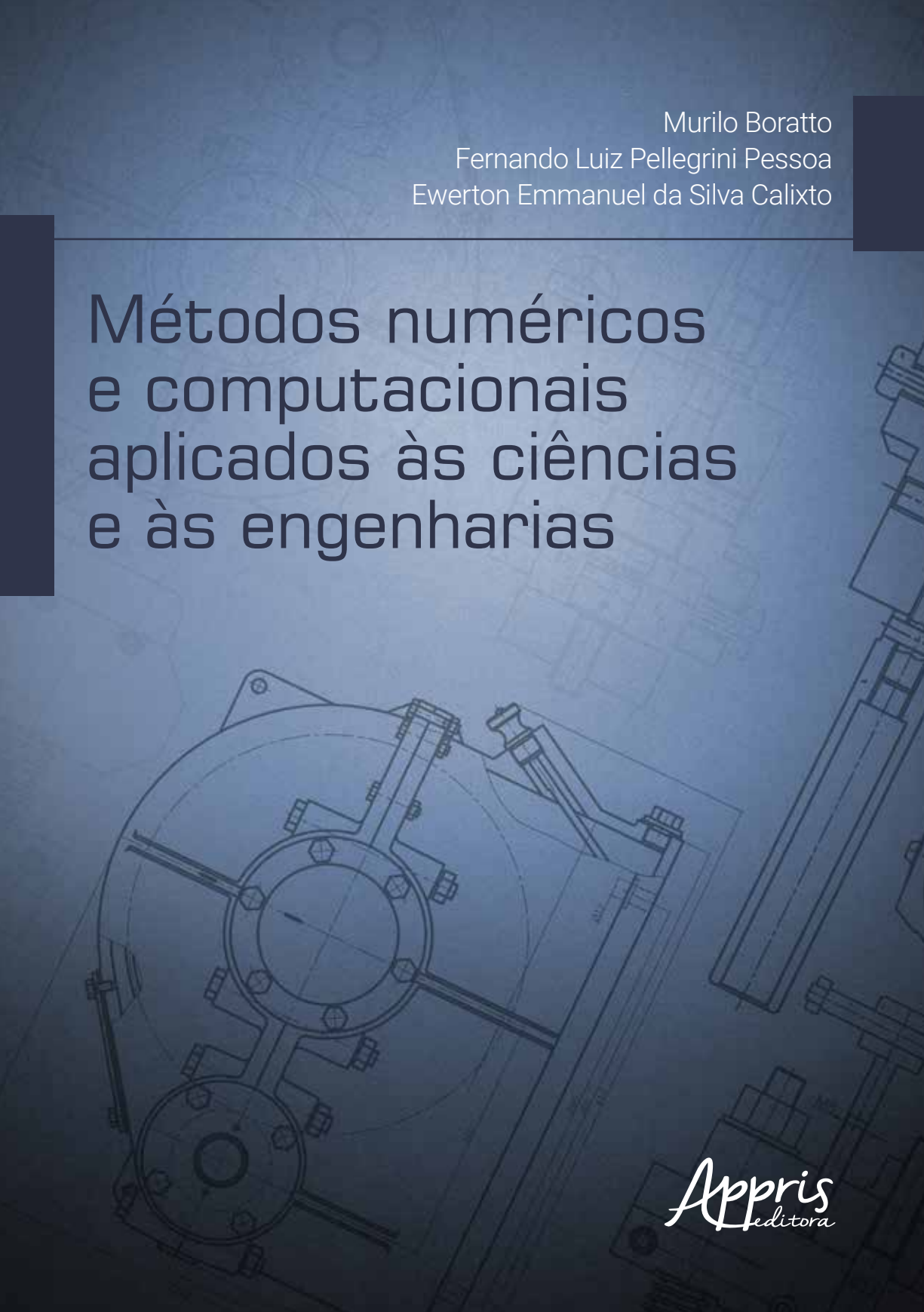
Murilo Boratto
Fernando Luiz Pellegrini Pessoa
Ewerton Emmanuel da Silva Calixto

Métodos numéricos e computacionais aplicados às ciências e às engenharias



Appris editora

MÉTODOS NUMÉRICOS E COMPUTACIONAIS APLICADOS ÀS CIÊNCIAS E ÀS ENGENHARIAS

Editora Appris Ltda.
1.ª Edição - Copyright© 2021 dos autores
Direitos de Edição Reservados à Editora Appris Ltda.

Catalogação na Fonte
Elaborado por: Josefina A. S. Guedes
Bibliotecária CRB 9/870

B726m
2021

Boratto, Murilo do Carmo
Métodos numéricos e computacionais aplicados às ciências e às engenharias / Murilo do Carmo Boratto, Fernando Luiz Pellegrini Pessoa, Ewerton Emmanuel da Silva Calixto. - 1. ed. - Curitiba : Appris, 2021.
108 p. ; 23 cm. – (Ensino de ciências).

Inclui bibliografia.
ISBN 978-65-250-0609-3

1. Matemática da computação. 2. Matemática aplicada. I. Pessoa, Fernando Luiz Pellegrini. II. Calixto, Ewerton Emmanuel da Silva. III. Título. IV. Série.

CDD – 519

Livro de acordo com a normalização técnica da ABNT

Editora e Livraria Appris Ltda.
Av. Manoel Ribas, 2265 – Mercês
Curitiba/PR – CEP: 80810-002
Tel. (41) 3156 - 4731
www.editoraappris.com.br

Printed in Brazil
Impresso no Brasil

Murilo Boratto
Fernando Luiz Pellegrini Pessoa
Ewerton Emmanuel da Silva Calixto

MÉTODOS NUMÉRICOS E COMPUTACIONAIS APLICADOS ÀS CIÊNCIAS E ÀS ENGENHARIAS

Appris editora

FICHA TÉCNICA

EDITORIAL	Augusto V. de A. Coelho Marli Caetano Sara C. de Andrade Coelho
COMITÊ EDITORIAL	Andréa Barbosa Gouveia - UFPR Edmeire C. Pereira - UFPR Iraneide da Silva - UFC Jacques de Lima Ferreira - UP Marilda Aparecida Behrens - PUCPR
ASSESSORIA EDITORIAL	Evelin Kolb
REVISÃO	Monalisa Morais Gobetti
PRODUÇÃO EDITORIAL	Lucielli Trevizan
DIAGRAMAÇÃO	Luciano Popadiuk
CAPA	Sheila Alves
COMUNICAÇÃO	Carlos Eduardo Pereira Débora Nazário Karla Pipolo Olegário
LIVRARIAS E EVENTOS	Estevão Misael
GERÊNCIA DE FINANÇAS	Selma Maria Fernandes do Valle

Às universidades públicas e privadas de qualidade...

AGRADECIMENTOS

Ao Centro de Supercomputação para Inovação Industrial SENAI CIMATEC, pelo interesse em apoiar a publicação desta primeira edição. Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), pelo suporte acadêmico. À Editora Appris, pelo interesse demonstrado em publicar este trabalho. Aos alunos, técnicos, pesquisadores e docentes da Universidade do Estado da Bahia (UNEB). Aos irmãos Spinola, Paulo e Taio, pelas figuras do livro. E, a todos aqueles que contribuíram direta ou indiretamente para a realização deste trabalho, os nossos sinceros agradecimentos.

PREFÁCIO

A partir do texto que tenho a honra de prologar, elimina-se de forma radical os argumentos que para alguns poderiam justificar críticas a uma visão excessivamente teórica dos professores de métodos numéricos. A mudança de ênfase surge de um grupo de professores de modelagem computacional que, preocupados com a qualidade do ensino, decidiram aproveitar as mudanças nas formas de aprendizagem nos cursos de Engenharia para revisar as ementas de métodos numéricos e apoiar a docência das disciplinas com este texto, sem esquecer os conceitos, dedicando um número de exercícios interligando a teoria matemática à prática cotidiana nos setores industriais.

Compartilho com os professores de métodos numéricos a visão de que a prática nas engenharias exigirá dos novos profissionais a capacidade de avançar no conhecimento, apontando soluções atuais aos novos problemas, esquecendo as antigas receitas que já não são capazes de abarcar a nova realidade.

Nessa linha, quero felicitar os autores deste excelente texto com toda a esperança de que o trabalho que hoje se apresenta anime outros docentes a desenvolver ferramentas que ajudem os discentes e pesquisadores envolvidos com métodos numéricos.

Domingo Giménez

Catedrático do Departamento de Informática e Sistemas
Universidad de Murcia

APRESENTAÇÃO

Ministrando disciplinas relacionadas a métodos numéricos ao longo de 10 anos, há pouco tempo algumas perguntas começaram a permear os meus pensamentos, por exemplo: "por que escrever um livro de métodos numéricos?". Consequentemente vieram outras perguntas como: "por que escrever um livro de métodos numéricos sendo que já existem tantos?". "Como seria possível escrever um livro de métodos numéricos diferente?". E talvez a resposta estivesse no desejo de transmitir as experiências pessoais, na espera de que elas fossem úteis para alguém. Como as disciplinas relacionadas a métodos numéricos, em sua grande maioria, esquecem-se do esforço de vinculação dos conceitos teóricos à aplicação prática nas ciências aplicadas, este texto tem o orgulho de desmistificar uma visão "demasiadamente teórica" em detrimento da aplicabilidade desses conceitos.

Este livro-guia mescla minhas experiências nas disciplinas ministradas ao longo da minha docência universitária e das minhas práticas discentes experimentadas nos cursos de graduação e pós-graduação realizados durante a minha carreira acadêmica. Tentei apresentar um texto claro e objetivo, evitando sempre que possível a notação matemática carregada, na tentativa de generalizar o público alvo. Múltiplos exemplos de algoritmos foram incluídos nos capítulos a fim de permitir uma programação rápida dos métodos propostos. Compartilho com todos os estudantes, profissionais e pesquisadores das ciências exatas as diversas visões práticas dos métodos numéricos. Parabenizo toda a equipe da Editora Appris pela oportunidade e pelo suporte, em especial os revisores. Simplesmente meu muito obrigado a todos que se encorajam a desenvolver ferramentas que ajudam a aprimorar habilidades e conhecimentos necessários para desenvolver suas tarefas numéricas com desenvoltura.

Murilo Boratto

Professor do Departamento de Ciências Exatas e da Terra I
Universidade do Estado da Bahia

SUMÁRIO

INTRODUÇÃO

Na Engenharia, na maioria das vezes, deparamo-nos com problemas que não possuem solução analítica. São problemas não lineares de grande complexidade, como por exemplo, problemas ligados à Termodinâmica, à Resistência de Materiais, à Cinética Química e outros. Na formulação desses problemas são necessárias equações não lineares, polinômios de graus elevados, integrais e equações diferenciais que muitas vezes não são de simples resolução. Para que os objetivos sejam alcançados, necessita-se de técnicas numéricas, as quais devem ser utilizadas para uma resolução plauzível.

O objetivo dos métodos numéricos é a resolução dos problemas que não podem ser resolvidos de forma analítica ou aparecem de forma inconveniente para uma interpretação direta. Estes podem ser utilizados para a aproximação de um problema complexo por outro mais receptível, para a obtenção da solução aproximada de um problema ou combinação dos dois.

Os métodos numéricos são um conjunto de ferramentas excelentes para buscar a razão de ser, fundamentar, e fazer inteligível o conhecimento científico. Essa é a ambição deste livro. Apresentamos uma série de problemas aplicados às ciências e às engenharias, dando aos métodos numéricos a importância que consideramos essencial.

Amigo leitor, serás tu quem terá o papel de julgar até que ponto conseguimos alcançar os nossos objetivos na produção deste livro. Estamos abertos a sugestões com a finalidade de sanar defeitos e poder enfocar com maior claridade e perspectiva futuras publicações.

Os autores

Capítulo 1

CONCEITOS BÁSICOS RELACIONADOS AOS MÉTODOS NUMÉRICOS

Neste primeiro capítulo, vamos apresentar alguns conceitos importantes para uma melhor compreensão dos conteúdos subsequentes. Conceitos como: convergência, algoritmo, tolerância, erros, séries entre outros, são extremamente úteis para o entendimento dos métodos numéricos e computacionais.

1.1 Definições Básicas

1.1.1 Algoritmo

A resolução de problemas utilizando métodos numéricos deve ser feita por meio de passos lógicos sequenciais (ou paralelos) e implementados por intermédio de uma estrutura computacional. A este conjunto de passos utilizados para a resolução de problemas denominamos **algoritmo**. Um exemplo simples é mostrado a seguir:

```
1.   function ALGORITMO(num, div) return i
2.   repeat
3.   for i ← 1, num do
4.     div ←  0
5.     for j ←  2, i − 1 do
6.       if num % div == 0 then div ←  div + 1
7.     end for
8.   end for
9.   until div == 0
10.  end function
```

Algoritmo 1. Exemplo de uma estrutura algorítmica.

Uma vez desenvolvido o algoritmo para a resolução de um problema, basta que este seja seguido tal como uma sequência de passos. Um outro fator importante é a tradução desse algoritmo para uma linguagem de programação (ex.: Fortran, C/C++, Python etc) e/ou a inserção de pacotes denominados API (*Application Programming Interface*) como MATLAB, MAPLE, BLAS, LAPACK, etc. O objetivo dessa portabilidade é a automatização e generalização da solução para diversos tipos de problemas.

1.1.2 Tolerância

É um número pequeno e predeterminado que é estabelecido antes da resolução do problema, que serve como uma espécie de limitador da precisão numérica e critério de parada. A tolerância[1] depende do tipo de problema a ser solucionado e da precisão numérica requerida, sendo adotado um critério para que seja alcançada uma solução.

1.1.3 Convergência

A solução de uma equação (ou um conjunto de equações) é dita numérica quando obtida por meio da manipulação de operações numéricas. A manipulação numérica utiliza-se de um método numérico para a resolução das equações. Nesse método está incluído um teste para verificar se a solução desejada é a correta. Essa solução é conhecida como a solução numérica do problema e é obtida por intermédio de aproximações sucessivas, abrindo-se mão de uma equação de recorrência, sendo que a solução aproximada é diferente da solução real.

Quando se atinge a solução numérica, diz-se que o método atingiu a convergência, isto é, que o critério de convergência foi satisfeito. Vários critérios de convergência são utilizados, podendo ser citado o erro absoluto de duas aproximações, que podem ser comparados com o critério numérico.

[1] A tolerância é simbolizada pela letra grega ε.

Suponhamos que:

$x \rightarrow$ *solução real*,

$x' \rightarrow$ *solução aproximada*,

$\varepsilon \rightarrow$ *tolerância.*

Um critério de convergência comumente utilizado é:

$$|x - x'| < \varepsilon \tag{1.1}$$

Isto é, o valor absoluto da diferença entre o valor real e o aproximado tem que ser menor que a tolerância escolhida para o método. Suponhamos que tenhamos $f(x)$:

$$|f(x)| < \varepsilon \tag{1.2}$$

Usando o método numérico, obtém-se a solução aproximada para o critério de convergência estabelecido.

1.1.4 Iterações

Refere-se ao número de aproximações realizadas até que se alcance a solução desejada, isto é, o número de repetições do método numérico é o que denominamos **iteração**. Caso a solução desejada não seja alcançada, significa que a convergência do método não foi atingida. Por isso, o número de iterações deve ser limitado de acordo com o propósito do problema. Para um problema de características divergentes, se não forem limitadas as iterações, será contabilizado um número infinito de etapas.

1.2 Tipos de Erros

O cálculo do erro é de extrema importância, pois nele está apoiada a confiabilidade de uma medida e de um método numérico. Os erros não são inerentes apenas ao ser humano, existem erros associados aos sistemas computacionais, pois estes são uma valiosíssima ferramenta na resolução de problemas. Temos outros tipos

de erros, como por exemplo, **erro relativo**, **erro percentual**, **erro de truncamento** e **arredondamento** que serão listados a seguir.

1.2.1 Erro Absoluto

O erro absoluto (E_A) é a diferença entre o valor exato de um número x e o seu valor aproximado x'. Matematicamente, o erro é expresso por:

$$E_A = |x - x'| \tag{1.3}$$

1.2.2 Erro Relativo

O erro relativo (E_R) é o erro absoluto dividido pelo valor aproximado. A expressão para o erro relativo é:

$$E_R = \frac{E_A}{x} \tag{1.4}$$

1.2.3 Erro Percentual

O erro percentual $(E_P)é$ o erro relativo em um formato porcentual, que nos dá uma ideia melhor quanto à precisão de um valor obtido, pois elimina erros quanto à ordem de grandeza.

$$E_P = E_R \cdot 100 \tag{1.5}$$

1.2.4 Erro de Arredondamento

É o erro relacionado à redução do número de dígitos de um valor. Esse tipo de erro é extremamente prejudicial no desenvolvimento de uma série de cálculos, ao longo da qual vamos fazendo arredondamentos, o que pode fornecer um valor bastante diferente no final dos cálculos.

1.2.5 Erro de Truncamento

Esse tipo de erro aparece quando se substitui uma sequência infinita de etapas (resultado exato) por uma sequência finita de etapas.

1.3 Propagação de Erro

Os erros aparecem em todos os campos dos métodos numéricos. Eles aparecem devido a erros provenientes de etapas anteriores, e acabam se propagando. Considerando que os valores de suas grandezas sejam representados por:

$$a' \pm \varepsilon_a \text{ e } b' \pm \varepsilon_b \tag{1.6}$$

em que a' e b' são os valores aproximados e ε_a e ε_b são os erros associados a estes números, e que a e b são os valores exatos, isto é:

$$a = a' \pm \varepsilon_a \text{ e } b = b' \pm \varepsilon_b \tag{1.7}$$

1.3.1 Adição

Para a adição de a com b, tem-se:

$$a + b = \left(a' \pm \varepsilon_a\right) + \left(b' \pm \varepsilon_b\right) \tag{1.8}$$

logo,

$$a + b = \left(a' + b'\right) \pm \left(\varepsilon_a + \varepsilon_b\right) \tag{1.9}$$

ou seja,

$$\varepsilon_{adição} = \varepsilon_a + \varepsilon_b \tag{1.10}$$

1.3.2 Subtração

Para a subtração entre os números a e b, tem-se:

$$a - b = (a' \pm \varepsilon_a) - (b' \pm \varepsilon_b) \tag{1.11}$$

logo,

$$a - b = (a' - b') \pm (\varepsilon_a + \varepsilon_b) \tag{1.12}$$

ou seja,

$$\varepsilon_{subtração} = \varepsilon_a + \varepsilon_b \tag{1.13}$$

1.3.3 Multiplicação

Para a multiplicação entre os números a e b, tem-se:

$$a \cdot b = (a' \pm \varepsilon_a) \cdot (b' \pm \varepsilon_b) \tag{1.14}$$

logo,

$$a \cdot b = (a' \cdot b') \pm (a' \cdot \varepsilon_b + b' \cdot \varepsilon_a + \varepsilon_a \cdot \varepsilon_b) \tag{1.15}$$

Considerando que $\varepsilon_a e \varepsilon_b$ são valores bem menores que a' e b' então $\varepsilon_a \cdot \varepsilon_b \to 0$, obtém-se:

$$a \cdot b = (a' \cdot b') \pm (a' \cdot \varepsilon_b + b' \cdot \varepsilon_a) \tag{1.16}$$

ou seja,

$$\varepsilon_{multiplicação} = (a' \cdot \varepsilon_b + b' \cdot \varepsilon_a) \tag{1.17}$$

1.3.4 Divisão

Para a divisão entre os números a e b, tem-se:

$$\frac{a}{b}=\frac{a'\pm\varepsilon_a}{b'\pm\varepsilon_a} \qquad \text{ou} \qquad \frac{a}{b}=(a'\pm\varepsilon_a)\cdot\frac{1}{b'\pm\varepsilon_b} \tag{1.18}$$

Analisando $\frac{1}{b'\pm\varepsilon_b}$ a partir da igualdade:

$$\frac{1}{1+x}\approx 1-x-x^2-\ldots \tag{1.19}$$

E tomando $\frac{1}{b'\pm\varepsilon_b}=\frac{1}{b'}\cdot\frac{1}{\left(1\pm\frac{\varepsilon_b}{b'}\right)}$, tem-se que:

$$\frac{1}{b'\pm\varepsilon_b}=\frac{1}{b'}-\frac{\varepsilon_b}{b'^2}-\frac{{\varepsilon_b}^2}{b'^3}-\frac{{\varepsilon_b}^3}{b'^4}-\ldots \tag{1.20}$$

E desprezando os termos superiores e 1, pois ε é um número pequeno, logo:

$$\frac{a}{b}=\left(a'\pm\varepsilon_a\right)\cdot\left(\frac{1}{b'}-\frac{\varepsilon_b}{b'^2}\right)=\left(a'\pm\varepsilon_a\right)\cdot\frac{1}{b'}\cdot\left(1\pm\frac{\varepsilon_b}{b'}\right) \tag{1.21}$$

$$\frac{a}{b}=\left(\frac{a'}{b'}\pm\frac{\varepsilon_a}{b'}\right)\cdot\left(1\pm\frac{\varepsilon_b}{b'}\right)=\frac{a'}{b'}\pm\frac{a'\cdot\varepsilon_b}{b'^2}\pm\frac{\varepsilon_a}{b'}\pm\frac{\varepsilon_a\cdot\varepsilon_b}{b'^2} \tag{1.22}$$

como $\varepsilon_a\cdot\varepsilon_b\to 0$, obtém-se:

$$\frac{a}{b}=\frac{a'}{b'}\pm\left(\frac{\varepsilon_a}{b'}+\frac{a'\cdot\varepsilon_b}{b'^2}\right) \text{ como } \varepsilon_{divisão}=\frac{\varepsilon_a}{b'}+\frac{a'\cdot\varepsilon_b}{b'^2} \tag{1.23}$$

1.4 Séries Numéricas

Algumas funções matemáticas podem ser escritas por meio de um polinômio com características especiais. Dizemos que essas funções, $f(x)$, podem ser escritas como séries de potência. As séries

de interesse no presente caso são as séries de Mac Laurin e Taylor, que são brevemente descritas a seguir.

1.4.1 Mac Laurin

Escrevendo-se de uma forma compacta, a série de **Mac Laurin** é representada por:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^n}{n!} x^n \tag{1.24}$$

Nesse caso são calculados todos os valores no ponto $x = 0$. Na série de Mac Laurin, existem inconvenientes para funções onde há descontinuidade, sendo que para esses casos utilizamos uma generalização, a série de **Taylor**.

1.4.2 Taylor

Na série de **Taylor** são conhecidas as funções $f(x)$ e suas derivadas em um dado ponto $x = x_0$. Expandindo-se $f(x)$ em série de potências infinitas em torno de x_0 em $(x - x_0)$, obtém-se:

$$f(x) = a_0 + a_1(x - x_0) + a_2(x - x_0)^2 + \ldots + a_n(x - x_0)^n + \ldots \tag{1.25}$$

Diferenciando *n* vezes, chega-se à:

$$f^n(x) = a_n n! + (n+1)!(x - x_0) + \ldots \tag{1.26}$$

Fazendo $x = x_0$:

$$f^n(x_0) = a_n n! \tag{1.27}$$

Logo, a série de Taylor é dada por:

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + \frac{f''(x_0)(x - x_0)^2}{2} + \ldots + \frac{f''(x_0)(x - x_0)^n}{n!} \tag{1.28}$$

Onde a forma compacta é:

$$f(x) = \sum_{n=0}^{\infty} \frac{f(x_0)(x - x_0)^n}{n!} \tag{1.29}$$

1.5 Exercícios Propostos

Exercício 1.5.1

Sendo $a = 3 \pm 0,5$ e $b = 2 \pm 0,1$, calcular $a + b$, $a - b$, $a \times b$, a / b.

Exercício 1.5.2

Escreva os **erros absoluto** e **relativo** da soma dos números: $6423 + 2,3230 \times 10^{-4} + 1,2340$, considerando quatro dígitos significativos.

Capítulo 2

RESOLUÇÃO NUMÉRICA DE SISTEMAS DE EQUAÇÕES LINEARES

A resolução de sistemas de equações lineares é um dos principais assuntos abordados no estudo dos métodos numéricos. Várias aplicações de diversas áreas das ciências e engenharias podem ser modeladas e analisadas, tratando-as sob o ponto de vista desses tipos de sistemas. Neste capítulo assumiremos que o leitor tenha uma certa familiaridade com os conceitos básicos da álgebra linear, de tal forma que trabalharemos com tipificações, representações e operações matriciais em nossos exemplos. Catalogaremos os métodos de resolução em dois grupos: métodos diretos, que são aqueles que conduzem à solução exata em um número finito de passos e os métodos iterativos, os quais fundamentam-se na construção de sequências de aproximações e cujos valores calculados anteriormente são usados para melhorar a aproximação da solução.

2.1 Métodos Diretos

Como definido anteriormente, os métodos diretos consistem em transformar um sistema de equações lineares em outro equivalente, cuja a resolução seja praticamente imediata. Essa transformação faz-se por meio de chamadas elementares em um determinado número de passos. Apresentaremos neste capítulo alguns métodos diretos para essa resolução, os quais destacamos:

- Resolução de Sistema de Equações Tridiagonais;
- Gauss-Jordan;
- Fatoração LU.

A escolha do método a ser utilizado dependerá de cada caso apresentado, de tal forma que o método tenha a vantagem de fornecer a solução sem a dependência de condições de convergência, apesar de que este pode se mostrar inviável quando o sistema é muito grande ou mal condicionado. Denotamos que um sistema de equações lineares com *n* linhas e *n* colunas pode ser expresso pela seguinte forma algébrica:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases} \tag{2.1}$$

De modo análogo a sua representação para um formato matricial genérico do tipo $Ax = h$ é expresso por:

$$\begin{pmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & & a_{nn} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix} \tag{2.2}$$

Logo, poderemos representar o sistema na forma matricial, solucionando-o em um número finito de passos, a partir da definição dos vetores colunas $x = (x_1, x_2, \ldots, x_n)^t$ e $b = (b_1, b_2, \ldots, b_n)^t$. Para isso, consideremos que A é uma matriz inversível, sendo que o sistema possui solução para $x = A^{-1}b$, onde A^{-1} representa a inversa da matriz, de tal forma que tenha solução pelos métodos diretos.

2.1.1 Resolução de Sistema de Equações Triangulares

A resolução de sistemas de equações triangulares tem como característica básica uma resolução substitutiva e retroativa. Essa resolução é marcada pela forma estrutural do sistema que, para os sistemas triangulares, possui a presença de elementos localizados abaixo ou acima da diagonal principal, facilitando os cálculos. A seguir, apresentamos a forma geral para um sistema triangular superior:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ \quad a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \quad\quad \ddots \quad\quad \vdots \quad \vdots \\ \quad\quad\quad\quad\quad a_{nn}x_n = b_n \end{cases} \tag{2.3}$$

O qual podemos calcular as incógnitas $x_1, x_2, \ldots, x_n$ fazendo as seguintes operações:

$$\begin{cases} x_n = \dfrac{b_n}{a_{nn}} \\ x_{n-1} = \dfrac{b_{n-1} - a_{n-1}x_n}{a_{n-1,n-1}} \\ \vdots \\ x_1 = \dfrac{b_1 - a_{1n}x_n - \ldots - a_{13}x_3 - a_{12}x_2}{a_{11}} \end{cases} \tag{2.4}$$

Dessa maneira estabelecemos o algoritmo para a resolução de sistemas de equações triangulares superiores:

```
1.   function SUPERIOR(a, b, n) return x
2.   repeat
3.   soma ← b_k
4.   for k ← n − 1, 1 do
5.     soma ← b_k
6.     for j ← k + 1, n do
7.       soma ← soma − a_kj x_j
8.     end for
9.   end for
10.  x_k ← soma / a_kk
11.  until x_n ← b_n / a_nn, soma ← 0
12.  end function
```

Algoritmo 2. Resolução de um sistema triangular superior.

O exemplo mostrado a seguir consiste na resolução de um sistema de equações triangulares superiores:

$$\begin{cases} 2x_1 - x_2 + x_3 = 2 \\ \quad x_2 + 2x_3 = 3 \\ \quad\quad\quad x_3 = 1 \end{cases} \tag{2.5}$$

O sistema pode ser facilmente solucionado por substituição de variáveis como mostrado no algoritmo anterior, tendo a solução final $x = (1, 1, 1)^t$. Como o sistema anterior caracteriza-se por sua simplicidade de resolução de um sistema triangular, sendo a matriz A triangular superior, podemos generaliza-la para os sistemas triangulares inferiores.

2.1.2 Gauss-Jordan

O **método de Gauss-Jordan**, é um dos métodos diretos mais conhecidos para a resolução de sistemas de equações lineares. A ideia consiste em transformar o sistema a ser resolvido em um sistema de equações triangulares. A partir de operações elementares, ocorre a redução a um sistema equivalente, isto é, um sistema triangular, cuja resolução é mais simples, sendo obtido o resultado por meio de substituições regressivas. O fundamento desse método consiste em efetuar essas operações sobre as equações do sistema, com a finalidade de obter um sistema diagonal equivalente. Por definição, um sistema diagonal equivalente é aquele em que os elementos a_{ij} da matriz de coeficiente A são iguais a zero, para $i \neq j$. A seguir, um exemplo ilustrativo:

$$A = \left(\begin{array}{ccc|c} 1 & 0 & 0 & a_{11} \\ 0 & 1 & 0 & a_{22} \\ 0 & 0 & 1 & a_{33} \end{array}\right) \tag{2.6}$$

A seguir, exemplifica-se os passos para a resolução de um problema concreto com dimensões pequenas para um problema em concreto proposto.

Exemplo: Resolva o sistema de equações lineares mostrado a seguir utilizando o **método de Gauss-Jordan**:

$$\begin{cases} x_2 + 2x_3 - x_4 = 1 \\ x_1 + x_3 + x_4 = 4 \\ -x_1 + x_2 - x_4 = 2 \\ 2x_2 + 3x_3 - x_4 = 7 \end{cases} \tag{2.7}$$

Passo 1: Construir a forma aumentada da matriz.

O primeiro passo consiste em transformar o sistema de equações lineares em um sistema equivalente, isto é, colocá-lo na forma aumentada. Logo, o sistema pode ser escrito na forma aumentada mesclando a matriz A e o vetor b:

$$A = \begin{pmatrix} 0 & 1 & 2 & -1 \\ 1 & 0 & 1 & 1 \\ -1 & 1 & 0 & -1 \\ 0 & 2 & 3 & -1 \end{pmatrix} \qquad b = \begin{pmatrix} 1 \\ 4 \\ 2 \\ 7 \end{pmatrix} \tag{2.8}$$

Transcritos na notação, M aumentada da matriz,

$$M = \left(\begin{array}{cccc|c} 0 & 1 & 2 & -1 & 1 \\ 1 & 0 & 1 & 1 & 4 \\ -1 & 1 & 0 & -1 & 2 \\ 0 & 2 & 3 & -1 & 7 \end{array}\right) \tag{2.9}$$

Passo 2: Redução da matriz M a um formato *escalonado reduzido por linhas.*

O objetivo desse passo é trazer a matriz aumentada M a um formato *escalonado reduzido por linhas*. Sendo assim, a matriz nesse formato:

- Tem o primeiro elemento, conhecido como *pivot*, com o valor 1,
- cada *pivot* é a única entrada diferente de zero, na sua coluna,
- cada linha tem pelo menos tantos zeros à esquerda como a fila anterior.

O processo de escalonamento é realizado a partir de 3 tipos de *operações elementares nas linhas*, as quais são permitidas:

- **Troca** da ordem das linhas;
- **Multiplicação** de uma das equações por um número real não-nulo;
- **Substituição** de uma das equações por uma combinação linear dela mesma com outra equação.

Aplicando o escalonamento na redução da matriz aumentada, temos:

$$\left(\begin{array}{cccc|c} 0 & 1 & 2 & -1 & 1 \\ 1 & 0 & 1 & 1 & 4 \\ -1 & 1 & 0 & -1 & 2 \\ 0 & 2 & 3 & -1 & 7 \end{array}\right) \xrightarrow[R_2 \leftrightarrow R_3]{R_1 \leftrightarrow R_2} \left(\begin{array}{cccc|c} 1 & 0 & 1 & 1 & 4 \\ -1 & 1 & 0 & -1 & 2 \\ 0 & 1 & 2 & -1 & 1 \\ 0 & 2 & 3 & -1 & 7 \end{array}\right) \xrightarrow{R_1 + R_2 \to R_3} \left(\begin{array}{cccc|c} 1 & 0 & 1 & 1 & 4 \\ 0 & 1 & 1 & 0 & 6 \\ 0 & 1 & 2 & -1 & 1 \\ 0 & 2 & 3 & -1 & 7 \end{array}\right) \xrightarrow[R_4 - 2R_2 \to R_4]{R_3 - R_2 \to R_3} \left(\begin{array}{cccc|c} 1 & 0 & 1 & 1 & 4 \\ 0 & 1 & 0 & 0 & 6 \\ 0 & 0 & 1 & -1 & -5 \\ 0 & 0 & 1 & -1 & -5 \end{array}\right) \xrightarrow{R_4 - R_3 \to R_4} \left(\begin{array}{cccc|c} 1 & 0 & 1 & 1 & 4 \\ 0 & 1 & 1 & 0 & 6 \\ 0 & 1 & 1 & -1 & -5 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right) \xrightarrow[R_2 - R_3 \to R_2]{R_1 - R_3 \to R_1} \left(\begin{array}{cccc|c} 1 & 0 & 1 & 2 & 9 \\ 0 & 1 & 1 & 1 & 11 \\ 0 & 0 & 1 & -1 & -5 \\ 0 & 0 & 1 & 0 & 0 \end{array}\right) \tag{2.10}$$

Passo 3: Obtendo a solução

- Se o bloco de esquerda da matriz de redução de linha não é quadrado, torná-lo quadrado consiste em adicionar ou remover linhas de zeros. Isto terá de ser feito de tal forma que os principais 1 em cada fila (o elemento *pivot*) encontrem-se na diagonal principal.
- A coluna mais à direita da matriz aumentada de redução de linha é uma solução particular.
- Para encontrar uma base para a solução geral do sistema, procede-se da seguinte forma: toma-se cada coluna da linha reduzida da matriz aumentada que tem um zero na diagonal. O conjunto desses vetores colunas é a base que você precisa.

No exemplo anterior, uma solução particular é $x = (9, 11, -5, 0)^t$ e a solução geral da equação é um subespaço com base no vetor qualquer. Portanto, a solução geral do sistema $Ax = b$ é:

$$A = \begin{pmatrix} 9 \\ 11 \\ -5 \\ 0 \end{pmatrix} + \lambda(\alpha) \tag{2.11}$$

2.1.3 Fatoração LU

Outro método bastante interessante é a decomposição, ou simplesmente **Fatoração LU**. A base do método apoia-se na simplicidade da resolução de sistemas triangulares. Supondo que é possível fatorar a matriz A em um produto de uma matriz triangular inferior L (com os elementos da diagonal principal iguais a 1), e uma matriz triangular superior U, isto é:

$$A = LU \tag{2.12}$$

Nessa condição, o sistema $Ax = b$ pode ser escrito na forma $LUx = b$, o que permite o desmembramento em dois sistemas triangulares:

$$\begin{aligned} Ly &= b \\ Ux &= y \end{aligned} \tag{2.13}$$

Podem-se encontrar os valores de L e de U a partir da definição de produtos de matrizes, montando um sistema com n^2 equações e n^2 incógnitas, que será progressivamente incrementado a partir de valores anteriormente calculados. Dessa maneira, se chamarmos l_{ij}, os elementos de L, e de u_{ij} os elementos de U, obteremos expressões que viabilizam uma implementação eficiente do método, tal que:

$$\begin{cases} u_{ij} = a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj} \quad para \quad i \leq j \\ l_{ij} = \dfrac{a_{ij} - \sum_{k=1}^{i-1} l_{ik} u_{kj}}{u_{jj}} \quad para \quad i > j \end{cases} \tag{2.14}$$

Sendo esta expressão usada no algoritmo a seguir:

```
1.   function LU(a_ij ← A) return L, U
2.   for i ← 1, n do
3.     for j ← i, n do
4.       u_ij ← a_ij − sum(k ← 1, i − 1) l_ik u_kj
5.     end for
6.     for j ← i + 1, n do
7.       l_ji ← (a_ji − sum(k ← 1, i − 1) l_jk u_ki) / u_ii
8.     end for
9.   end for
10.  end function
```

Algoritmo 3. Fatoração LU.

Exemplo: Calcule o sistema de equações lineares utilizando a **Fatoração LU** para o seguinte sistema:

$$\begin{cases} x_1 + 2x_2 - x_3 = 2 \\ 2x_1 + 3x_2 - 2x_3 = 3 \\ x_1 - 2x_2 + x_3 = 0 \end{cases} \tag{2.15}$$

Colocamos o sistema de equações no formato matricial $Ax = b$:

$$\begin{pmatrix} 1 & 2 & -1 \\ 2 & 3 & -2 \\ 1 & -2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ 0 \end{pmatrix} \tag{2.16}$$

A partir da matriz A calculando l_{ij} e u_{ij}, pelas fórmulas apresentadas nesta seção, teremos:

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 4 & 1 \end{pmatrix} \leftrightarrow U = \begin{pmatrix} 1 & 2 & -1 \\ 0 & -1 & 0 \\ 0 & 0 & 2 \end{pmatrix} \tag{2.17}$$

Para resolvermos o sistema $Ax = b$, resolvemos, $Ly = b$:

$$\begin{pmatrix} 1 & 0 & -1 \\ 2 & 1 & -2 \\ 1 & 4 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 3 \\ 0 \end{pmatrix} \overset{Ly=b}{\rightarrow} y = (2, -1, 2)^t \tag{2.18}$$

e com estes valores, calculamos x a partir de $Ux = y$:

$$\begin{pmatrix} 1 & 2 & -1 \\ 0 & -1 & 0 \\ 0 & 0 & 2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ -1 \\ 2 \end{pmatrix} \overset{Ux=y}{\rightarrow} x = (1, 1, 1)^t \tag{2.19}$$

No exemplo anterior, a solução geral da equação é um subespaço com base unitária. Portanto, a solução geral do sistema $Ax = b$ é $x = (1, 1, 1)^t$. Isto ocorrerá sempre que o vetor solução b for formado pela soma dos elementos das linhas correspondentes:

$$\sum_{j=1}^{n} a_{ij} = b_i \tag{2.20}$$

2.2 Métodos Iterativos

Os métodos iterativos estão associados aos conceitos de iteração e caracterizam-se por envolver os seguintes elementos:

1. **[Aproximação Inicial] —** Consiste na primeira aproximação para a resolução numérica do problema.
2. **[Critério de Parada] —** É o instrumento por meio do qual o procedimento iterativo é finalizado.
3. **[Sequência Iterativa] —** Como sendo uma transformação do conjunto dos inteiros positivos no conjunto dos reais. O número real associado a k é designado por x_k .

Esse processo iterativo gera uma sucessão de aproximações x_k , cada uma com um erro associado. Logo, um método é definido por uma equação iterativa, com a qual se constrói aproximações à solução do problema. A aplicação numérica da equação iterativa obriga o conhecimento de uma aproximação inicial e à definição de um conjunto de condições que garantam que a aproximação calculada, em uma certa iteração, encontra-se suficientemente próxima à solução final, levando em conta o fator de tolerância. Quando estas condições forem verificadas, obtém-se o critério de parada para este valor. Se no sistema $Ax = b$ os elementos da diagonal principal são diferentes de zero, $a_{ij} \neq 0$, $i = 1 : n$, então, podemos explicitar x_1 usando a primeira

equação, x_2 usando a segunda equação e assim sucessivamente. Ou seja, reescrevemos o sistema:

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1 \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n = b_2 \\ \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \ldots + a_{nn}x_n = b_n \end{cases} \tag{2.21}$$

em um formato que seja conveniente para o método iterativo:

$$x_1 = \frac{\left(b_1 - \sum_{j=2}^{n} a_{1j}x_j\right)}{a_{11}} \ldots x_i = \frac{\left(b_i - \sum_{j\neq i}^{n} a_{ij}x_j\right)}{a_{ii}} \ldots x_n = \\ = \frac{\left(b_n - \sum_{j=1}^{n-1} a_{nj}x_j\right)}{a_{nn}} \tag{2.22}$$

Nesta seção apresentaremos os seguintes métodos iterativos:

- Jacobi;
- Gauss-Seidel.

2.2.1 Jacobi

Esse método iterativo é utilizado para calcular uma sequência $x^{(1)}$, $x^{(2)}$, ..., $x^{(n)}$. Sendo que a partir de uma aproximação inicial conhecida, que chamaremos de $x^{(0)}$, usaremos o primeiro termo à direita da primeira equação para definir uma generalização para uma nova aproximação para x_i:

$$x_i^{(1)} = \frac{\left(b_i - \sum_{\substack{j=2 \\ j\neq i}}^{n} a_{ij}x_j^{(0)}\right)}{a_{ii}}, i = 1:n \tag{2.23}$$

Em resumo, o **método de Jacobi** consiste em calcularmos as aproximações $x^{(1)}$, $x^{(2)}$, ..., $x^{(n)}$ usando a generalização anterior.

```
1.   function JACOBI(A, b, n, x^(0), max, tol) return x
2.   for k ← 0, max do
3.     for i ← 1, n do
4.       x_i^(k+1) ← (b_i − sum(j ← 1, j != i, n) a_ij
     x_j^(k)) / a_ii
5.     end for
6.     if |x_i^(k+1) − x_i^(k)|_inf / |x_i^(k+1)|_inf < tol then
7.       x ← x^(k+1)
8.     end if
9.   end for
10.  end function
```

Algoritmo 4. Método de Jacobi.

A aplicação do método terá êxito se a sequência de vetores construída sucessivamente convergir para a solução do sistema. Isto é, o valor calculado se aproxima de zero quando k cresce. Essa condição estabelece o critério de parada a seguir, baseado no cálculo da discrepância da norma em relação ao infinito, o qual definimos como a tolerância (ε):

$$\frac{x_i^{(k+1)} - x_i^{(k)}{}_{\infty}}{x_i^{(k+1)}{}_{\infty}} < \varepsilon \tag{2.24}$$

Exemplo:

Seja o sistema:

$$\begin{cases} 10x_1 + 2x_2 + x_3 = 14 \\ x_1 + 5x_2 + x_3 = 11 \\ 2x_1 + 3x_2 + 10x_3 = 8 \end{cases} \tag{2.25}$$

Calcule a resolução do sistema de equações lineares utilizando o **método de Jacobi**, tomando como condição inicial $x^{(0)} = (0, 0, 0)^t$, para uma tolerância de 10^{-2} e 4 algarismos significativos. Sendo assim as iterações são calculadas por:

$$\begin{aligned} x_1^{(k+1)} &= \frac{14}{10} - \frac{2}{10}x_2^{(k)} - \frac{1}{10}x_3^{(k)} \\ x_2^{(k+1)} &= \frac{11}{5} - \frac{1}{5}x_1^{(k)} - \frac{1}{5}x_3^{(k)} \end{aligned} \tag{2.26}$$

$$x_3^{(k+1)} = \frac{8}{10} - \frac{2}{10} x_1^{(k)} - \frac{3}{10} x_2^{(k)}$$

Na tabela a seguir, apresentamos os resultados das 6 primeiras iterações do **método de Jacobi**. Na quinta coluna, apresentamos o critério de parada, o qual definimos como a tolerância.

i	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	ε
0	0,0000	0,0000	0,0000	–
1	1,4000	2,2000	0,8000	1,0000
2	0,8800	1,7600	– 0,1400	0,5341
3	1,0340	2,0520	0,0960	0,1423
4	0,9800	1,9739	– 0,0225	0,0600
5	1,0075	2,0085	0,0118	0,0172
6	0,9971	1,9961	– 0,0041	0,0062

Tabela 2.1. Tabela com a solução do método de Jacobi.

Para as condições lançadas anteriormente temos que a solução do sistema por meio do **método de Jacobi** é:

$$x = \left(0,9971,\ 1,9961,\ -0,0041\right)^t \tag{2.27}$$

Comparando o valor anterior calculado para as condições estipuladas com o valor real, temos:

$$x = \left(1,\ 2,\ 0\right)^t \tag{2.28}$$

Notamos uma certa discrepância entre os valores calculados e a solução real do sistema. Os reais motivos para isso ocorrer são basicamente dois:

- O valor numérico alto da tolerância;
- O erro numérico de truncamento para o baixo número de casas decimais.

A partir dessas conclusões parciais inferimos que para diminuirmos essa discrepância e obtermos um valor mais próximo do real, basta revertermos os dois pontos citados. Apesar de aparentemente ser a forma mais simples, nem sempre é a mais comum e possível, devido à consequência direta de incremento do esforço computacional associado. Isto é, um número maior de cômputo de instruções estará diretamente ligado ao seu tempo de execução, sendo muitas vezes impraticável a sua implementação sobre uma arquitetura computacional.

2.2.2 Gauss-Siedel

O **método de Gauss-Siedel** pode ser olhado como uma otimização do **método de Jacobi**. Nesse método utilizaremos uma técnica de otimização que consiste em aproveitar os cálculos já atualizados de outros componentes, para atualizar os componentes que estão sendo calculados. Assim, o valor recém calculado para $x_1^{(k+1)}$ será usado no cálculo de $x_2^{(k+1)}$ e assim sucessivamente. Dessa maneira, as iterações serão definidas pelo algoritmo a seguir:

```
1.   function GAUSS-SEIDEL(A, b, n, x(0), max,
     tol) return x
2.   for k ← 0, max do
3.     for i ← 1, n do
4.     x_i(k+1) ← (b_i − sum(j ← 1,i−1) a_ij x_j(k+1) − sum(j ← i+1,n)
     a_ij x_j(k)) / a_ii
5.     end for
6.     if |x_i(k+1) − x_i(k)|_inf / |x_i(k+1)|_inf < tol then
7.       x ← x(k+1)
8.     end if
9.   end for
10.  end function
```

Algoritmo 5. Método de Gauss-Seidel.

Para o mesmo exemplo anterior, utilizando o **método de Gauss-Siedel**, temos que as iterações, partindo de $x^{(0)} = (0,0,0)^t$, são calculados por:

$$x_1^{(k+1)} = \frac{14}{10} - \frac{2}{10} x_2^{(k)} - \frac{1}{10} x_3^{(k)}$$

$$x_2^{(k+1)} = \frac{11}{5} - \frac{1}{5}x_1^{(k+1)} - \frac{1}{5}x_3^{(k)} \qquad (2.29)$$

$$x_3^{(k+1)} = \frac{8}{10} - \frac{2}{10}x_1^{(k+1)} - \frac{3}{10}x_2^{(k+1)}$$

Na tabela a seguir, apresentamos os resultados das 4 primeiras iterações do **método Gauss-Siedel**. Na quinta coluna, apresentamos o critério de parada, o qual definimos como a tolerância.

i	$x^{(1)}$	$x^{(2)}$	$x^{(3)}$	ε
0	0,0000	0,0000	0,0000	–
1	1,4000	1,9200	– 0,0560	1,0000
2	1,0216	2,0069	– 0,0064	0,1885
3	0,9993	2,0014	– 0,0003	0,0111
4	0,9998	2,0001	0,0001	0,0007

Tabela 2.2. Tabela com a solução do método de Gauss-Siedel.

Para as condições lançadas anteriormente temos que a solução do sistema por meio do **método de Gauss-Siedel,** o sistema converge na 4ª iteração com os valores:

$$x = \left(0,9998,\ 2,0001,\ 0,0001\right)^t \qquad (2.29)$$

Comparando o valor anterior calculado para as condições estipuladas com o valor real, temos:

$$x = \left(1,\ 2,\ 0\right)^t \qquad (2.30)$$

Notamos uma menor discrepância entre os valores calculados e o real. Logo, concluímos que com a atualização imediata dos dados, viu-se que a convergência foi mais rápida. Embora isso ocorra em quase a totalidade dos casos, não podemos generalizar, pois há casos em que o **método de Gauss-Siedel** diverge e há convergência para o **método de Jacobi** ou vice-versa.

2.2.3 Critérios de Convergência

O processo iterativo nos leva a concluir que a convergência para a solução exata não é uma garantia para qualquer sistema apresentado. Logo, existem certas condições que devem ser satisfeitas por um sistema de equações lineares para se garantir a existência de convergência. Dentre os muitos critérios, utilizaremos uma técnica que possui uma condição suficiente e necessária, chamada **critério de convergência de Sassenfeld**. Esse critério consiste em satisfazer a seguinte regra:

$$\max\left(\beta_i\right) < 1 \tag{2.31}$$

de tal forma que β_i corresponde à:

$$\beta_i = \left|\frac{1}{a_{ii}}\right| \sum_{j=1}^{i-1} \left|a_{ii}\right| \beta_j + \sum_{j=i+1}^{n} \left|a_{ij}\right|, i = 2,3,\ldots,n \tag{2.32}$$

Exemplo: O sistema de equações lineares a seguir converge para o **método de Gauss-Siedel**:

$$\begin{cases} 2x_1 + x_2 - 0,2x_3 + 0,2x_4 = 1 \\ 0,6x_1 + 3x_2 - 0,6x_3 - 0,3x_4 = 4 \\ -0,1x_1 - 0,2x_2 + x_3 + 0,2x_4 = 2 \\ 0,4x_1 + 1,2x_2 + 0,8x_3 + 4x_4 = 2 \end{cases} \tag{2.33}$$

Utilizando o **critério de convergência de Sassenfeld**, temos os cálculos de β_i, a partir da matriz de adjacência:

$$\begin{cases} 2x_1 + x_2 - 0,2x_3 + 0,2x_4 = 1 \\ 0,6x_1 + 3x_2 - 0.6x_3 - 0,3x_4 = 4 \\ -0,1x_1 - 0,2x_2 + x_3 + 0,2x_4 = 2 \\ 0,4x_1 + 1,2x_2 + 0,8x_3 + 4x_4 = 2 \end{cases} \tag{2.34}$$

$$\beta_1 = \left|\frac{1}{a_{11}}\right| \cdot \left(\left|a_{12}\right| + \left|a_{13}\right| + \left|a_{14}\right|\right) = 0,7000$$

$$\beta_2 = \left|\frac{1}{a_{22}}\right| \cdot \left(\left|a_{21}\right| \beta_1 + \left|a_{23}\right| + \left|a_{24}\right|\right) = 0,4400$$

$$\beta_3 = \left|\frac{1}{a_{33}}\right| \cdot \left(|a_{31}|\beta_1 + |a_{32}|\beta_2 + |a_{34}|\right) = 0,3580$$

$$\beta_4 = \left|\frac{1}{a_{44}}\right| \cdot \left(|a_{41}|\beta_1 + |a_{42}|\beta_2 + |a_{43}|\beta_3\right) = 0,2736$$

Como $M = \max(\beta_i) = 0,7$ sendo menor que a unidade, implica que a solução desse sistema irá convergir usando o **método de Gauss-Seidel**.

2.3 Exercícios Propostos

2.3.1 Exercícios Gerais

Exercício 2.3.1.1

Verificar as condições de convergência do **método de Gauss-Seidel** no sistema a seguir (Considere $x^{(0)} = (0, 0, 0)^t$, tolerância de 10^{-4} e 4 algarismos significativos):

$$\begin{cases} 2x_1 + x_2 + 3x_3 = 3 \\ -x_2 + x_3 = 1 \\ x_1 + 3x_3 = 3 \end{cases}$$

Qual a sua conclusão a respeito da verificação de convergência do referido método?

Exercício 2.3.1.2

Obter a solução com 4 algarismos significativos do sistema linear

$$\begin{cases} 4x_1 + 2x_2 + x_3 = 11 \\ -x_1 + 2x_2 = 3 \\ 2x_1 + x_2 + 4x_3 = 16 \end{cases}$$

usando os **métodos de Jacobi**, **Gauss-Seidel** e **Fatoração LU**. Compare a velocidade de convergência nos dois métodos iterativos com o número de etapas do método direto. (Considere $x^{(0)} = (0, 0, 0)^t$,

tolerância de 10^{-2} e 4 algarismos significativos para os métodos iterativos).

$$L = \begin{pmatrix} 1 & 0 & 0 \\ -0,25 & 1 & 0 \\ 0,5 & 0 & 1 \end{pmatrix} \leftrightarrow U = \begin{pmatrix} 4 & 2 & 1 \\ 0 & 2,5 & 0,25 \\ 0 & 0 & 3,5 \end{pmatrix}$$

Exercício 2.3.1.3

Calcular a solução do sistema de equações lineares utilizando a **Fatoração LU**, considerando 4 algarismos significativos:

$$\begin{cases} 3x_1 + 2x_2 + 4x_3 = 1 \\ x_1 + x_2 + 2x_3 = 2 \\ 4x_1 + 3x_2 - 2x_3 = 3 \end{cases}$$

Sabe-se que:

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 0,33 & 1 & 0 \\ 1,33 & 1 & 1 \end{pmatrix} \leftrightarrow U = \begin{pmatrix} 3 & 2 & 4 \\ 0 & 0,33 & 0,66 \\ 0 & 0 & -8 \end{pmatrix}$$

2.3.2 Exercícios Aplicados

Exercício 2.3.2.1 - Aplicado à Engenharia de Produção (Logística de Produção)

Uma indústria produz 3 tipos de produtos: (1), (2) e (3), os quais são processados e produzidos no decorrer da semana. Para a produção de cada unidade desses produtos, necessita-se de 3 diferentes tipos de matérias primas (A), (B) e (C). A indústria possui em estoque 19, 17 e 22 unidades de A, B e C respectivamente, conforme a tabela a seguir:

	A	B	C
1	8	4	6
2	2	9	7
3	9	4	9
Estoque	19	17	22

Pede-se:

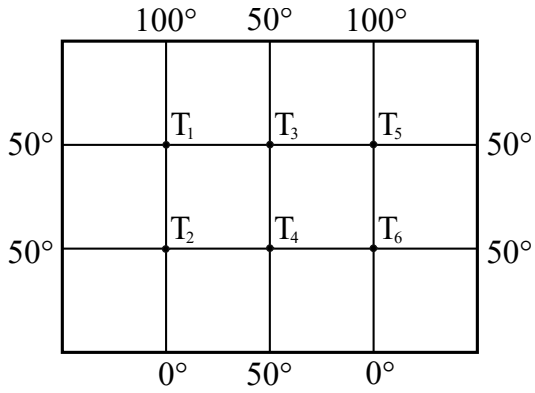
a) Resolva o sistema de equações lineares utilizando a **Fatoração LU**. Sabe-se que a *LU* da matriz de adjacência anterior é:

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 1/2 & 1 & 0 \\ 3/4 & 0 & 1 \end{pmatrix} \leftrightarrow U = \begin{pmatrix} 8 & 2 & 9 \\ 0 & 8 & -1/2 \\ 0 & 0 & 83/32 \end{pmatrix}$$

b) Utilizando o **critério de convergência de Sassenfeld** no sistema anterior, pode-se garantir a convergência para a solução exata utilizando o **método de Gauss-Seidel**? Caso a convergência seja garantida, resolva o sistema de equações lineares dado utilizando o **método de Gauss-Seidel** (Considere $x^{(0)} = (0, 0, 0)^t$, tolerância de 10^{-2} e 4 algarismos significativos).

Exercício 2.3.2.2 - Aplicado à Engenharia Química (Transferência de Calor)

O diagrama a seguir representa a discretização do problema do calor em uma placa. O problema trata de determinar a temperatura nos nós interiores da placa T_j para $j = 1,2,\ldots,6$, conhecidas as temperaturas das bordas e supondo que há equilíbrio térmico, isto é, as temperaturas não variam. Logo, supõem-se que, no equilíbrio, a temperatura em cada nó é a média das temperaturas nos nós vizinhos.



Pede-se:

a) O sistema de equações lineares correspondente aos dados da figura anterior, formulando as condições que verifiquem as temperaturas dos nós vizinhos.

b) Resolva o sistema escolhendo um método direto para a sua resolução.

c) Resolva o sistema de equações lineares utilizando o **método de Gauss-Seidel** e compare com o **método de Jacobi** (Considere $x^{(0)} = (0, 0, 0, 0, 0)^t$, tolerância de 10^{-2} e 4 algarismos significativos).

Exercício 2.3.2.3 - Aplicado à Física (Discretização do Laplaciano)

Uma maneira de se obter a solução da Discretização do Laplaciano em uma região retangular consiste em se fazer uma discretização que transforma a equação em um problema aproximado, consistindo em uma equação de diferenças cuja solução, em um caso particular, exige a solução do seguinte sistema de equações lineares, dada sua representação matricial:

$$\begin{pmatrix} 5 & -1 & 2 & -1 \\ 1 & 9 & -3 & 4 \\ 0 & 3 & -7 & 2 \\ -2 & 2 & -3 & 10 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 5 \\ 26 \\ -7 \\ 33 \end{pmatrix}$$

a) Encontre os valores das incógnitas, a partir da decomposição da matriz de adjacência, utilizando a **Fatoração LU** e compare utilizando o **método de Gauss-Jordan**. Dadas as matrizes *L* e *U*:

$$L = \begin{pmatrix} 5 & 0 & 0 & 0 \\ 1 & 9,2000 & 0 & 0 \\ 0 & 3 & -5,8913 & 0 \\ -2 & 1,6000 & -1,6087 & 8,6974 \end{pmatrix} \leftrightarrow U = \begin{pmatrix} 1 & -0,2000 & 0,4000 & -0,2000 \\ 0 & 1 & -0,3696 & 0,4565 \\ 0 & 0 & 1 & -0,1070 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

b) Considerando o **critério de convergência de Sassenfeld** no sistema anterior, pode-se garantir a convergência para a solução exata, utilizando o **método de Gauss–Seidel**? Justifique sua resposta.

c) Caso a convergência seja garantida, resolva o sistema de equações lineares dado utilizando o **método de Gauss-Seidel** (Considere $x^{(0)} = (0, 0, 0, 0)^t$, tolerância de 10^{-2} e 4 algarismos significativos).

Capítulo 3

INTERPOLAÇÃO NUMÉRICA

Interpolação é o processo de estimar valores de uma certa função $f(x)$ para valores conhecidos. Suponhamos que temos pontos $x_0, x_1, x_2, \ldots, x_n$ e conhecidos valores da função $y_0 = f(x_0)$, $y_1 = f(x_1)$, $\ldots y_n = f(x_n)$, partiremos desses dados para encontrar uma aproximação, a qual consideramos que existe uma curva de ajuste que passe pelos pontos dados. Logo, a interpolação, é o processo de encontrar esta função $f(x)$, satisfazendo as condições $f(x_k) = f_k$ para $k = 0, \ldots, n$. Como na maioria das vezes as funções escolhidas para o ajuste são polinômios, apresentaremos neste capítulo duas formas de interpolação polinomial:

- Lagrange;
- Newton.

A explicação teórica do processo de interpolação consiste em ideias que são centrais para grande parte dos métodos numéricos. Na maioria das vezes um método computacional é útil apenas se for consistente e estável. Assim, todas as formas de interpolação discutidas aqui serão consistentes, mesmo sabendo que algumas exceções para a interpolação polinomial de alta ordem podem ser instáveis.

A interpolação tem muitos usos na computação científica, além do uso óbvio de estimar valores de funções específicas, muitos métodos de interpolação são empregados para estimar as derivadas de uma função por meio da resolução de equações diferenciais ordinárias.

3.1 Lagrange

Estamos interessados em estabelecer métodos numéricos que permitam encontrar um polinômio que passe por $n+1$ pontos conhecidos. Nessa classe de problemas, coloca-se a aproximação de funções que são conhecidas em pontos discretos e determina-se um

polinômio $P(x)$, tal que $P(x) = f(x_0),\ f(x_1), \ldots f(x_n)$. Esse problema é chamado de **interpolação de Lagrange,** e o polinômio $P(x)$ que satisfaz essas condições é chamado de **polinômio interpolador** de $f(x)$ nos pontos $x_0, x_1, x_2, \ldots, x_n$. Seja a função $f(x)$ conhecida nos $n+1$ pontos distintos de $x_0, x_1, x_2, \ldots, x_n$. Existe um único polinômio $P(x)$ de grau menor ou igual a n, tal que:

$$P(x_i) = \sum_{i=0}^{n} L_i(x) f(x_i), \forall i = 0 : n \tag{3.1}$$

Seja

$$P(x_i) = L_0(x) f(x_0) + L_1(x) f(x_1) + \ldots + L_n(x) f(x_n) \tag{3.2}$$

onde $L_i(x)$ são os polinômios de Lagrange definidos por:

$$L_i(x) = \prod_{\substack{i=0 \\ i \neq k}}^{n} \frac{(x - x_i)}{(x_k - x_i)} = \frac{(x - x_0)(x - x_1)\ldots(x - x_n)}{(x_k - x_0)(x_k - x_1)\ldots(x_k - x_n)} \tag{3.3}$$

A verificação de que $P(x)$ é o polinômio interpolador segue diretamente da substituição dos pontos, levando-se em conta que:

$$L_i(x_i) = 0,\ \forall i \neq k \tag{3.4}$$

Exemplo: Encontre um **polinômio interpolador de Lagrange** $P(x)$ que passa pelos seguintes pontos:

i	x	$f(x)$
0	0	-5
1	1	1
2	3	25

Como serão utilizados 3 pontos, o polinômio interpolador $P(x)$ tem grau 2. Pela substituição dos pontos de interpolação nas expressões de $L_i(x)$, calculamos os polinômios de Lagrange:

$$P_2(x) = \sum_{i=0}^{2} L_i(x) f(x_i),\ \forall i = 0 : 2 \tag{3.5}$$

$$L_0(x) = \frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} = \frac{(x-1)(x-3)}{(0-1)(0-3)} = \frac{x^2-4x+3}{3} \quad (3.6)$$

$$L_1(x) = \frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} = \frac{(x-0)(x-3)}{(1-0)(1-3)} = \frac{-x^2+3}{2} \quad (3.7)$$

$$L_2(x) = \frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} = \frac{(x-0)(x-1)}{(3-0)(3-1)} = \frac{x^2-x}{6} \quad (3.8)$$

Usando a expressão anterior, teremos o polinômio interpolador $P(x)$ de grau 2:

$$P_2(x) = \sum_{i=0}^{2} L_i(x) f(x_i) \quad (3.9)$$

$$P_2(x) = L_0(x) f(x_0) + L_1(x) f(x_1) + L_2(x) f(x_2) \quad (3.10)$$

$$P_2(x) = -5\left(\frac{x^2-4x+3}{3}\right) + \left(\frac{-x^2+3}{2}\right) + 25\left(\frac{x^2-x}{6}\right) \quad (3.11)$$

$$P_2(x) = 2x^2 + 4x - 5 \quad (3.12)$$

3.2 Newton

Outro método bastante utilizado é a interpolação de Newton. A base do método apoia-se na característica de recorrência dos pontos conhecidos. O **polinômio interpolador de Newton** de grau $n+1$ pode ser calculado usando o polinômio de grau n e um novo ponto. Pode-se escrever o polinômio da seguinte forma:

$$P(x) = c_0 + c_1(x_1 - x_0) + c_2(x - x_0) + \ldots + c_n(x - x_0)(x - x_1)\ldots(x - x_{n-1}) \quad (3.13)$$

onde as constantes $c_0, c_1, c_2, \ldots, c_n$ serão determinadas pelas condições de interpolação. Tomando $x = x_0$ anteriormente e usando a condição de interpolação $P(x_0) = f(x_0)$, obtemos:

$$c_0 = f(x_0) \quad (3.14)$$

No cálculo de c_1 definimos a diferença dividida de segunda ordem associada aos pontos x_0 e x,

$$f\left[x_0, x\right] = \frac{f(x) - f(x_0)}{x - x_0} \tag{3.15}$$

usamos $c(x_0) = f(x_0)$ e reorganizamos os termos para estabelecer

$$\begin{aligned} \frac{P_n(x) - f(x_0)}{x - x_0} &= c_1 + c_2(x - x_1) + \ldots + \\ &+ c_n(x - x_0)(x - x_1)\ldots(x - x_{n-1}) \end{aligned} \tag{3.16}$$

Fazendo $x = x_1$ nesta última equação, o lado direito é igual a c_1. Usando a condição de interpolação $P_0(x_1) = f(x_1)$, obtemos o valor de c_1:

$$c_1 = f\left[x_0, x_1\right] = \frac{f(x_1) - f(x_0)}{x_1 - x_0} \tag{3.17}$$

A ideia é repetir esse procedimento até encontrarmos todos os coeficientes $c_0, c_1, c_2, \ldots, c_n$. Para generalizar o procedimento, a relação de recorrência, definiremos o conceito de diferenças divididas.

Diferenças Divididas:

Se $f(x_i)$ e $f(x_j)$ são quaisquer dois valores de $f(x_j)$, não necessariamente consecutivos, então chamam-se diferenças divididas de 1ª ordem a relação:

$$f\left(x_i, x_j\right) = \frac{f(x_j) - f(x_i)}{x_j - x_i} \tag{3.18}$$

Generalizando, consideramos que $f[x_0] = f(x_0)$. Por recorrência definimos para qualquer k,

$$\begin{aligned} &f\left(x_0, x_j, \ldots, x_{k-1}, x_k, x\right) = \\ &= \frac{f\left[x_0, x_j, \ldots, x_{k-1}\right] - f\left[x_0, x_j, \ldots, x_{k-1}, x_k\right]}{x - x_k} \end{aligned} \tag{3.19}$$

Normalmente as diferenças divididas são apresentadas na forma tabular seguinte:

x	$f(x)$			
x_0	$f(x_0)$			
		$f(x_0, x_1)$		
x_1	$f(x_1)$		$f(x_0, x_1, x_2)$	
		$f(x_1, x_2)$		$f(x_0, x_1, x_2, x_3)$
x_2	$f(x_2)$		$f(x_1, x_2, x_3)$	
		$f(x_2, x_3)$		
x_3	$f(x_3)$			

Figura 3.2. Exemplo da estrutura de uma tabela das diferenças divididas.

Utilizando o **polinômio interpolador de Newton** por meio das diferenças progressivas, temos:

$$\begin{aligned} P(x) &= f(x_0) + (x - x_0) f(x_0, x_1) + \\ &+ (x - x_0)(x - x_1) f(x_0, x_1, x_2) + \\ &+ (x - x_0)(x - x_1)(x - x_2) f(x_0, x_1, x_2, x_3) \end{aligned} \tag{3.20}$$

Exemplo: Faça a aproximação de $P(1,05)$ usando os dados a seguir e do **polinômio interpolador de Newton**:

x	$f(x)$
0	3
1	0
3	18
4	63
6	285

O resultado das diferenças divididas fica assim:

x	$f(x)$				
$x_0 = 0$	3				
		-3			
$x_1 = 1$	0		4		
		9		2	
$x_2 = 3$	18		12		0
		45		2	
$x_3 = 4$	63		22		
		111			
$x_4 = 6$	285				

Figura 3.3. Representação das diferenças divididas para o exemplo anterior.

Logo, o polinômio interpolador na forma de Newton utilizando as diferenças progressivas fica,

$$P_4(x) = 3 - 3(x-0) + 4x(x-1) + 2x(x-1)(x-3) \qquad (3.21)$$

e substituíndo o valor $1,05$ no polinômio:

$$P_4(1,05) = -3,144775 \qquad (3.22)$$

3.3 Exercícios Propostos

3.3.1 Exercícios Gerais

Exercício 3.3.1.1

Dados os seguintes pontos: $x_0 = 2$, $x_1 = 2,5$ e $x_2 = 4$, determine o **polinômio interpolador de Lagrange** para a curva $f(x) = 1/x$.

Exercício 3.3.1.2

Encontre os **polinômios interpoladores de Lagrange** e **Newton** de grau 3 para os pontos: $(0,2)$, $(1,4)$, $(3,5)$ *e* $(4,0)$.

Exercício 3.3.1.3

Calcule o valor de $P(0,05)$ para um **polinômio interpolador de Newton** os seguintes pontos: $(-1,4),(0,1),(2,-1)$.

3.3.2 Exercícios Aplicados

Exercício 3.3.2.1 - Aplicado à Engenharia Civil (Resistência dos Materiais)

Uma deflexão de uma longa placa retangular carregada de forma uniforme e sobre tensão axial pode ser calculada pela seguinte equação:

$$\begin{cases} y''(x) - \dfrac{S}{D}y(x) = -\dfrac{9L}{2D} + \dfrac{9L}{2D}x^2,\ 0 \le x \le L \\ y(0) = y(L) = 0 \end{cases}$$

Placa Retangular		
Dado	**Símbolo**	**Valor**
Comprimento da Placa	L	$1m$
Rigidez	D	$10^7\ N \cdot m$
Força Axial	S	$200N \cdot cm$

a) Calcule a aproximação para a deflexão em cada ponto de partição utilizando os **polinômios interpoladores de Lagrange** e **Newton**.

b) Qual o valor da deflexão para o ponto $L = 0,5\ m$?

Capítulo 4

ZEROS DA FUNÇÃO

Um problema que ocorre com uma grande frequência nas áreas aplicadas é o calculo de raízes equações na forma: $f(x)=0$. Resolver esse tipo de equação consiste em torná-la igual a zero, isto é, consiste em determinar a solução (ou soluções) real ou complexa dessa função. A resolução dessas equações é uma atividade realizada desde a antiguidade. A história da matemática registra que desde os primórdios já se usava técnicas matemáticas com o objetivo de aproximações de raízes. As primeiras tentativas de resolução de equações foram resolvidas pelos povos gregos e árabes, antes de Cristo, por meio de métodos geométricos e aritméticos. No século XVI, os italianos resolveram, analiticamente, as equações cúbicas e quadráticas e deram início à modelagem dos primeiros métodos numéricos iterativos, os quais possuem concepção nos dias atuais. Sendo assim, os métodos numéricos iterativos são os mais usuais e eficientes para determinar aproximadamente a solução real do valor de x. Nesses métodos, para determinar uma solução quando esta é um valor real, necessitamos de uma solução inicial. A partir dessa solução, geramos uma sequência de soluções aproximadas que, sob determinadas condições teóricas, convergem para a solução desejada para um critério de parada determinado. Embora exista uma grande quantidade de métodos destinados a essa resolução, optamos por alguns métodos iterativos que apresentam bons resultados de convergência denominados:

- Bisseção;
- Newton-Raphson;
- Secante.

Neste capítulo abordaremos esses métodos iterativos e mostraremos as diferenças estruturais entre os algoritmos computacionais.

4.1 Teoria do Valor Fundamental

A teoria do valor fundamental é a base das operações dos métodos numéricos iterativos. O seu significado é expresso por meio do seguinte axioma:

Sendo $f(x)$ *contínua em um intervalo* $[a,b]$, *se* $f(a)\cdot f(b)<0$, *então existe pelo menos um ponto* $x=a$ *entre* a *e* b *que é zero de* $f(x)$.

4.2 Zeros da Função

A aplicação de métodos numéricos iterativos destinados ao refinamento de raízes caracteriza-se por uma série de instruções executáveis sequencialmente, algumas das quais repetidas em ciclos (iterações) resultam em valores aproximados de raízes das equações. A seguir, apresentamos alguns métodos numéricos para os zeros da função.

4.2.1 Bisseção

O **método da Bisseção** é um algoritmo de busca de raízes que funciona obtendo aproximações sucessivamente melhores das raízes de funções com valor real. O método é enunciado por meio do seguinte axioma:

Dada uma função $f(x)$ *contínua no intervalo* $[a,b]$ *onde existe uma raiz única, é possível determinar tal raiz subdividindo sucessivas vezes o intervalo que a contém pelo ponto médio de* a *e* b.

E seus passos podem ser resumidos por meio do seguinte algoritmo, da seguinte forma:

```
1.    function BISSECAO(f(x), a, b, max, tol) return x*
2.    repeat
3.    for k ← 0, max do
4.      x ← (a + b) / 2
5.      if f(a) · f(b) < 0 then b ← x
6.      else a ← x
```

```
7.    end for
8.    until |b - a| < tol
9.    x* ←  (a + b) / 2
10.   end function
```

Algoritmo 6. Método da Bisseção.

Apesar do método ser bastante simples e previsível, ele sofre um conjunto de desvantagens, como por exemplo: lentidão, baixa acurácia, alto custo computacional, embora forneça aproximações dentro do intervalo inicial.

4.2.2 Newton-Raphson

O **método de Newton-Raphson** é um algoritmo de busca de raízes que funciona obtendo aproximações sucessivamente melhores das raízes de funções com valor real. O método em uma variável pode ser enunciado por meio do seguinte axioma:

Dada uma função $f(x)$ *contínua no intervalo* $[a,b]$ *onde existe uma raiz única, é possível determinar uma aproximação de tal raiz a partir da interseção da tangente à curva em um ponto* x_0 *com o eixo das abscissas.*

O método começa com uma função $f(x)$ definida sobre os números reais x, tal que a derivada da função $f'(x)$ e um valor inicial x_0 para uma raiz da função $f(x)$. Se a função satisfizer as suposições feitas na derivação da fórmula e a suposição inicial estiver próxima, então apresenta uma melhor aproximação para x_1, sendo:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{4.35}$$

Geometricamente, $(x_1, 0)$ é a interseção com o eixo x da tangente ao gráfico de $f(x)$ em $(x_0, f(x_0))$. O processo apresentado é repetido como

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{4.2}$$

até que o critério de parada seja alcançado. De forma análoga à anterior, os passos podem ser resumidos por meio do seguinte algoritmo:

```
1.   function NEWTON_RAPHSON(x0, f(x), f'(x),
     tol) return x*
2.   for k ← 1, 2, ... do
3.     xk ← xk-1 − (f(xk-1) / f'(xk-1))
4.     if |f(xk)| < tol then x* ← xk
5.   end for
6.   end function
```

Algoritmo 7. Método de Newton-Raphson.

O **método de Newton-Raphson** é provalvelemente a técnica mais usada para o cálculo das raízes das equações algébricas, superando todos os outros pela rapidez e eficiência. A única ressalva do método seria para os casos em que a derivada analítica é de difícil obtenção, sendo utilizada para esses casos a derivação numérica, ou no caso de uma estimativa inicial estar muito afastada da solução.

4.2.3 Secante

O **método da Secante** diferencia-se dos métodos anteriores por não precisar estabelecer o intervalo de busca, isto é, realiza o cálculo de uma secante a partir de dois pontos pré-existentes. A fundamentação básica consiste em substituir a derivada numérica, de forma análoga ao método de Newton, só que calculando a secante a partir de dois pontos elegíveis, sendo o valor de *x* desconhecido. O método da Secante é enunciado mediante o seguinte axioma:

Dada uma função $f(x)$ *contínua no intervalo* $[a,b]$ *onde existe uma raiz única, é possível determinar uma aproximação de tal raiz a partir da interseção da secante à curva em dois ponto* x_0 *e* x_1 *com o eixo das abscissas.*

O axioma anterior explica que para $f(x)$, sendo uma função contínua em um intervalo fechado $[a,b]$ tal que $f(a)\cdot f(b)<0$, existe uma solução da equação $f(x)=0$, para $x\in[a,b]$, sendo garantida pelo Teorema do Valor Intermediário. Isto é, considera-se a linha que

conecta os valores do ponto final e a linha que liga esses dois pontos uma linha denominada secante e a mesma é dada pela seguinte fórmula:

$$x_{i+1} = \frac{x_{i-1} f(x_i) - x_i f(x_{i-1})}{f(x_i) - f(x_{i-1})} \tag{4.3}$$

O algoritmo a seguir resume o método:

```
1.   function SECANTE(x0, x1, f(x), tol) return x*
2.   for k ← 1, 2, ... do
3.     xk+1 ← ((xk-1 · f(xk))-(xk · f(xk-1)))/(f(xk)-f(xk-1))
4.     if |f(xk+1)| < tol then x* ←  xk+1
5.   end for
6.   end function
```

Algoritmo 8. Método da Secante.

Destacam-se as seguintes vantagens do **método da Secante**: Rapidez processo de convergência, cálculos mais convenientes que do método de Newton e um desempenho elevado.

4.3 Exercícios Propostos

4.3.1 Exercícios Gerais

Exercício 4.3.1.1

Utilizando o **método da Bisseção**, calcule o valor numérico do zero das funções para cada caso (considerando 4 algarismos significativos):

a) $f(x) = x\log(x) - 1 \rightarrow I = [2,3] \rightarrow \varepsilon = 0,002$

b) $f(x) = x^2 + ln(x) \rightarrow I = [0,1,1] \rightarrow \varepsilon = 0,001$

c) $f(x) = x^3 - x - 1 \rightarrow I = [1,2] \rightarrow \varepsilon = 0,02$

d) $f(x) = x^2 - 3 \rightarrow I = [1,2] \rightarrow \varepsilon = 0,01$

e) $f(x) = x^3 - 10 \rightarrow I = [2,3] \rightarrow \varepsilon = 0,1$

Exercício 4.3.1.2

Utilizando o **método de Newton-Raphson**, calcule o valor numérico do zero das funções para cada caso (considerando 4 algarismos significativos):

a) $f(x) = x^3 + 2x - 1 \rightarrow x_0 = 0,5 \rightarrow \varepsilon = 0,002$

b) $f(x) = xe^x - 1 \rightarrow x_0 = 1 \rightarrow \varepsilon = 0,001$

c) $f(x) = x^2\left(x^2 - 1\right) \rightarrow x_0 = 1,5 \rightarrow \varepsilon = 0,001$

d) $f(x) = 2^x - x^2 \rightarrow x_0 = -0,7 \rightarrow \varepsilon = 0,001$

Exercício 4.3.1.3

Utilizando o **método da Secante**, calcule o valor numérico da raiz (considerando 6 algarismos significativos) de cada função para cada caso, dados x_0 e x_1:

a) $f(x) = x^3 - 2x^2 - 5 \rightarrow x_0 = 1, x_1 = 2,5 \rightarrow \varepsilon = 0,001$

b) $f(x) = x - cos(x) \rightarrow x_0 = 0, x_1 = 1 \rightarrow \varepsilon = 0,001$

c) $f(x) = \sqrt{x} - 5e^{-x} \rightarrow x_0 = 1,4, x_1 = 1,5 \rightarrow \varepsilon = 0,001$

d) $f(x) = x^3 - \frac{1}{2} \rightarrow x_0 = 0, x_1 = 1 \rightarrow \varepsilon = 0,001$

4.3.2 Exercícios Aplicados

Exercício 4.3.2.1 - Aplicado à Engenharia Ambiental (Cálculo do Nível de Oxigênio em um Rio Poluído)

Em Engenharia Ambiental, a equação a seguir pode ser usada para calcular o nível de oxigênio em um rio poluído, após a chegada de uma descarga de esgoto.

$$N_{oxig} = 10 - e^x,$$

onde a variável N_{oxig} indica o nível de oxigênio em função de x, que é a distância rio abaixo em quilômetros (km). Determine em que distância, após a descarga de esgoto, o nível de oxigênio terá caído para 0 (utilize 6 algarismos significativos e uma tolerância de 0,0001).

Capítulo 5

INTEGRAÇÃO NUMÉRICA

O cálculo de integrais realizado por meio de formas numéricas pode evidenciar múltiplos formatos, por exemplo, funções integradoras representadas por uma tabela de valores, ou ainda por uma expressão analítica $f(x)$, sendo que pode ser, muitas vezes, complicada ou pode não resultar na obtenção da primitiva correspondente. Sendo assim, os métodos numéricos que calculam integrais surgem a partir da necessidade de contornar essa complexidade de cálculo, a fim de fornecer aproximações melhores. A integração numérica consiste na aproximação da função integradora a um polinômio interpolador. A eleição desse polinômio é uma das referências que determinará a utilização do método de integração. Uma vez estabelecido o método e a aproximação polinomial, tornam-se quase que imediatos à sua integração. Como as fórmulas de integração são somatórios cujas parcelas são valores da função $f(x)$ calculados em pontos e multiplicados por pesos convenientemente escolhidos, temos a seguinte generalização:

$$\int_a^b f(x)dx \approx \sum_{i=0}^{n} w_i f(x_i) \tag{5.1}$$

onde $a \leq x_0 < x_1 < \ldots < x_n \leq b$ são chamados de pontos e w_i são pesos da integração. Para cada método serão estabelecidos os pontos e os pesos correspondentes ao critério de aproximação adotado. Optamos por duas vertentes de métodos que apresentam bons resultados, denominados:

- Fórmulas de Newton-Cotes
  - Trapézios;
  - 1/3 de Simpson.
- Fórmulas Gaussianas
  - Gauss.

Nesse capítulo abordaremos estes métodos e mostraremos as diferenças estruturais entre os algoritmos computacionais.

5.1 Fórmulas de Newton-Cotes

As Fórmulas de Newton-Cotes são caracterizadas por pontos de integração igualmente espaçados no intervalo de integração (a,b). Tal que é expresso por meio do seguinte axioma:

Seja n um número inteiro e $h=(b-a)/n$ a amplitude intervalar, tal que, os pontos de integração são definidos por $x_j = a + j \cdot h$ em $0 \le j \le n$.

A seguir, analisaremos alguns detalhes de alguns métodos particulares de integração para as Fórmulas de Newton-Cotes, nas quais os pesos w_i, são calculados com diferentes valores de n.

5.1.1 Trapézios

O **método dos Trapézios** corresponde à interpolação da função $f(x)$ a ser integrada por um polinômio de grau n. Como a interpolação linear pede 2 pontos, usaremos os extremos do intervalo de integração, isto é, $a = x_0$ e $b = x_n$. Como a soma da área de n trapézios, cada qual definido pelo seu subintervalo, temos a seguinte generalização do método:

$$\int_a^b f(x)dx \approx \int_{x_1}^{x_0} f(x)dx + \ldots + \int_{x_{n-1}}^{x_n} f(x)dx \approx \tag{5.2}$$

$$\approx \frac{h}{2}\cdot\left[f(x_0)+f(x_1)\right]+\ldots+\frac{h}{2}\cdot\left[f(x_{n-1})+f(x_n)\right] \tag{5.3}$$

Como somente os termos $f(x_0)$ e $f(x_n)$ não se repetem, assim, esta fórmula pode ser simplificada da seguinte forma:

$$\int_a^b f(x)dx \approx \frac{h}{2}\cdot\left\{f(x_0)+2\left[f(x_1)+\ldots+f(x_{n-1})\right]+f(x_n)\right\} \tag{5.4}$$

Pode-se traduzir o **método dos Trapézios** no seguinte algoritmo:

```
1.   function TRAPEZIOS(f(x_i), h) return I_T
2.   for k ← 1, n − 1 do
3.     soma ← soma + f(x_i)
4.   end for
5.   I_T = (h/2) · [2 · soma + f(x_0) + f(x_n)]
6.   end function
```

Algoritmo 9. Método dos Trapézios.

Exemplo: Calcule o valor aproximado da integral a seguir utilizando o **método dos Trapézios**, comparando os valores para 1, 2 e 9 subintervalos.

$$\int_0^4 \frac{1}{\sqrt{1+x^2}}\,dx \tag{5.5}$$

O primeiro passo para a resolução da integração numérica é calcular os valores de x na função $f(x)=\dfrac{1}{\sqrt{1+x^2}}$:

x	$f(x)$
0	1,0000
0,5	0,8944
1,0	0,7071
1,5	0,5547
2,0	0,4472
2,5	0,3713
3,0	0,3162
3,5	0,2747
4,0	0,2425

A partir do cálculo anterior consideramos os valores para 1 subintervalo (x_0, x_1):

$$\int_0^4 \frac{1}{\sqrt{1+x^2}}\,dx = \frac{h}{2}\cdot\left(f(x_0)+f(x_1)\right) = 2\cdot\left(1,0000+0,2425\right) = 2,4850 \tag{5.6}$$

Para 2 subintervalos (x_0, x_1, x_2):

$$\int_0^4 \frac{1}{\sqrt{1+x^2}}\,dx = \frac{h}{2}\cdot\left(f(x_0)+2f(x_1)+f(x_2)\right) = \\ = 1\cdot(1,0000+2\cdot(0,4472)+0,2425) = 2,1369 \tag{5.7}$$

E para 9 subintervalos $(x_0, x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9)$:

$$\int_0^4 \frac{1}{\sqrt{1+x^2}}\,dx = \frac{h}{2}\{f(x_0)+2[f(x_1)+f(x_2)+f(x_3)+ \\ +f(x_4)+f(x_5)+f(x_6)+f(x_7)]+f(x_8)\} = 2,0936 \tag{5.8}$$

A partir dos cálculos anteriores concluímos que a aproximação para 9 subintervalos possui uma melhor precisão, dado que o valor real é 2,0947.

5.1.2 1/3 de Simpson

Ainda dentro do escopo das fórmulas de Newton-Cotes, estabelecemos o **método 1/3 de Simpson** por meio da interpolação da função *f(x)* usando um polinômio de grau n que coincide com essa função nos pontos x_0 $x_1, x_2, \ldots, x_n$. Generalizando a regra, subdividiremos o intervalo $[a,b]$ de integração em n subintervalos de amplitude h, onde n é um número par de subintervalos de forma que $x_0 = a$ e $x_n = b$, se aplicarmos em cada 2 subintervalos consecutivos, assim, consideramos que se n é um número par:

$$\int_a^b f(x)\,dx \approx \int_{x_0}^{x_2} f(x)\,dx + \int_{x_2}^{x_4} f(x)\,dx + \ldots + \int_{x_{n-2}}^{x_n} f(x)\,dx \approx \tag{5.9}$$

$$\approx \frac{h}{3}\cdot\{f(x_0)+4[f(x_1)+f(x_3)+\ldots+ \\ +f(x_{n-1})]+2[f(x_2)+f(x_4)+\ldots+f(x_{n-2})]+f(x_n)\} \tag{5.10}$$

pode-se traduzir o **método 1/3 de Simpson** no seguinte algoritmo:

```
1.   function SIMPSON(f(x_i), h) I ← 0 : n_par return I_S
2.   for i ← 1, 3, 5, ..., n − 1 do
3.     soma ← soma + 2 · f(x_i) + f(x_i+1)
4.   end for
5.   I_S = (h/3) · [2 · soma + f(x_0) − f(x_n)]
6.   end function
```

Algoritmo 10. Método 1/3 de Simpson.

Exemplo: Calcule o valor aproximado da integral a seguir utilizando o **método 1/3 de Simpson**, comparando os valores para 2, 4 e 6 subintervalos.

$$\int_0^3 \left(xe^x + 1\right) dx \tag{5.11}$$

A solução para 2 subintervalos:

$$h = \frac{\left(x_n - x_0\right)}{n} = 1,5 \tag{5.12}$$

x	$f\left(x\right)$
0	1,0000
1,5	7,7225
3,0	61,2566

$$\int_0^3 \left(xe^x + 1\right) dx = 46,5733 \tag{5.13}$$

$$= \frac{h}{3} \cdot \left[f(x_0) + 4f(x_1) + f(x_2)\right] \tag{5.14}$$

$$= \frac{1,5}{3} \cdot \left[1 + 4\left(7,7225\right) + 61,2566\right] \tag{5.15}$$

Solução para 4 subintervalos:

$$h = \frac{\left(x_n - x_0\right)}{n} = 0,75 \tag{5.16}$$

x	$f(x)$
0	1,0000
0,75	2,5878
1,5	7,7225
2,25	22,3474
3,0	61,2566

$$\int_0^3 \left(xe^x + 1\right) dx = 44,3606 \tag{5.17}$$

$$= \frac{h}{3} \cdot \{f(x_0) + 4\left[f(x_1) + f(x_3)\right] + 2f(x_2) + f(x_4)\} \tag{5.18}$$

$$= \frac{0,75}{3} \cdot \left\{1 + 4\left[2,5878 + 22,3474\right] + 2\left(7,7225\right) + 61,2566\right\} \tag{5.19}$$

Solução para 6 subintervalos:

$$h = \frac{\left(x_n - x_0\right)}{n} = 0,50 \tag{5.20}$$

x	$f(x)$
0	1,0000
0,5	1,8244
1,0	3,7183
1,5	7,7225
2,0	15,7781
2,5	31,4562
3,0	61,2566

$$\int_0^3 \left(xe^x + 1\right) dx = 44,2103 \tag{5.21}$$

$$= \frac{h}{3} \cdot \{f(x_0) + 4\left[f(x_1) + f(x_3) + f(x_5)\right] + 2f(x_2) + f(x_4) + f(x_6)\} \tag{5.22}$$

$$= \frac{0,50}{3} \cdot \begin{Bmatrix} 1 + 4\left[1,8244 + 7,7225 + 31,4562\right] + \\ +2\left(7,7225 + 15,7781\right) + 61,2566 \end{Bmatrix} \tag{5.23}$$

A tabela a seguir mostra os resultados da integração numérica para 2, 4 e 6 subintervalos utilizando o **método 1/3 de Simpson**:

Subintervalos	2	4	6
Valores	46,5733	44,3606	44,2103

A partir dos valores a seguir, conclui-se que quanto maior o número de subintervalos da integral, maior a aproximação do seu valor real.

5.2 Fórmulas Gaussianas

As Fórmulas Gaussianas, ou simplesmente, Quadraturas Gaussianas, para Integração Numérica, foram criadas para obter melhores resultados em comparação às fórmulas de Newton-Cotes. A ideia destes métodos fundamenta-se na aproximação da função integradora a um polinômio interpolador, mas nesse caso a polinômios ortogonais de alto grau, chamados polinômios de Legendre.

5.2.1 Método de Gauss

Dentre as muitas aproximações das Fórmulas Gaussianas, escolhemos um método de mesmo nome, chamado de **método de Gauss**. Esse método utiliza pontos diferentemente espaçados, onde este espaçamento é determinado pelos polinômios ortogonais de Legendre $(L_i(x))$. Sendo assim, usaremos o Polinômio Interpolador de Legendre para aproximar a integral, utilizando a combinação linear dos valores da função.

$$\int_a^b f(x)\,dx \approx \int_{-1}^{1} L_i f(x_i)\,dx \tag{5.24}$$

A seguir, serão estudados alguns aspectos do **método de Gauss** a partir de um problema genérico apresentado.

Exemplo: Calcule o valor aproximado da integral a seguir utilizando o **método de Gauss**, para 1 e 2 subintervalos.

$$\int_1^3 3e^x dx \tag{5.25}$$

Independentemente do número de etapas a serem calculadas, faz-se necessário realizar uma parametrização da função *f(x)* em referência ao polinômio de Legendre, isto é, faz-se uma mudança de variável simples:

$$\int_1^3 3e^x dx = \int_{-1}^1 F(t)\,dt = \sum_{i=0}^{n} P_i F(t_i) \tag{5.26}$$

$$x = \frac{1}{2}(b-a)t + \frac{1}{2}(b+a) = \frac{1}{2}(3-1)t + \frac{1}{2}(3+1) = t+2 \tag{5.27}$$

$$F(t) = 3e^{t+2} \tag{5.28}$$

Logo após, faz-se uma substituição dos coeficientes para <u>1 subintervalo</u>:

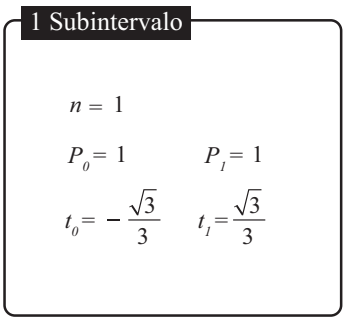


Figura 5.1. Valores das constantes para 1 subintervalo para o método de Gauss.

$$\int_1^3 f(x)\,dx = \int_{-1}^1 F(t)\,dt = P_0 F(t_0) + P_1 F(t_1) + P_2 F(t_2) \tag{5.29}$$

$$\int_1^3 f(x)\,dx = \int_{-1}^{1} F(t)\,dt = F\left(-\frac{\sqrt{3}}{3}\right) + F\left(\frac{\sqrt{3}}{3}\right) = 51{,}9303 \qquad (5.30)$$

De forma análoga, faz-se uma substituição dos coeficientes para 2 subintervalos:

2 Subintervalos

$n = 2$

$P_0 = \frac{5}{9}$ $\quad P_1 = \frac{8}{9}$ $\quad P_2 = \frac{5}{9}$

$t_0 = -\frac{\sqrt{3}}{5}$ $\quad t_1 = 0$ $\quad t_2 = \frac{\sqrt{3}}{5}$

Figura 5.2. Valores das constantes para 2 subintervalos para o método de Gauss.

$$\int_1^3 f(x)\,dx = \int_{-1}^{1} F(t)\,dt = P_0 F(t_0) + P_1 F(t_1) + P_2 F(t_2) \qquad (5.31)$$

$$= \frac{5}{9} F\left(-\frac{\sqrt{3}}{5}\right) + \frac{8}{9} F(0) + \frac{5}{9} F\left(\frac{\sqrt{3}}{5}\right) = 52{,}1004 \qquad (5.32)$$

A tabela a seguir mostra os resultados da integração numérica para 1 e 2 subintervalos utilizando o **método de Gauss**:

Subintervalos	1	2
Valores	51,9303	52,1004

De forma similar, os valores da tabela anterior concluem que quanto maior o número de subintervalos, maior a aproximação do seu valor real, só que atingido com um número baixo de subintervalos.

5.3 Exercícios Propostos

5.3.1 Exercícios Gerais

Exercício 5.3.1.1

Use o **método 1/3 de Simpson** com 2 subintervalos para calcular a integral:

$$\int_0^2 \left(3x^3 - 3x + 1\right) dx$$

Compare seu resultado com o valor exato da integral. Qual seria sua explicação para justificar um resultado tão bom?

Exercício 5.3.1.2

Calcular a integral numérica da curva dada pela tabela a seguir entre os pontos $x = 2$ e $x = 10$ pelo **método 1/3 de Simpson**:

x	2	4	6	8	10
$f(x)$	28	54	38	27	12

Exercício 5.3.1.3

Calcule a integral $\int_{-1}^{5} f(x)\,dx$ usando os **métodos dos Trapézios** e **1/3 de Simpson,** comparando os resultados obtidos com a forma analítica ou com alguma ferramenta numérica para os pontos a seguir:

x	−1	0	1	2	3	4	5
$f(x)$	0,61	0,62	0,82	1,91	1,92	1,99	2,14

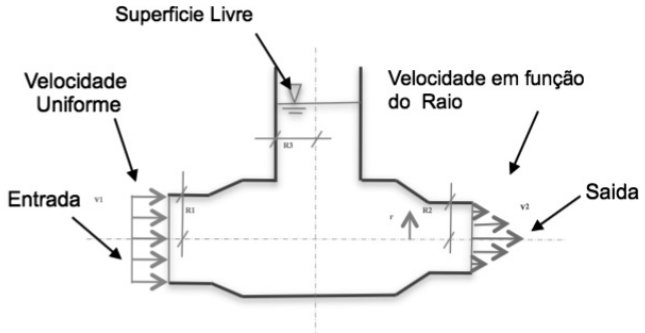
Exercício 5.3.1.4

Calcule a integral $\int_0^4 \sqrt{x}dx$ para 2 subintervalos usando o **método de Gauss**.

5.3.2 Exercícios Aplicados

Exercício 5.3.2.1 - Aplicado à Engenharia Química (Escoamento de um Fluído Incompressível por Tubulação)

A ilustração a seguir representa um esquema bastante comum na Engenharia Química, o escoamento de um fluído. Na taxonomia da Dinâmica dos Fluídos, um escoamento é considerado como incompressível se a densidade do fluído não muda em relação à pressão. Uma maneira bastante simples de entender esse conceito é imaginar partículas de um fluído movendo-se ao longo de uma trajetória bem definida, como em uma tubulação. Se imaginarmos que essa tubulação possui também uma superfície livre, constataremos que haverá uma perda de carga nas paredes da tubulação e que influenciará diretamente na velocidade do fluído. Tendo como base o esquema a seguir e os valores de velocidade parciais em função dos raios da tubulação, calcule a velocidade média desse fluído na saída. Faça uma análise comparativa das soluções, para valores similares de subintervalos, utilizando os **métodos de 1/3 de Simpson**, **Trapézios** e **Gauss**, considerando 4 casas decimais válidas.



x	$f(x)$
0	100,0000
0,05	99,2700
0,10	97,2200
0,15	93,7500
0,20	88,8900
0,25	82,6400
0,30	75,0000
0,35	65,9700
0,40	55,9700

Capítulo 6

RESOLUÇÃO NUMÉRICA DE SISTEMAS DE EQUAÇÕES NÃO LINEARES

A resolução de sistema de equações não lineares surge naturalmente durante a modelagem de problemas práticos relacionados às mais diferentes áreas. Uma série de aplicações práticas relacionadas com engenharia, física, ciência da computação, biologia, dentre outras, são solucionadas por meio dessas técnicas. Dentre elas destacam-se: problemas de cálculo de trajetórias, soluções de equações de transferência de radiação, modelagem de dinâmica populacional, problemas de estabilidade em aeronaves e outros. Além das aplicações citadas, ressaltamos também outros importantes problemas práticos que envolvem esse tipo de resolução, por exemplo, cálculo de estruturas eletrônicas e modelos complexos de transporte. Embora exista uma grande quantidade de métodos destinados a essa resolução, optamos por um variante do método de zeros da função explicado no capítulo anterior, adaptado a sistemas de equações não lineares, que apresenta bons resultados de convergência chamado de Newton-Raphson Modificado. As características desse método e suas inúmeras aplicações em casos relacionados às áreas da engenharia e ciências lhe asseguram uma grande popularidade e por inúmeras aplicações em casos relacionados às áreas da engenharia e ciências. Neste capítulo abordaremos o método de Newton-Raphson Modificado e mostraremos as diferenças estruturais em relação ao original.

6.1 Newton-Raphson Modificado

Seja o sistema de equações não lineares acoplados entre si a seguir. Consideramos para formular o problema, n funções $f_1, f_2, \ldots, f_n$ onde cada uma delas é uma função de n variáveis $x_1, x_2, \ldots, x_n$:

$$\begin{cases} f_1(x_1, x_2, \ldots, x_n) = 0 \\ f_2(x_1, x_2, \ldots, x_n) = 0 \\ \vdots \\ f_n(x_1, x_2, \ldots, x_n) = 0 \end{cases} \tag{6.1}$$

Na tentativa de simplificar, usaremos uma representação vetorial, tal que $x = (x_1, x_2, \ldots, x_n)$ é um vetor com n componentes e $F(x)$ é uma função vetorial de variáveis:

$$F(x) = \left(f_1(x), f_2(x), \ldots, f_n(x)\right) \tag{6.2}$$

Representando o sistema de equações por uma única equação vetorial, temos que:

$$F(x) = 0 \tag{6.3}$$

Como sabemos, a série de Taylor é uma outra maneira de se obter a função de iteração do método de Newton, sendo assim fazemos uma linearização por meio desta série:

$$F(x) = F\left(x^{(k)}\right) + F'(x^{(k)})\left(x - x^{(k)}\right) \tag{6.4}$$

Tendo como base a teoria do cálculo diferencial e integral, a derivada das funções representa uma matriz que contém todas as derivadas parciais de todos os componentes da função $F(x)$, chamada de matriz Jacobiana $J(x)$:

$$J(x) = F'(x) = J_{ij} = \frac{\partial f_i(x)}{\partial x_j} \tag{6.5}$$

de tal forma que expressamos a representação matricial como:

$$J\left(x^{(k)}\right) = \begin{pmatrix} \frac{\partial f_1 x^{(k)}}{\partial x_1} & \frac{\partial f_1 x^{(k)}}{\partial x_2} & \cdots & \frac{\partial f_1 x^{(k)}}{\partial x_n} \\ \frac{\partial f_2 x^{(k)}}{\partial x_1} & \frac{\partial f_2 x^{(k)}}{\partial x_2} & \cdots & \frac{\partial f_2 x^{(k)}}{\partial x_n} \\ \vdots & \vdots & & \vdots \\ \frac{\partial f_n x^{(k)}}{\partial x_1} & \frac{\partial f_n x^{(k)}}{\partial x_2} & \cdots & \frac{\partial f_n x^{(k)}}{\partial x_n} \end{pmatrix} \tag{6.6}$$

Considerando $x^{(k+1)}$ igual a:

$$x^{(k+1)} = x^{(k)} - \left[J(x^{(k)}) \right]^{-1} F\left(x^{(k)}\right) \tag{6.7}$$

E fazendo $\Delta x^{(k)} = x^{(k+1)} - x^{(k)}$, temos:

$$\Delta x^{(k)} = -[J(x^{(k)})]^{-1} F\left(x^{(k)}\right) \tag{6.8}$$

Sendo assim temos uma expressão genérica aproximada para calcular o sistema de equações não lineares, dado por:

$$J(x^{(k)}) \cdot \Delta x^{(k)} = -F\left(x^{(k)}\right) \tag{6.9}$$

A resolução de sistemas de equações não lineares é simplesmente a formulação de um algoritmo numérico iterativo que manipula as funções contidas em $F(x)$ para vários valores de $\boldsymbol{x^{(k)}}$. A seguir será estudado o **método de Newton-Raphson Modificado** a partir de um problema genérico.

Exemplo: Encontre a solução do **sistema de equações não lineares** utilizando o **método de Newton-Raphson Modificado** para uma tolerância de $0{,}1000$ e um vetor inicial $x^{(0)} = (0{,}5000,\ 0{,}5000,\ 0{,}5000)^t$.

$$\begin{cases} x_1^{\,2} + x_2^{\,2} + x_3^{\,2} = 1 \\ 2x_1^{\,2} + x_2^{\,2} - 4x_3 = 0 \\ 3x_1^{\,2} - 4x_2 + x_3^{\,2} = 0 \end{cases} \tag{6.10}$$

Como o processo iterativo é dado pela equação:

$$J(x^{(k)}) \cdot \Delta x^{(k)} = -F\left(x^{(k)}\right) \tag{6.11}$$

Iteração 1:

O primeiro passo é calcular a matriz Jacobiana:

$$J\left(x^{(k)}\right) = \begin{pmatrix} 2x_1^{(k)} & 2x_2^{(k)} & 2x_3^{(k)} \\ 4x_1^{(k)} & 2x_2^{(k)} & -4 \\ 6x_1^{(k)} & -4 & 2x_3^{(k)} \end{pmatrix} \tag{6.12}$$

Para $x^{(0)} = (0{,}5000,\ 0{,}5000,\ 0{,}5000)^t$, tem-se o Jacobiano:

$$J\left(x^{(0)}\right) = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 1 & -4 \\ 3 & -4 & 1 \end{pmatrix} \tag{6.13}$$

Para $x^{(0)} = (0{,}5000,\ 0{,}5000,\ 0{,}5000)^t$, tem-se $F(x^{(0)})$:

$$\begin{cases} 0{,}5000^2 + 0{,}5000^2 + 0{,}5000^2 - 1 = -0{,}2500 \\ 2(0{,}5000)^2 + 0{,}5000^2 - 4(0{,}5000) = -1{,}2500 \\ 3(0{,}5000)^2 - 4(0{,}5000) + 0{,}5000^2 = -1{,}0000 \end{cases} \tag{6.14}$$

$F\left(x^{(0)}\right) = (-0{,}2500,\ -1{,}2500,\ -1{,}0000)$. Tem-se o sistema linear:

$$\begin{pmatrix} 1 & 1 & 1 \\ 2 & 1 & -4 \\ 3 & -4 & 1 \end{pmatrix} \cdot \begin{pmatrix} \Delta x_1^{(0)} \\ \Delta x_2^{(0)} \\ \Delta x_3^{(0)} \end{pmatrix} = \begin{pmatrix} 0{,}2500 \\ 1{,}2500 \\ 1{,}0000 \end{pmatrix} \tag{6.15}$$

Resolvendo o sistema linear de equações, obtêm-se $\Delta x^{(0)} = (0{,}3750,\ 0{,}0000,\ -0{,}1250)^t$. Os novos valores do vetor x são dados por:

$$x^{(1)} = x^{(0)} + \Delta x^{(0)} = \begin{pmatrix} 0{,}5000 \\ 0{,}5000 \\ 0{,}5000 \end{pmatrix} + \begin{pmatrix} 0{,}3750 \\ 0 \\ -0{,}1250 \end{pmatrix} = \begin{pmatrix} 0{,}8750 \\ 0{,}5000 \\ 0{,}3750 \end{pmatrix} \tag{6.16}$$

O critério de parada só satisfaz para a segunda coordenada, logo o método continua com os novos valores do vetor x:

$$\begin{cases} \dfrac{\left|x_1^{\,1} - x_1^{\,0}\right|}{\left|x_1^{\,1}\right|} = \dfrac{\left|0,8750 - 0,5000\right|}{\left|0,8750\right|} = 0,4286 > 0,1000 \\ \\ \dfrac{\left|x_2^{\,1} - x_2^{\,0}\right|}{\left|x_2^{\,1}\right|} = \dfrac{\left|0,5000 - 0,5000\right|}{\left|0,5000\right|} = 0,0000 < 0,1000 \\ \\ \dfrac{\left|x_3^{\,1} - x_3^{\,0}\right|}{\left|x_3^{\,1}\right|} = \dfrac{\left|0,3750 - 0,5000\right|}{\left|0,3750\right|} = 0,3333 > 0,1000 \end{cases} \tag{6.17}$$

Iteração 2:

Para $x^{(1)} = \left(0,8750,\ 0,5000,\ 0,3750\right)^t$, tem-se o Jacobiano:

$$J\left(x^{(1)}\right) = \begin{pmatrix} 1,7500 & 1,0000 & 0,7500 \\ 3,5000 & 1,0000 & -4,0000 \\ 5,2500 & -4,0000 & 0,7500 \end{pmatrix} \tag{6.18}$$

Para $x^{(1)} = \left(0,8750,\ 0,5000\ 0,3750\right)^t$, tem-se $F(x^{(1)})$, que é calculado por:

$$\begin{cases} 0,8750^2 + 0,5000^2 + 0,3750^2 = 1,1563 \\ 2\left(0,8750\right)^2 + 0,5000^2 - 4\left(0,3750\right) = 0,2813 \\ 3\left(0,8750\right)^2 - 4\left(0,5000\right) + 0,3750^2 = 0,4375 \end{cases} \tag{6.19}$$

$F\left(x^{(1)}\right) = \left(0,1563,\ 0,2813,\ 0,4375\right)$. Tem-se o sistema linear:

$$\begin{pmatrix} 1,7500 & 1,0000 & 0,7500 \\ 3,5000 & 1,0000 & -4,0000 \\ 5,2500 & -4,0000 & 0,7500 \end{pmatrix} \cdot \begin{pmatrix} \Delta x_1^{(1)} \\ \Delta x_2^{(1)} \\ \Delta x_3^{(1)} \end{pmatrix} = \begin{pmatrix} -0,1563 \\ -0,2813 \\ -0,4375 \end{pmatrix} \tag{6.20}$$

Resultando em $\Delta x^{(1)} = \left(-0,00852,\ -0,0034,\ -0,0051\right)^t$. Os novos valores do vetor x são dados por:

$$x^{(2)} = x^{(1)} + \Delta x^{(1)} = \begin{pmatrix} 0,8750 \\ 0,5000 \\ 0,3750 \end{pmatrix} + \begin{pmatrix} -0,0085 \\ -0,0034 \\ -0,0051 \end{pmatrix} = \begin{pmatrix} 0,7898 \\ 0,4966 \\ 0,3699 \end{pmatrix} \qquad (6.21)$$

O critério de parada só satisfaz para a segunda e a terceira coordenadas, logo o método continua com os novos valores do vetor x:

$$\begin{cases} \dfrac{\left|x_1^{\,2} - x_1^{\,1}\right|}{\left|x_1^{\,2}\right|} = \dfrac{\left|0,7898 - 0,8750\right|}{\left|0,7898\right|} = 0,1079 > 0,1000 \\ \dfrac{\left|x_2^{\,2} - x_2^{\,1}\right|}{\left|x_2^{\,2}\right|} = \dfrac{\left|0,4966 - 0,5000\right|}{\left|0,4966\right|} = 0,0068 < 0,1000 \\ \dfrac{\left|x_3^{\,2} - x_3^{\,1}\right|}{\left|x_3^{\,2}\right|} = \dfrac{\left|0,3699 - 0,3750\right|}{\left|0,3699\right|} = 0,0138 < 0,1000 \end{cases} \qquad (6.22)$$

Iteração 3:

Para $x^{(2)} = \left(0,7898,\ 0,4966,\ 0,3699\right)$, tem-se o Jacobiano:

$$J\left(x^{(2)}\right) = \begin{pmatrix} 1,5796 & 0,9932 & 0,7399 \\ 3,1593 & 0,9932 & -4,0000 \\ 4,7389 & -4,0000 & 0,7399 \end{pmatrix} \qquad (6.23)$$

Para $x^{(2)} = \left(0,7898,\ 0,4966,\ 0,3699\right)^t$, tem-se que $F(x^{(2)})$ que é calculado por:

$$\begin{cases} 0,7898^2 + 0,4966^2 + 0,3699^2 = 1,0073 \\ 2\left(0,7898\right)^2 + 0,4966^2 - 4\left(0,3699\right) = 0,0145 \\ 3\left(0,7898\right)^2 - 4\left(0,4966\right)^2 + 0,3699^2 = 0,0218 \end{cases} \qquad (6.24)$$

Para $F\left(x^{(2)}\right) = (0,0073,\ 0,0145,\ 0,0218)$ tem-se o sistema linear:

$$\begin{pmatrix} 1,5796 & 0,9932 & 0,7399 \\ 3,1593 & 0,9932 & -4,0000 \\ 4,7389 & -4,0000 & 0,7399 \end{pmatrix} \cdot \begin{pmatrix} \Delta x_1^{(2)} \\ \Delta x_2^{(2)} \\ \Delta x_3^{(2)} \end{pmatrix} = \begin{pmatrix} -0,0073 \\ -0,0145 \\ -0,0218 \end{pmatrix} \qquad (6.25)$$

Resultando em $\Delta x^{(2)} = (-0,0046,\ 0,0000,\ 0,0000)^t$. Os novos valores do vetor x são dados por:

$$x^{(3)} = x^{(2)} + \Delta x^{(2)} = \begin{pmatrix} 0,7898 \\ 0,4966 \\ 0,3659 \end{pmatrix} + \begin{pmatrix} -0,0046 \\ 0,0000 \\ 0,0000 \end{pmatrix} = \begin{pmatrix} 0,7852 \\ 0,4966 \\ 0,3699 \end{pmatrix} \qquad (6.26)$$

Por fim o critério de parada satisfaz todas as coordenadas, logo o método finaliza com a solução do sistema com os valores do vetor x:

$$\begin{cases} \dfrac{\left|x_1^{\,3} - x_1^{\,2}\right|}{\left|x_1^{\,3}\right|} = \dfrac{|0,7852 - 0,7898|}{|0,7852|} = 0,0059 < 0,1000 \\ \\ \dfrac{\left|x_2^{\,3} - x_2^{\,2}\right|}{\left|x_2^{\,3}\right|} = \dfrac{|0,4966 - 0,4966|}{|0,4966|} = 0,0000 < 0,1000 \\ \\ \dfrac{\left|x_3^{\,3} - x_3^{\,2}\right|}{\left|x_3^{\,3}\right|} = \dfrac{|0,3659 - 0,3699|}{|0,3659|} = 0,0109 < 0,1000 \end{cases} \qquad (6.27)$$

Solução: O **método de Newton-Raphson Modificado** converge para uma tolerância de $0,1000$ e converge na 3ª iteração, com o seguinte resultado:

$$x^{(3)} = (0,7852,\ 0,4966,\ 0,3699)^t \qquad (6.28)$$

6.2 Exercícios Propostos

6.2.1 Exercícios Gerais

Exercício 6.2.1.1

Resolver os seguintes sistemas de equações não lineares, usando o **método de Newton Raphson Modificado** com uma tolerância de 10^{-1}, utilizando 4 dígitos significativos:

a) $\begin{cases} x_1 + 2x_2 = 3 \\ 3x_1^{\,2} + x_2 = 7 \end{cases}$, $x_0 = (0{,}5,\ 0{,}5)$

b) $\begin{cases} x^2 + y^2 = 2 \\ x^2 - y^2 = 1 \end{cases}$, $x_0 = (1{,}2,\ 0{,}7)$

c) $\begin{cases} 10(y - x^2) = 0 \\ 1 - x = 0 \end{cases}$, $x_0 = (-1{,}2,\ 1)$

d) $\begin{cases} x^2 + y^2 = 2 \\ e^{x-1} + y^3 = 2 \end{cases}$, $x_0 = (1{,}5,\ 2)$

6.2.2 Exercícios Aplicados

Exercício 6.2.2.1 - Aplicado à Engenharia Civil (Tensão em Solos)

A pressão total necessária para afundar um objeto grande e pesado em um solo homogêneo é dada pela seguinte equação:

$$P = k_1 e^{k_2 r} k_3 r$$

Para determinar o tamanho mínimo de uma placa para sustentar uma grande carga, produzem-se 3 equações não lineares:

$$\begin{cases} m_1 = k_1 e^{k_2 r_1} + k_3 r_1 \\ m_2 = k_1 e^{k_2 r_2} + k_3 r_2 \\ m_3 = k_1 e^{k_2 r_3} + k_3 r_3 \end{cases}$$

Solucione o sistema de equações não lineares utilizando o **método de Newton-Raphson Modificado**, considerando 4 casas decimais válidas, $m_0 = (x, y, z)$ e tolerância de 10^{-2}.

Capítulo 7

RESOLUÇÃO NUMÉRICA DE EQUAÇÕES DIFERENCIAIS ORDINÁRIAS

Modelos matemáticos usados em problemas aplicados em sua grande maioria envolvem equações cujas incógnitas são funções de uma variável e de suas derivadas. Uma equação diferencial ordinária (EDO) é definida como uma equação que envolve uma função incógnita y e algumas de suas derivadas avaliadas em uma variável independente x, da forma:

$$y'(x) = f\left(x, y(x)\right) \ e \ y''(x) = f\left(x, y(x), y'\right) \tag{7.1}$$

em que $f(x, y)$ e $f(x, y, y')$ são funções conhecidas. Para definirmos técnicas para obter $y(x)$, temos de fornecer dados adicionais à EDO. Os dados adicionais podem definir o problema a ser resolvido, para o qual são necessárias condições de contorno adicionais, chamado de problema do valor inicial (PVI). Na falta de soluções analíticas para equações diferenciais, os métodos numéricos têm atuado como importante ferramenta para problemas aplicados à engenharia e às ciências. Neste capítulo, faremos uso de $y(x)$ para denotar soluções de EDOs com PVIs, apresentando algumas técnicas úteis no cálculo dessas aproximações numéricas, tais como os métodos de:

- Euler;
- Runge-Kutta.

O ponto de partida para a resolução de uma EDO é uma condição inicial de contorno associada a uma EDO. Essa condição é um ponto x que pertence ao domínio da função $f(x)$ o qual é solução do PVI. Nosso objetivo é que a função $y(x)$, satisfaça a EDO para $x > a$ e atenda às condições preestabelecidas no início do intervalo de resolução, ou seja:

$$y(a) = y_0 \quad e \quad y'(a) = v_0 \tag{7.2}$$

sendo conhecidas as condições adicionais, podemos obter soluções particulares para a EDO e se não são conhecidas condições adicionais poderemos obter a solução geral.

7.1 Resolução Analítica versus Numérica

Existem várias formas de se solucionar as EDOs, dentre as quais se destacam preponderantemente as formas analíticas e as numéricas. A grande maioria dos métodos analíticos tem como base a separação de variáveis a partir de processos de integração, sendo que nem sempre é possível obter uma solução analítica, e em muitas vezes, torna-se mais complexa tal resolução. Os métodos analíticos têm como característica base a análise qualitativa, expressando um comportamento geométrico das soluções e os aspectos das curvas integrais descritos por meio de campos de direções. A seguir, pode-se exemplificar resoluções analíticas de problemas aplicáveis de EDOs por meio de exemplos.

Exemplo: A partir da EDO a seguir, conseguiu-se caracterizar o crescimento de uma população de bactérias com uma taxa proporcional ao numero de indivíduos. Foi considerado com PVI ($N(t=0) = N_0$).

$$\frac{dN}{dt} = K \cdot N \tag{7.3}$$

Solução: Separando e integrando as variáveis da EDO simultaneamente, que pode ser mostrado a seguir:

$$\int \frac{dN}{dt} = \int K \cdot N \tag{7.4}$$

$$\int \frac{dN}{N} = \int K \cdot dt \tag{7.5}$$

$$\int \frac{dN}{N} = K \int dt \tag{7.6}$$

$$\ln(N) = Kt + C \tag{7.7}$$

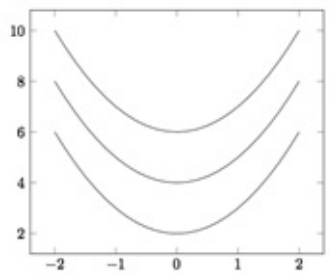
$$N = e^{Kt} + e^{C} \tag{7.8}$$

Aplicando a condição inicial $N(0) = N_0$, teremos como solução:

$$N(t) = N_0 \cdot e^{Kt} \tag{7.9}$$

Exemplo: Pode-se descrever uma Parábola na forma explícita, onde o valor de y para cada x descreve a inclinação em cada ponto da mesma a partir da seguinte equação diferencial ordinária $dy/dx = 2x$.

Solução: Separando e integrando as variáveis da EDO simultaneamente, tem-se o seu resultado gráfico para a solução genérica, que pode ser mostrado a seguir:



$$\frac{dy}{dx} = 2x \Rightarrow dy = 2xdx \tag{7.10}$$

$$\int dy = 2\int xdx \tag{7.11}$$

$$y = x^2 + C \tag{7.12}$$

Figura 8.1. Solução genérica para o exemplo anterior.

Exemplo: O desenvolvimento de um sistema de controle de nível de um reservatório contendo água, pode ser modelado por meio de EDOs, conforme a ilustração a seguir:

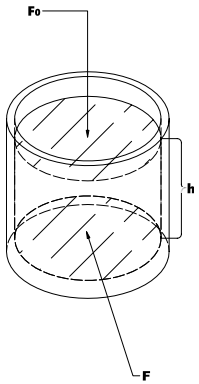


Figura 8.2. Esquema de um sistema de controle de nível de um reservatório contendo água.

O primeiro passo seria considerar o balanço mássico de matéria que entra e sai do reservatório:

$$Entra - Sai + Gerado - Acumula = 0 \tag{7.13}$$

Como não há reação por ser água, temos que:

$$Entra - Sai = Acumula \tag{7.14}$$

Aplicando os parâmetros da modelagem:

$$F_o - F = \frac{dV}{dt} \tag{7.15}$$

Como o volume do recipiente é cilíndrico, temos $V = A \cdot h$,

$$F_o - F = A\frac{dh}{dt} \tag{7.16}$$

$$\frac{F_o - F}{A} = \frac{dh}{dt} \tag{7.17}$$

$$\frac{dh}{dt} = \frac{1}{A}\left(F_o - F\right) \tag{7.18}$$

$$\int dh = \frac{1}{A}\int\left(F_o - F\right)dt \tag{7.19}$$

$$h(t) = \frac{1}{A}\left(F_o - F\right)t \tag{7.20}$$

Aplicando a Segunda Lei de Newton que aborda o escoamento gravitacional, temos:

$$h(t) = R \cdot F_o\left(1 - e^{-\frac{t}{R \cdot A}}\right) \tag{7.21}$$

O controle do sistema anterior pode ser classificado como controle em malha aberta, isto é, nenhuma informação da saída real do sistema é utilizado para modificar a entrada. Em contraposição o esquema de malha fechada pode ser esquematizado a seguir:

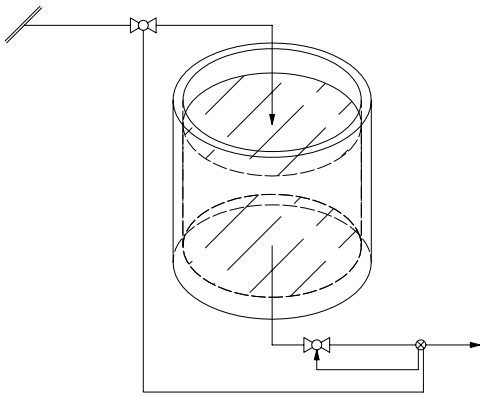


Figura 8.3. Esquema de um sistema de controle de nível de um reservatório contendo água, por meio da inserção de uma válvula.

A inserção de uma válvula adicionada a um controlador proporcional integral e/ou derivativo automatiza o processo a nível de um controle do processo apresentado.

7.2 Métodos Explícitos de Passo Simples

Neste capítulo teremos como escopo os métodos numéricos de resolução de EDOs ditos de **passo simples**, isto é, quando o cálculo de todo y_{i+1} é realizado em função apenas de x_i e y_i; e **explícitos**, que ocorrem quando todas as grandezas no lado direito da equação são conhecidas:

$$y_{i+1} = f\left(x_i, y_i, x_{i-1}, \ldots, x_{i-n}, y_{i-n}\right) \tag{7.22}$$

7.2.1 Euler

A essência de um método numérico para a resolução de uma EDO está na discretização dos valores nos quais se busca a solução, pois a mensuração do problema em um número finito de cálculos torna possível a aplicação desse método. Como uma incógnita de uma EDO é uma função $y(x)$ definida em todos os pontos de intervalo no

qual a equação está sendo resolvida. Tendo uma equação diferencial de uma certa ordem:

$$\frac{dy^n}{d^n x} = f(x, y) \rightarrow PVI : y(x_1) = y_1 \tag{7.23}$$

O **método de Euler** faz o cálculo das derivadas presentes na equação diferencial ordinária substituindo-as por aproximações, isto é, para cada ponto da equação existe uma solução numérica aproximada em (x_{i+1}, y_{i+1}), sendo calculada a partir da solução conhecida no ponto (x_i, y_i):

$$\begin{cases} x_{i+1} = x_i + h \\ y_{i+1} = y_i + f(x_i, y_i) \cdot h \end{cases} \tag{7.24}$$

onde h é o passo de integração e a inclinação é uma constante que estima dy/dx no intervalo de x_i a x_{i+1}.

Exemplo: Calcule o valor aproximado da EDO de 1ª ordem a seguir utilizando o **método explícito de Euler** com **passo simples**, considerando $h = 0,5$ no intervalo $[0, 1,5]$:

$$\frac{dy}{dx} = -2x^3 + y \rightarrow PVI : y(0) = 1 \tag{7.25}$$

Na tabela a seguir, apresentam-se os valores calculados usando $h = 0,5$ para o intervalo e condições de contorno propostas. As colunas representam o índice da iteração, os valores x_i e y_i, respectivamente.

i	x_i	y_i
0	0	1,0000
1	0,5	1,5000
2	1,0	2,1250
3	1,5	2,1875

Tabela 7.1. Resultados compilados do exemplo anterior.

7.2.2 Runger-Kutta

O **método de Runge-Kutta** é o mais utilizado dentre aqueles apropriados para os problemas de valor inicial. Destacam-se como principais características: a simplicidade, a alta precisão e versalidade nas aplicações. Tal método pode ser expresso de uma maneira geral:

$$\begin{cases} x_{i+1} = x_i + h \\ y_{i+1} = y_i + f\left(x_i, y_i\right) \cdot h + \dfrac{h^2}{2!} \cdot f''\left(x_i, y_i\right) \end{cases} \tag{7.26}$$

onde f' e f'' são chamadas de funções de incremento, que podem ser consideradas representativa da inclinação do intervalo h.

Exemplo: Calcule o valor aproximado da EDO a seguir utilizando o **método explícito de Runge Kutta** com **passo simples**, considerando $h = 0{,}1$ no intervalo $[0, 1]$:

$$\frac{dy}{dx} = x - y + 2 \rightarrow PVI : y(0) = 2 \tag{7.27}$$

Na tabela a seguir, apresentam-se os valores calculados usando $h = 0{,}1$ para o intervalo e as condições de contorno propostas. As colunas representam o índice da iteração, os valores x_i e y_i, respectivamente.

i	x_i	y_i
0	0	2
1	0,1	2,0050
2	0,2	2,0190
3	0,3	2,0412
4	0,4	2,0708
5	0,5	2,1070
6	0,6	2,1494
7	0,7	2,1972
8	0,8	2,2499
9	0,9	2,3072
10	1,0	2,3685

Tabela 7.2. Resultados compilados do exemplo anterior.

7.3 Exercícios Propostos

7.3.1 Exercícios Gerais

Exercício 7.3.1.1

Dada a seguinte EDO de 1ª ordem:

$$\frac{dy}{dx} = -1,2y + 7e^{-0,3x} \rightarrow PVI : y(0) = 3$$

Considerando o intervalo de $x = 0$ à $x = 2$, resolva a equação diferencial ordinária anterior para $h = 0,5$. Compare os resultados com a solução (analítica) exata a seguir, utilizando os **métodos explícitos de passo simples de Euler e Runger-Kutta**.

$$y = \frac{70}{9}e^{-0,3x} - \frac{43}{9}e^{-1,2x}$$

Exercício 7.3.1.2

Dada a seguinte EDO de 1ª ordem:

$$\frac{dy}{dx} = yx - x^3 \rightarrow PVI : y(0) = 1$$

Considerando o intervalo de $x = 0$ à $x = 1,8$, resolva a equação diferencial ordinária anterior para $h = 0,6$. Compare os resultados com a solução (analítica) exata a seguir, utilizando o **método explícito de passo simples de Runger-Kutta**.

$$y = x^2 - e^{0,5x^2} + 2$$

7.3.2 Exercícios Aplicados

Exercício 7.3.2.1 – Aplicado à Engenharia Civil (Cálculo do Momento Fletor em Vigas)

O momento fletor máximo de uma viga de concreto armado, $M(x)$ em função do deslocamento da largura (l) da viga, situado no primeiro andar de um edíficio, é conhecido e igual a:

$$M(x) = \frac{1}{2}\left(plx - px^2\right)$$

sendo p a carga de partida e E é o fator elástico da viga. Sabendo que a equação diferencial ordinária associada à viga é:

$$Ey'' = M(x)$$

Pode-se calcular do momento fletor máximo para a viga em função dos seguintes parâmetros: p, l, E. Considerando as condições de contorno $y(0) = 0$ e $y(l) = 0$.

Capítulo 8

PRINCÍPIOS DE OTIMIZAÇÃO

Alguns métodos numéricos que consistem na determinação de uma solução ou um conjunto de soluções ótimas para uma determinada função ou conjunto de funções encontram-se na área da otimização, sendo a sua base fundamentada no conceito de solução ótima, a qual é inerente ao problema que se deseja otimizar. Por exemplo, em uma situação em que a modelagem matemática por uma única função na qual existe a necessidade de determinar um valor ótimo, tal que seja mínimo, ou máximo, ou em uma situação cujo modelo seja expresso por múltiplas funções, na qual pretende-se otimizar. Nesse caso, pode-se ter uma única solução, um conjunto de soluções ou ainda não haver solução que satisfaça todas as funções. À medida que o número de funções e o número de variáveis aumentam, a dificuldade em se determinar o conjunto de soluções ótimas também aumenta. É nesse contexto que surge a necessidade de desenvolver técnicas matemáticas e computacionais que refinem o processo de otimização, dado que este é amplamente utilizado para resolver problemas aplicados.

Os métodos para a solução de problemas de otimização dividem-se em dois grupos: os baseados em cálculos determinísticos e os estocásticos. No grupo dos métodos estocásticos mais utilizados estão: Algoritmos Genéticos, *Simulated Annealing*, Redes Neurais, Evolução Diferencial etc. Quanto à presença de limitantes ao problema, tem-se a otimização sem restrições e a otimização com restrições. Na otimização com restrições existem os métodos indiretos e os diretos.

O objetivo deste capítulo são os métodos de otimização sem restrição e aplicá-los na solução de problemas nas ciências e engenharias. Sendo assim, estudaremos um grupo particular de métodos determinísticos, baseados em aproximações. Essas técnicas são aplicadas em processos de minimizacão e maximização de problemas de otimização.

8.1 Métodos Numéricos Unimodais e Univariáveis

Define-se uma função $f(x)$ como unimodal e como univariável em um intervalo $[a,b]$ se existe um único $x \in [a,b]$ de tal modo que, algum $x_1, x_2 \in [a,b]$, para $x_1 < x_2$, logo:

$$f(x) = \begin{cases} f(x_1) > f(x_2), \text{ se } x_2 < x^* \\ f(x_1) < f(x_2), \text{ se } x_1 < x^* \end{cases} \quad (8.1)$$

A partir da definição anterior é possível concluir que se a função $f(x)$ é unimodal e univariável em $[a,b]$, faz-se possível a redução do intervalo e a comparção dos valores de $f(x)$ em $[a,b]$ a dois pontos prévios.

Os métodos unimodais e univariáveis são simplesmente algoritmos numéricos que manipulam a função $f(x)$ a fim de encontrar de forma mais aproximada os valores de máximo ou mínimo. A seguir, serão estudados alguns métodos a partir de um problema genérico apresentado:

Exemplo: Encontre o **máximo** da função $f(x) = 9x - 0{,}1x^2$ no intervalo de 0 a 100 com uma tolerância de $0{,}5\%$ do intervalo inicial. Utilize os métodos unimodais e univariáveis (considere 4 casas decimais válidas).

A partir do problema enunciado será possível explicar 4 métodos unimodais e univariáveis: **5 Pontos**, **Dicótoma**, **Fibonacci** e **Secção Áurea**.

8.1.1 5 Pontos

O **método dos 5 Pontos** tem como base buscar um valor ótimo em um determinado intervalo dividido por 5 pontos a uma distância equidistante. O algoritmo de busca consiste em checar se a cada passo ocorre a condição de parada, e neste caso sendo encontrado o valor ótimo. O método caracteriza-se por ser recursivo, já que a busca elimina a cada passo regiões de valores máximos ou mínimos (dependendo de sua condição unimodal). A seguir encontramos um

fluxograma com os passos de resolução por meio do método, e a descrição de cada passo de forma explicativa:

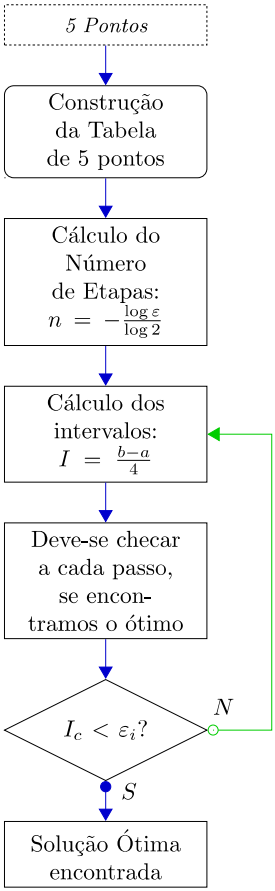


Figura 8.1. Fluxograma do método dos 5 Pontos.

Para o problema, dado anteriormente, a função $f(x) = 9x - 0{,}1x^2$, tem como condição suficiente e necessária a transformação do valor porcentual da tolerância (ε) em um valor numérico relativo (ε_I) ao intervalo dado.

$$\varepsilon_I = 0{,}5\% \; de \; [0, 100] \rightarrow \frac{0{,}5}{100} \cdot (100 - 0) = 0{,}5 \qquad (8.2)$$

Assim temos que o valor dado de $0,5\%$ corresponde ao valor numérico de $0,5$. De forma subsequente o cálculo de intervalos tem como base os 5 pontos da tabela com 4 subintervalos, logo o valor numérico do primeiro intervalo (I) deverá ser calculado da seguinte forma:

$$I = \frac{b-a}{4} = \frac{100-0}{4} = 25 \tag{8.3}$$

E para se ter uma previsão do número de etapas, substitui-se na fórmula a seguir, o valor numérico da tolerância:

$$n = -\frac{\log \varepsilon}{\log 2} = -\frac{\log 0,005}{\log 2} = 7,6400 \cong 8 \ \textit{etapas} \tag{8.4}$$

E a cada passo calculamos o critério de parada que consiste em calcular uma discrepância entre os valores de intervalos calculados (I_c) e a tolerância numérica relativa calculada (ε_I):

$$I_c = 45,1172 - 44,7266 < \varepsilon_I \rightarrow 0,3907 < 0,5000 \tag{8.5}$$

A tabela a seguir representa a resolução do **método dos 5 Pontos**. Como o método fundamenta-se na eliminação de região, tem-se nas áreas sublinhadas as áreas eliminadas. Recordando sempre a característica inversa da eliminação da região, isto é, para valores de **máximo** se eliminam as regiões com os 2 menores valores numéricos para $f(x)$ na etapa calculada, e para o **mínimo** o procedimento inverso. Os últimos quadros representam os valores intervalados do ótimo, e o máximo para $f(x)$.

0,0000	**25,0000**	**50,0000**	**75,0000**	**100,0000**
0,0000	162,5000	200,0000	112,5000	-100,0000
25,0000	**37,5000**	**50,0000**	**62,5000**	**75,0000**
162,5000	196,8750	200,0000	189,8437	171,8750
37,5000	**43,7500**	**50,0000**	**56,2500**	**62,5000**
196,8750	202,3437	200,000	189,8437	171,8750
37,5000	**40,6250**	**43,7500**	**46,8750**	**50,0000**
196,8750	200,5859	202,3437	202,1484	200,0000
40,6250	**42,1875**	**43,7500**	**45,3125**	**46,8750**
200,5859	201,7090	202,3437	202,4902	202,1484
43,7500	**44,5312**	**45,3125**	**46,0937**	**46,8750**
202,3437	202,4780	202,4902	202,3804	202,1484
44,5312	**44,9218**	**45,3125**	**45,7031**	**46,0937**
202,4780	202,4994	202,4902	202,4506	202,3804
44,5312	**44,7265**	**44,9218**	**45,1172**	**45,3125**
202,4780	202,4925	202,4994	202,4986	202,4902

Tabela 8.1. Tabela com a solução do método dos 5 Pontos.

Solução: O **método dos 5 Pontos** converge para um valor de máximo de $202,4994$ para o intervalo $[44,7266,\ 45,1172]$, sendo consideradas 4 casas decimais e tolerância de $0,5\%$ do intervalo dado.

8.1.2 Dicótoma

O **método da Dicótoma** consiste em uma forma bastante simples e eficiente de eliminação de região a fim de obter o valor ótimo da função. A partir de 2 pontos de busca localizados no centro da região, separados por uma distância ε_I, eliminamos regiões até encontrar o máximo ou o mínimo da função. Tal que, o critério de parada consista em comparar o intervalo calculado com a tolerância a cada iteração, isto é, $I_c < \varepsilon_I$ de forma análoga ao método anterior. A seguir temos a resolução do problema anterior, utilizando o conceito de dicotomia.

i	I	x_i	$f(x_i)$
1	0	49,7500	200,2437
	100	~~50,2500~~	~~199,7437~~
2	0	~~24,7500~~	~~161,4937~~
	50	25,2500	163,4937
3	25	~~37,2500~~	~~196,4937~~
	50	37,7500	197,2437
4	37,5000	~~43,5000~~	~~202,2750~~
	50	44,0000	202,4000
5	43,7500	46,6250	202,2359
	50	~~47,1250~~	~~202,0484~~
6	43,7500	45,0625	202,4996
	46,8750	~~45,5625~~	~~202,4684~~
7	43,7500	~~44,2812~~	~~202,4883~~
	45,3125	44,7812	202,4952
8	44,5212	~~44,6718~~	~~202,4892~~
	45,3125	45,1718	**202,4970**
9	**44,9218**		
	45,3129		

Tabela 8.2. Tabela com a solução do método da Dicótoma.

Para o problema anteriormente a função $f(x) = 9x - 0,1x^2$, tem como condição suficiente e necessária a transformação do valor porcentual da tolerância em um valor numérico relativo ao intervalo dado.

$$\varepsilon_I = 0,5\% \; de \; [0,100] \rightarrow \frac{0,5}{100} \cdot (100 - 0) = 0,5 \tag{8.6}$$

De forma subsequente o cálculo do valor infinitesimal (dx) que será utilizado como diferença intervalar será calculado da seguinte forma:

$$dx = \frac{\varepsilon_I}{2} = \frac{0,5}{2} = 0,25 \tag{8.7}$$

E para se ter uma previsão do número de etapas substitui-se na fórmula a seguir o valor numérico da tolerância:

$$n = -\frac{\log \varepsilon}{\log 2} = -\frac{\log 0,005}{\log 2} = 7,6400 \cong 8 \ etapas \tag{8.8}$$

O critério de parada é o mesmo dado pelo método anterior para o cálculo da discrepância:

$$I_c = 0,3907 < 0,5000 \tag{8.9}$$

Solução: O **método da Dicótoma** converge para um valor de máximo de $202,4970$ para o intervalo $[44,9218, 45,3129]$, sendo consideradas 4 casas decimais e tolerância de $0,5\%$ do intervalo dado. Os valores obtidos são distintos, devido à busca ser fundamentada no cálculo de um valor diferencial. Da mesma forma, o erro associado será minimizado a partir do momento que seja incrementado o número de casas decimais válidas e a tolerância inicial. Na tabela anterior temos as áreas sublinhadas como as áreas eliminadas. Os quadros finais representam os valores intervalados do ótimo e do máximo para $f(x)$.

8.1.3 Fibonacci

O **método de Fibonacci** é um método numérico bastante eficiente para encontrar um valor unimodal e univariável baseado na redução de intervalos, por meio da especificação da escolha de dois pontos interiores em cada etapa tendo como referência a série de Fibonacci. A partir da característica da série de Fibonacci ficou evidente que o método seria mais eficiente se escolhêssemos os pontos delimitantes os próprios números da série de modo que fossem reutilizados durante as iterações subsequentes. Dessa forma, apenas uma nova avaliação de $f(x)$ seria necessária em cada iteração até atingir o critério de parada. Logo, definiu-se como uma estratégia ótima aquela que produz a redução máxima no intervalo de incerteza de um determinado intervalo para o estudo de uma função unimodal e univariável. E basicamente o método de Fibonacci possui 3 passos, enunciados a seguir:

Passo 1: Cálculo Fibonacci Inicial	Passo 2: Ajuste da Tolerância	Passo 3: Posicionamento
$F_i = \{1,1,2,3,5,8,\ldots\}$ $F_o = \dfrac{b-a}{\varepsilon_I}$	$\varepsilon_I \rightarrow \varepsilon_I^*$ $\varepsilon_I^* = \dfrac{b-a}{F_o}$	$P = F_{i-1} \cdot \varepsilon_I^*$ $\begin{cases} x_1 = b - P \\ x_2 = a + P \end{cases}$

A seguir apresentamos a resolução numérica do método e a descrição de cada passo de forma explicativa:

P	I	x_i	$f(x_i)$
61,7900	0	38,2000	197,8700
	100	~~61,7900~~	~~174,3000~~
38,1800	0	~~23,6000~~	~~156,7900~~
	61,7900	38,1900	197,8600
23,6000	23,6000	~~38,1900~~	~~197,8600~~
	61,7900	47,2000	202,0100
14,5800	38,1900	47,2000	202,0200
	61,7900	~~52,7700~~	~~196,4600~~
9,0100	38,1900	43,7500	202,4200
	52,7700	~~47,2000~~	~~202,0100~~
5,5700	38,1900	~~41,6200~~	~~201,3500~~
	47,2000	43,7600	202,3400
3,4300	41,6200	~~43,7600~~	~~202,3400~~
	47,2000	45,0500	202,4900
2,1400	43,7600	~~45,0500~~	~~202,4900~~
	47,2000	45,9000	202,4100
1,2800	45,0500	~~44,6100~~	~~202,4800~~
	47,2000	45,0400	202,4900
0,8500	44,6100	45,0400	202,5100
	47,2000	~~45,4600~~	~~202,4700~~
0,4200	**44,6100**	45,0300	**202,4999**
	45,4600	~~45,0390~~	~~202,4998~~

Tabela 8.3. Tabela com a solução do método de Fibonacci.

Para o problema dado anteriormente, a função $f(x) = 9x - 0{,}1x^2$, faz–se necessário primeiro calcular o valor do Fibonacci inicial (F_o),

$$F_{13} = \{1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233\} \tag{8.10}$$

$$F_o = \frac{(b-a)}{\varepsilon_I} = \frac{(100-0)}{0{,}5} = 233 \tag{8.11}$$

De forma subsequente corrigir o valor da tolerância adaptando o valor dado à série de Fibonacci, sendo calculado da seguinte forma:

$$\varepsilon_I^{\,*} = \frac{(b-a)}{F_o} = \frac{(100-0)}{233} = 0{,}4291 \tag{8.12}$$

E a partir dos cálculos anteriores posicionam-se os valores dados em função da série de Fibonacci. Por exemplo, na primeira iteração teremos o seguinte posicionamento:

$$P = F_{i-1} \cdot \varepsilon_I^{\,*} \tag{8.13}$$

Solução: Similar aos métodos anteriores, a tabela anterior representa a resolução do **método de Fibonacci** convergindo para um valor de máximo de $202{,}4999$ para o intervalo $[44{,}6100,\ 45{,}4600]$, sendo consideradas 4 casas decimais e tolerância de $0{,}5\%$ do intervalo dado. Como esse método também tem como base a eliminação de região, só que em função da série de Fibonacci, temos nas áreas sublinhadas, as áreas eliminadas. Os últimos quadros representam os valores intervalados do ótimo e o máximo para $f(x)$.

8.1.4 Secção Áurea

O **método da Secção Áurea** consiste em uma forma bastante eficiente de eliminação de região. Dispõe de 2 pontos calculados a partir do número *b*=0,6180 e seu complementar para a unidade. Calculamos os posicionamentos dados por um *dx* em função do número áureo até satisfazer o critério de parada. E basicamente o método da Secção Áurea possui 3 passos, enunciados a seguir:

Passo 1: Cálculo número de etapas	Passo 2: Cálculo dx	Passo 3: Posicionamento
$n = 1 + \frac{\ln \varepsilon}{\ln 0,6180}$	$dx = 0,6180 \cdot (b - a)$	$\begin{cases} x_1 = x_i - dx \\ x_2 = x_f + dx \end{cases}$

A seguir mostramos a tabela com a resolução numérica do método e a descrição de cada passo de forma explicativa:

dx	I	x_i	$f(x_i)$
61,8000	0	38,2000	197,8760
	100	~~61,8000~~	~~174,2760~~
38,1924	0	~~23,6076~~	~~156,7365~~
	61,8000	38,1924	197,8657
23,6029	23,6076	~~38,1971~~	~~197,8721~~
	61,8000	47,2105	202,0114
14,5865	38,1971	47,2134	202,0101
	61,8000	~~52,7837~~	~~196,4414~~
9,01451	38,1971	43,7692	202,3485
	52,7837	~~47,2116~~	~~202,0109~~
5,5709	38,1971	~~41,6406~~	~~201,3715~~
	47,2116	43,7681	202,3482
3,4428	41,6406	~~43,7688~~	~~202,3484~~
	47,2116	45,0835	202,4993
2,1277	43,7688	45,0839	202,4993
	47,2116	~~45,8964~~	~~202,4196~~
1,3149	43,7688	~~44,5815~~	~~202,4825~~
	45,8964	45,0837	202,4993
0,8126	44,5815	45,0838	202,4993
	45,8964	~~45,3941~~	~~202,4845~~
0,5022	**44,5815**	~~44,8919~~	~~202,4988~~
	45,3941	45,0837	**202,4993**

Tabela 8.4. Tabela com a solução do método da Secção Áurea.

Para o problema dado anteriormente, a função $f(x) = 9x - 0{,}1x^2$, faz-se necessário primeiro o cálculo do número de etapas:

$$n = 1 + \frac{\ln 0{,}005}{\ln 0{,}6180} \cong 13 \ etapas \tag{8.14}$$

De forma subsequente, calcula-se o valor de *dx* junto ao posicionamento para o intervalo inicial. Por exemplo, para a primeira iteração teremos o valor de $dx = 0{,}6180 \cdot (100 - 0) = 61{,}8000$ para o posicionamento:

$$\begin{cases} x_1 = 100 - 61{,}80 = 38{,}2000 \\ x_2 = 0 + 61{,}80 = 61{,}8000 \end{cases} \tag{8.15}$$

O critério de parada está marcado pela comparação do valor do decréscimo no cálculo de posicionamento ($dx < \varepsilon_1$).

Solução: Similar aos métodos anteriores a tabela representa a resolução do **método da Secção Áurea** convergindo para um valor de máximo de $202{,}4993$ para o intervalo $[44{,}5815,\ 45{,}3941]$, sendo consideradas 4 casas decimais e tolerância de $0{,}5\%$ do intervalo dado. Como esse método também tem como base a eliminação de região, só que em função de um diferencial (dx), temos nas áreas sublinhadas, as áreas eliminadas. Os quadros em negrito representam os valores intervalados do ótimo, e o valor máximo para a $f(x)$.

8.2 Exercícios Propostos

8.2.1 Exercícios Gerais

Exercício 8.2.1.1

Encontre o mínimo da função $f(x) = (2x - 9)^2$ no intervalo de 0 a 10 com uma tolerância de $0{,}5\%$ do intervalo inicial. Utilizando o **método dos 5 Pontos** (considere 4 casas decimais válidas).

Exercício 8.2.1.2

Encontre o mínimo da função $f(x) = x^2 + 2x + 3$ no intervalo de -10 a 10 com uma tolerância de $0,2\%$ do intervalo inicial. Utilizando os métodos a seguir (considere 4 casas decimais válidas):

a) 5 pontos

b) Dicótoma

c) Fibonacci

d) Secção Áurea

8.2.2 Exercícios Aplicados

Exercício 8.2.2.1 - Aplicado à Engenharia Civil (Cálculo da Deflexão Máxima em Vigas)

A situação problemaesume-se em encontrar a deflexão máxima em uma ponte da via ferroviária, contendo 12 metros de extensão, sobre a qual, por acidente, um trem carregado pesando 57 toneladas, distribuídas pelos seus 2 vagões e pela sua locomotiva, parou de funcionar. Sabe-se que a deflexão máxima de uma viga, ocorre em um ponto em que a inclinação *theta* em relação ao eixo horizontal é nula e que a equação do momento fletor é dada pela função unimodal e univariável $f(x) = 5x^2 - 15x + 4$ (considere 6 casas decimais válidas e uma tolerância de $0,001\%$ do intervalo inicial).

SOLUÇÕES DOS EXERCÍCIOS GERAIS

CAPÍTULO 2. Resolução Numérica de Sistemas de Equações Lineares

Exercício 2.3.1.1

Resolução:

```
import math
import numpy as np

def Sassenfeld(A):
        coefficients = []
        for i in range(len(A)):
                b = 0
                for j in range(len(A)):
                        if(i!=j and i ==0) or i < j:
                          b +=A[i][j]
                        elif i!=j and i!=0:
                           b += A[i][j]*coefficients[j]
                b/=A[i][i]
                coefficients.append(b)

        majorCoefficient = max(coefficients)

        if majorCoefficient < 1:
          print('O sistema converge!')
        else:
          print('O sistema não converge pelo critério de Sassenfeld!')


A = np.array ([ [2,   1,   3 ],
                [0,  -1,   1 ],
                [1,   0,   3] ])
```

Sassenfeld(A)

Respostas: o sistema de equações lineares não converge, pois $\beta_1 = 2 > 1$.

Exercício 2.3.1.2

Resolução:

1. Jacobi

```
import numpy as np

x1 = 0
x2 = 0
x3 = 0
k  = 0
epsilon = 10**(-2)
converged = False

x_old = np.array([x1, x2, x3])

print('Método de Gass Jacobi')
print('k     x1     x2     x3 ')
print("%d  %.4f  %.4f  %.4f"%(k, x1, x2, x3))

for k in range(1, 1000):
    x1_new = (11 - 2*x2 - x3)/4
    x2_new =  (3 + x1)/2
    x3_new = (16 - 2*x1 - x2)/4

    x = np.array([x1_new, x2_new, x3_new])
    dx = np.sqrt(np.dot(x-x_old, x-x_old))

    print("%d  %.4f  %.4f  %.4f"%(k, x1_new, x2_new, x3_new))

    x1 = x1_new
    x2 = x2_new
    x3 = x3_new
```

```
    if dx < epsilon:
        converged = True
        print('Converge!')
        break

    x_old = x

if not converged:
   print('Não converge!')
```

Resposta: $\rightarrow x_7 = \left(0,9978,\ 1,9927,\ 3,0034\right)^t$

2. Gauss-Siedel

```
x1 = 0
x2 = 0
x3 = 0
k  = 0
epsilon = 10**(-2)
converged = False

x_old = np.array([x1, x2, x3])

print('Método de Gass Jacobi')
print('k     x1     x2     x3 ')
print("%d  %.4f  %.4f  %.4f"%(k, x1, x2, x3))

for k in range(1, 1000):
    x1_new = (11 - 2*x2 - x3)/4
    x2_new =  (3 + x1)/2
    x3_new = (16 - 2*x1 - x2)/4

    x = np.array([x1_new, x2_new, x3_new])
    dx = np.sqrt(np.dot(x-x_old, x-x_old))

    print("%d  %.4f  %.4f  %.4f"%(k, x1_new, x2_new, x3_new))
```

```
    x1 = x1_new
    x2 = x2_new
    x3 = x3_new

    if dx < epsilon:
        converged = True
        print('Converge!')
        break

    x_old = x

if not converged:
  print('Não converge!')
```

Resposta: $\rightarrow x_4 = \left(0,9985,\ 1,9992,\ 3,0009\right)^t$

3. Fatoração LU

```
import pprint
import numpy as np
import scipy.linalg as linalg

A = np.array([ [ 4,   2,  1 ],
               [-1,   2,  0 ],
               [ 2,   1,  4] ])

B = np.array([11, 3, 16])

LU = linalg.lu_factor(A)

x = linalg.lu_solve(LU, B)

print (x)
```

Resposta: $\rightarrow x = \left(1,\ 2,\ 3\right)^t$

Exercício 2.3.1.3

Resolução:

```
import pprint
import numpy as np
import scipy.linalg as linalg

A = np.array([ [ 3,   2,  4 ],
               [ 1,   1,  2 ],
               [ 4,   3,  -2] ])

B = np.array([1, 2, 3])

LU = linalg.lu_factor(A)

x = linalg.lu_solve(LU, B)

print (x)
```

Resposta: $\rightarrow x = \left(0,\ 5,\ -3\right)^{t}$

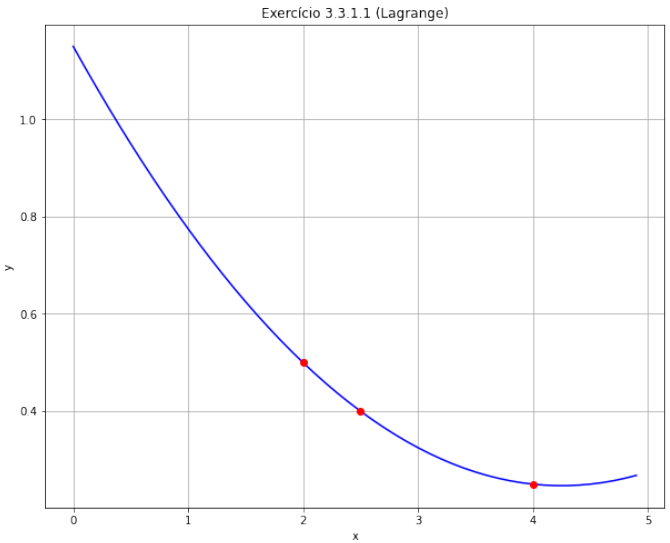
CAPÍTULO 3. Interpolação Numérica

Exercício 3.3.1.1

Resolução:

```
import numpy as np
from numpy.polynomial.polynomial import Polynomial
import pylab as plt
from scipy.interpolate import lagrange

x = [2, 2.5, 4]
y = [0.5, 0.4, 0.25]

f = lagrange(x, y)
```

```
print(Polynomial(f).coef)

x_new = np.arange(0, 5, 0.1)

fig = plt.figure(figsize = (10,8))
plt.plot(x_new, f(x_new), 'b', x, y, 'ro')
plt.title('Exercício 3.3.1.1 (Lagrange)')
plt.grid()
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



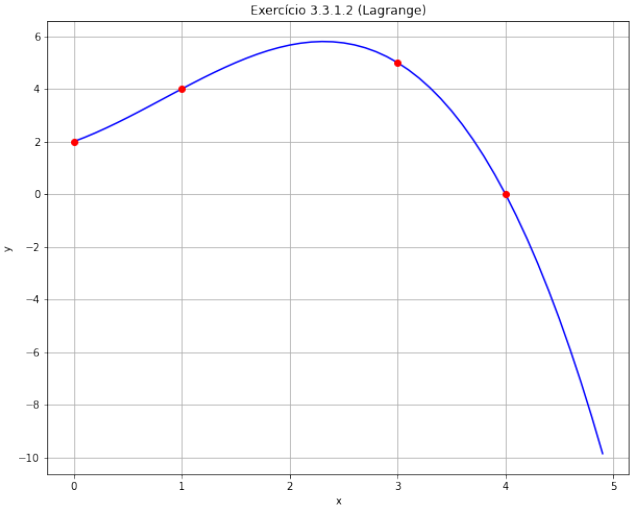
Resposta: $P(x) = 0,05x^2 - 0,425x + 1,15$

Exercício 3.3.1.2

Resolução:

```
import numpy as np
from numpy.polynomial.polynomial import Polynomial
import pylab as plt
from scipy.interpolate import lagrange

x = [0, 1, 3, 4]
```

```
y = [2, 4, 5, 0]

f = lagrange(x, y)

print(Polynomial(f).coef)

x_new = np.arange(0, 5, 0.1)

fig = plt.figure(figsize = (10,8))
plt.plot(x_new, f(x_new), 'b', x, y, 'ro')
plt.title('Exercício 3.3.1.2 (Lagrange)')
plt.grid()
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



Resposta:

$$P(x) = \frac{(x-1)(x-3)(x-4)}{-6} + \frac{2x(x-3)(x-4)}{3} + \frac{5x(x-1)(x-4)}{-6}$$

Exercício 3.3.1.3

Resolução:

```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

def newton_interpolation(x, y, xi):
    n = len(x)
    fdd = [[None for x in range(n)] for x in range(n)]
    yint = [None for x in range(n)]
    ea = [None for x in range(n)]

    for i in range(n):
        fdd[i][0] = y[i]
    for j in range(1,n):
        for i in range(n-j):
            fdd[i][j] = (fdd[i+1][j-1] - fdd[i][j-1])/(x[i+j]-x[i])

    fdd_table = pd.DataFrame(fdd)
    print(fdd_table)

    xterm = 1
    yint[0] = fdd[0][0]
    for order in range(1, n):
        xterm = xterm * (xi - x[order-1])
        yint2 = yint[order-1] + fdd[0][order]*xterm
        ea[order-1] = yint2 - yint[order-1]
        yint[order] = yint2

    return map(lambda yy, ee : [yy, ee], yint, ea)

x = [-1, 0, 2]
y = [4, 1, -1]

a  = newton_interpolation(x, y, 2)
df = pd.DataFrame(a)
```

Resposta: $P(x) = \frac{2}{3}x^2 - \frac{7}{3}x + 1 \rightarrow P(0,05) = 0,8850$

CAPÍTULO 4. Zeros da Função

Exercício 4.3.1.1 (Bisseção)

Resolução:

```
import numpy as np
from scipy import optimize
from numpy import log as ln

def f_a(x):
    return (x*np.log10(x)-1)

def f_b(x):
    return (x**2 + ln(x))

def f_c(x):
    return (x**3 - x - 1)

def f_d(x):
    return (x**2 - 3)

def f_e(x):
    return (x**3 - 10)
```

Resposta:

a) $x = 2{,}5078$ (7 etapas)
b) $x = 0{,}6528$ (10 etapas)
c) $x = 1{,}3281$ (6 etapas)
d) $x = 1{,}7343$ (6 etapas)
e) $x = 2{,}1562$ (5 etapas)

Exercício 4.3.1.2 (Newthon-Raphson)

Resolução:

```
import numpy as np
from scipy import optimize

def f_a(x):
    return (x**3+2*x-1)

def f_b(x):
    return (x*np.exp(x)-1)

def f_c(x):
    return (x**2*(x**2-1))

def f_d(x):
    return (2**x - x**2)
```

Resposta:

a) $x = 0{,}453398$ (2 etapas)
b) $x = 0{,}567230$ (3 etapas)
c) $x = 1{,}000308$ (4 etapas)
d) $x = -0{,}766667$ (2 etapas)

Exercício 4.3.1.3 (Secante)

Resolução:

```
import numpy as np
import scipy as sp
from scipy import optimize

def f_a(x):
    return(x**3-2*x**2-5)

def f_b(x):
    return(x-np.cos(x))

def f_c(x):
    return(np.sqrt(x)-5*np.exp(-x))

def f_d(x):
    return(x**3-0.5)
```

Resposta:

a) $x = 2{,}690695$ (7 etapas)

b) $x = 0{,}739119$ (3 etapas)

c) $x = 1{,}430419$ (2 etapas)

d) $x = 0{,}793479$ (6 etapas)

CAPÍTULO 5. Integração Numérica

Exercício 5.3.1.1

Resolução:

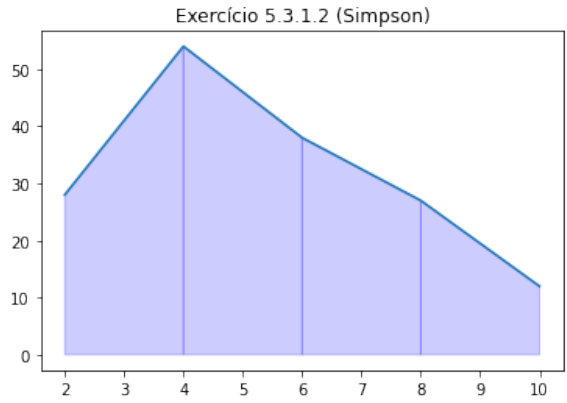
```
import sympy as sp
sp.init_printing()

x = sp.symbols('x')
```

```
fx = 3*x*x*x - 3*x + 1

I = sp.integrate(fx, (x,0,2))

I
```

Resposta: $I_S = 8$

Exercício 5.3.1.2

```
import scipy.integrate as spi
import numpy as np
import matplotlib.pyplot as plt

N = 5 - 1
a = 2
b = 10

x = [2, 4, 6, 8, 10]
y = [28, 54, 38, 27, 12]

I_s = spi.simps(y, x)
print("Simpson Integral: ", I_s)

plt.plot(x,y)

for i in range(N):
    xs = [x[i], x[i], x[i+1], x[i+1]]
    ys = [0, y[i], y[i+1], 0]
    plt.fill(xs, ys,'b',edgecolor='b',alpha=0.2)

plt.title('Exercício 5.3.1.2 (Simpson)')
plt.show()
```



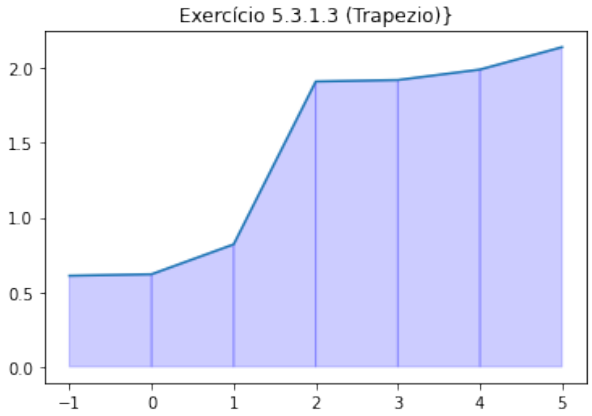
Resposta: I_S = **293,33**

Exercício 5.3.1.3

Resolução:

```
import numpy as np
import matplotlib.pyplot as plt

def trapz(x, y, a, b, N):
   y_right = y[1:]
   y_left = y[:-1]
   dx = (b - a)/N
   T = (dx/2) * np.sum(y_right + y_left)
   return T

a = -1
b = 5
N = 7 - 1

x = [-1, 0, 1, 2, 3, 4, 5]
y = [.61, .62, .82, 1.91, 1.92, 1.99, 2.14]

T = trapz(x, y, a, b, N)
print("Trapezio Integral: ", T)
```

```
#Plot
plt.plot(x,y)

for i in range(N):
   xs = [x[i], x[i], x[i+1], x[i+1]]
   ys = [0, y[i], y[i+1], 0]
   plt.fill(xs, ys,'b',edgecolor='b',alpha=0.2)

plt.title('Exercício 5.3.1.3 (Trapezio)}')
plt.show()
```

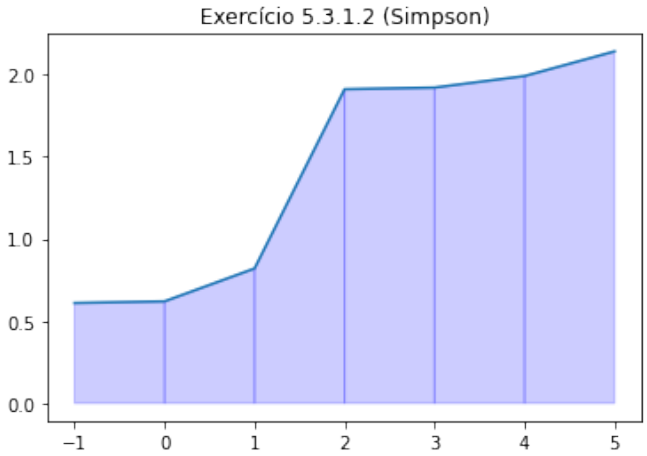


Resposta: $I_S = 8{,}7699$

Resolução:

```
import scipy.integrate as spi
import numpy as np
import matplotlib.pyplot as plt

a = -1
b = 5
N = 7 - 1

x = [-1, 0, 1, 2, 3, 4, 5]
y = [.61, .62, .82, 1.91, 1.92, 1.99, 2.14]
```

```
I_s = spi.simps(y, x)
print("Simpson Integral: ", I_s)

plt.plot(x,y)

for i in range(N):
    xs = [x[i], x[i], x[i+1], x[i+1]]
    ys = [0, y[i], y[i+1], 0]
    plt.fill(xs, ys,'b',edgecolor='b',alpha=0.2)

plt.title('Exercício 5.3.1.2 (Simpson)')
plt.show()
```



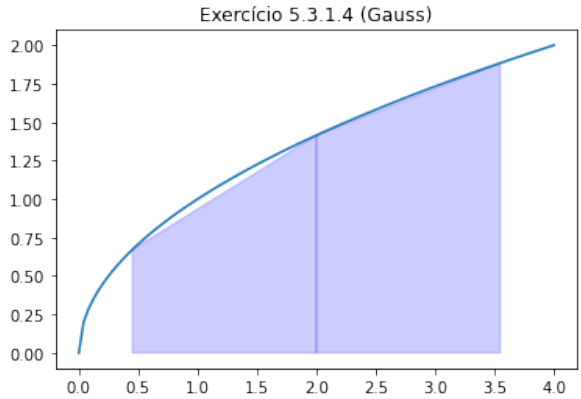
Resposta: $I_T = 8{,}6350$

Exercício 5.3.1.4

Resolução:

```
import numpy as np
import matplotlib.pyplot as plt

def gaussLegendre(f, a, b, E, A):
  #Plot
  N = 2
```

```
    X = np.linspace(a, b, 100)
    Y = f(X)
    plt.plot(X,Y)

    x = np.zeros(3)
    for i in range(3):
        x[i] = (b+a)/2 + (b-a)/2 * t[i]

    for i in range(N):
        xs = [x[i], x[i], x[i+1], x[i+1]]
        ys = [0, f(x[i]), f(x[i+1]), 0]
        plt.fill(xs, ys,'b',edgecolor='b',alpha=0.2)

    plt.title('Exercício 5.3.1.4 (Gauss)')
    plt.show()

    return (b-a)/2 * (P[0]*f(x[0]) + P[1]*f(x[1]) + P[2]*f(x[2]))

# coefficients for 2 subintervals

P = np.array([0.555556, 0.888889, 0.555556])  #Po = 5/9            P1 =
8/9    P2 = 5/9
t = np.array([-0.774597, 0.000000, 0.774597]) #t0 = -(3/5)**(-1/2) t1 = 0
t2 = +(3/5)**(-1/2)

f = lambda x: x**(1/2)
a = 0
b = 4

print("Gauss Integral: ", gaussLegendre(f, a, b, t, P))
```



Resposta: $I_G = 5{,}6077$

CAPÍTULO 6. Resolução Numérica de Sistemas de Equações Não Lineares

Exercício 6.1

a) Resolução:

```
import numpy as np

def jacobian_exercise_6_2_1_1_a(xy):
    x, y = xy
    return [[1,    2],
            [6*x,  1]]

def function_exercise_6_2_1_1_a(xy):
    x, y = xy
    return [x+2*y-3, 3*x**2+y-7]


def newtonRapshonModified(f, jacobian, x_init):
    max_iter = 50
    epsilon = 0.1
```

```
    x_last = x_init

  for k in range(max_iter):
      # Solve J(xn)*( xn+1 - xn ) = -F(xn)
      J = np.array(jacobian(x_last))
      F = np.array(f(x_last))

      diff = np.linalg.solve( J, -F )
      x_last = x_last + diff

      # Stop condition
      if np.linalg.norm(diff) < epsilon:
          print('Iterações =', k+1 )
          break

      else:
            print('Não converge!')

      return x_last

print('S =', newtonRapshonModified(function_exercise_6_2_1_1_a, jaco-
bian_exercise_6_2_1_1_a, [0.5,0.5]))
```

Resposta: $S = (1{,}4400,\ 0{,}7799)$

b) Resolução:

```
import numpy as np

def jacobian_exercise_6_2_1_1_b(xy):
    x, y = xy
    return [[2*x,  2*y],
            [2*x,  -2*y]]

def function_exercise_6_2_1_1_b(xy):
    x, y = xy
    return [x**2+y**2-2, x**2-y**2-1]
```

```
def newtonRapshonModified(f, jacobian, x_init):
    max_iter = 50
    epsilon = 0.1

    x_last = x_init

    for k in range(max_iter):
        # Solve J(xn)*( xn+1 - xn ) = -F(xn)
        J = np.array(jacobian(x_last))
        F = np.array(f(x_last))

        diff = np.linalg.solve( J, -F )
        x_last = x_last + diff

        # Stop condition
        if np.linalg.norm(diff) < epsilon:
            print('Iterações =', k+1 )
            break

  else:
    print('Não converge!')

  return x_last

print('S =', newtonRapshonModified(function_exercise_6_2_1_1_b, jaco-
bian_exercise_6_2_1_1_b, [1.2,0.7]))
```

Resposta: $S = (1{,}2252, 0{,}7071)$

c) Resolução:

```
import numpy as np

def jacobian_exercise_6_2_1_1_c(xy):
    x, y = xy
    return [[-20*x,    10],
            [-1,        0]]
```

```
def function_exercise_6_2_1_1_c(xy):
    x, y = xy
    return [10*(y-x**2), 1-x]


def newtonRapshonModified(f, jacobian, x_init):
    max_iter = 50
      epsilon = 0.1

    x_last = x_init

    for k in range(max_iter):
        # Solve J(xn)*( xn+1 - xn ) = -F(xn)
        J = np.array(jacobian(x_last))
        F = np.array(f(x_last))

    diff = np.linalg.solve( J, -F )
    x_last = x_last + diff

     # Stop condition
     if np.linalg.norm(diff) < epsilon:
        print('Iterações =', k+1 )
        break

     else:
         print('Não converge!')

  return x_last

print('S =', newtonRapshonModified(function_exercise_6_2_1_1_c, jaco-
bian_exercise_6_2_1_1_c, [-1.2, 1]))
```

Resposta: $S = (1{,}0000,\ 1{,}0000)$

d) Resolução:

```
import numpy as np

def jacobian_exercise_6_2_1_1_d(xy):
    x, y = xy
    return [[2*x,              2*y],
            [np.exp(x-1),      3*y**2]]

def function_exercise_6_2_1_1_d(xy):
    x, y = xy
    return [x**2+y**2-2, np.exp(x-1)+y**3-2]

def newtonRapshonModified(f, jacobian, x_init):
    max_iter = 50
    epsilon = 0.1

    x_last = x_init

    for k in range(max_iter):
        # Solve J(xn)*( xn+1 - xn ) = -F(xn)
        J = np.array(jacobian(x_last))
        F = np.array(f(x_last))

        diff = np.linalg.solve( J, -F )
        x_last = x_last + diff

        # Stop condition
        if np.linalg.norm(diff) < epsilon:
            print('Iterações =', k+1 )
            break

  else:
    print('Não converge!')

  return x_last
```

print('S =', newtonRapshonModified(function_exercise_6_2_1_1_d, jacobian_exercise_6_2_1_1_d, [1.5, 2]))

Resposta: $S = (1{,}0000, 1{,}0000)$

CAPÍTULO 7. Resolução Numérica de Equações Diferenciais Ordinárias

Exercício 7.3.1.1

a) Euler

Resolução:

```
import numpy as np

def jacobian_exercise_6_2_1_1_a(xy):
    x, y = xy
    return [[1,    2],
            [6*x,  1]]

def function_exercise_6_2_1_1_a(xy):
     x, y = xy
     return [x+2*y-3, 3*x**2+y-7]


def newtonRapshonModified(f, jacobian, x_init):
     max_iter = 50
     epsilon = 0.1

     x_last = x_init

     for k in range(max_iter):
          # Solve J(xn)*( xn+1 - xn ) = -F(xn)
          J = np.array(jacobian(x_last))
          F = np.array(f(x_last))

          diff = np.linalg.solve( J, -F )
```

```
        x_last = x_last + diff

        # Stop condition
        if np.linalg.norm(diff) < epsilon:
            print('Iterações =', k+1 )
            break

    else:
        print('Não converge!')

    return x_last

print('S =', newtonRapshonModified(function_exercise_6_2_1_1_a, jacobian_exercise_6_2_1_1_a, [0.5,0.5]))
```

Resposta:

i	x_i	y_i
0	0	3,0000
1	0,5	4,7000
2	1,0	4,8925
3	1,5	4,5499
4	2,0	4,0516

b) Runger-Kutta

```
import numpy as np

def jacobian_exercise_6_2_1_1_b(xy):
    x, y = xy
    return [[2*x,  2*y],
            [2*x,  -2*y]]

def function_exercise_6_2_1_1_b(xy):
    x, y = xy
```

```
    return [x**2+y**2-2, x**2-y**2-1]


def newtonRapshonModified(f, jacobian, x_init):
    max_iter = 50
    epsilon = 0.1

    x_last = x_init

    for k in range(max_iter):
        # Solve J(xn)*( xn+1 - xn ) = -F(xn)
        J = np.array(jacobian(x_last))
        F = np.array(f(x_last))

        diff = np.linalg.solve( J, -F )
        x_last = x_last + diff

        # Stop condition
        if np.linalg.norm(diff) < epsilon:
            print('Iterações =', k+1 )
            break

  else:
        print('Não converge!')

  return x_last

print('S =', newtonRapshonModified(function_exercise_6_2_1_1_b, jaco-
bian_exercise_6_2_1_1_b, [1.2,0.7]))
```

Resposta:

i	x_i	y_i
0	0	3,0000
1	5	3,9275
2	1,0	4,1607
3	1,5	4,0338
4	2,0	3,7344

Exercício 7.3.1.2

Runger-Kutta

Resolução:

```
import sympy as sp
sp.init_printing()

x = sp.symbols('x')

y = sp.Function('y')

dy = sp.Derivative(y(x),x)

edo = dy + x*x*x -y(x)*x

sp.expand(sp.dsolve(edo))

#print(sp.dsolve(edo))

C1 = sp.symbols('C1')

x_solved = C1*sp.exp(x**2/2) + x**2 + 2
```

Resposta:

i	x_i	y_i
0	0	1,0000
1	0,6	1,1800
2	1,2	1,5463
3	1,8	1,1512

CAPÍTULO 8. Princípios de Otimização

Exercício 8.2.1.1

$$Mínimo = 0{,}0020,\ Intervalo = [4{,}4726,\ 4{,}5117]$$

Exercício 8.2.1.2

a) $Mínimo = 2{,}0000,\ Intervalo = [-1{,}0156,\ -0{,}9766]$

b) $Mínimo = 2{,}0000,\ Intervalo = [-1{,}0156,\ -0{,}9766]$

c) $Mínimo = 2{,}0003,\ Intervalo = [-1{,}0492,\ -0{,}9836]$

d) $Mínimo = 2{,}0000,\ Intervalo = [-1{,}0156,\ -0{,}9766]$

REFERÊNCIAS

[1] ACTON, Forman S. *Numerical Methods That Work*. Harper and Row, New York, 1970. Reprinted by Mathematical Association of America, Washington, D.C., with new preface and additional problems, 1990.

[2] ALTMAN, Yair. *Accelerating MATLAB Performance. 1101 Tips to Speed up MATLAB Programs*. 2015.

[3] APRAHAMIAN, Mary and HIGHAM, Nicholas J. Matrix inverse trigonometric and inverse hyperbolic functions: Theory and algorithms. 37(4):1453–1477, 2016.

[4] ASCHER, Uri M. and PETZOLD, Linda R. *Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations*. 1998.

[5] ATKINSON, Kendall E. *An Introduction to Numerical Analysis*. Second edition, 1989.

[6] BAI, Zhaojun; DEMMEL, James W.; DONGARRA, Jack J.; RUHE, Axel and VORST, Henk A. Van der (ed.). *Templates for the Solution of Algebraic Eigenvalue Problems: A Practical Guide*. 2000.

[7] BARRETT, Richard; BERRY, Michael; CHAN, Tony F.; DEMMEL, James; DONATO, June; DONGARRA, Jack; EIJKHOUT, Victor; POZO, Roldan; ROMINE, Charles; VORST, Henk Van der. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*. 1994.

[8] BENTLEY, Jon L. *Programming Pearls*. 1986.

[9] BENTLEY, Jon L. *More Programming Pearls: Confessions of a Coder*. 1988.

[10] BORLAND, David; TAYLOR II, Russell M. Rainbow color map (still) considered harmful. *IEEE Computer Graphics and Applications*, 27(2):14–17, March/April 2007.

[11] BORNEMANN, Folkmar; LAURIE, Dirk; WAGON, Stan; WALDVOGEL, Jörg. *The SIAM 100 Digit Challenge: A Study in High-Accuracy Numerical Computing*. 2004.

[12] K. E. BRENAN, S. L. CAMPBELL, and L. R. PETZOLD. *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*. 1996. Corrected republication of work first published in 1989 by North--Holland, New York.

[13] BUCHHEIM, A.. On the theory of matrices. 16:63–82, 1884.

[14] CHANG, Yen-Ching. N-Dimension Golden Section Search: Its Variants and Limitations. pages 1 – 6, 11 2009.

[15] CHAPRA, Steven C. and CANALE, Raymond. *Numerical Methods for Engineers*. McGraw-Hill, Inc., New York, NY, USA, 5 edition, 2006.

[16] CONN, Andrew R.; SCHEINBERG, Katya; VICENTE, Luis N. *Introduction to Derivative-Free Optimization*. 2009.

[17] COOLEY, James W.; TUKEY, John W. An algorithm for the machine calculation of complex Fourier series. 19(90):297–301, 1965.

[18] COOLEY, James W.. How the FFT gained acceptance. In Stephen G. Nash, editor, *A History of Scientific Computing*, pages 133–140, 1990.

[19] CORLESS, Robert M.; FILLION, Nicolas. *A Graduate Introduction to Numerical Methods From the Viewpoint of Backward Error Analysis*. 2013.

[20] CORLESS, Robert M. *Essential Maple 7: An Introduction for Scientific Programmers*. 2002.

[21] CRILLY, Tony. The appearance of set operators in Cayley's group theory. *Notices of the South African Mathematical Society*, 31:9–22, 2000.

[22] CUNHA, M. C. *Métodos Numéricos*. Editora da Unicamp, São Paulo, Brasil, 2 edition, 2000.

[23] DAHLQUIST, Germund; BJÖRCK, Åke. *Numerical Methods*. 1974. Translated by Ned Anderson.

[24] DAVIS, Harold T. *Introduction to Nonlinear Differential and Integral Equations*. 1962.

[25] DAVIS, Timothy A. An unsymmetric-pattern multifrontal method. 30(2):196–199, June 2004.

[26] DAVIS, Timothy A. *Direct Methods for Sparse Linear Systems*. 2006.

[27] DAVIS, Timothy A. Algorithm 930: FACTORIZE: An object-oriented linear system solver for MATLAB. 39(4):28:1–28:18, July 2013.

[28] DEADMAN, Edvin; HIGHAM, Nicholas J.. Testing matrix function algorithms using identities. 42(1):4:1–4:15, January 2016.

[29] DEMMEL, James W., DONGARRA, Jack, EIJKHOUT, Victor, FUENTES, Erika, PETITET, Antoine, VUDUC, Richard, WHALEY, R. Clint, YELLICK, Katherine. Self-adapting linear algebra algorithms and software. *Proc. IEEE*, 93(2):293–312, 2005.

[30] DEMMEL, James W. *Applied Numerical Linear Algebra*. 1997.

[31] DENNIS Jr., J. E.; SCHNABEL, Robert B. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Soc for Industrial & Applied Math, 1996.

[32] DIJKSTRA, Edsger W. On the cruelty of really teaching computing science. https://www.cs.utexas.edu/users/EWD/transcriptions/EWD10xx/EWD1036.html, December 1998.

[33] DINGLE, Nicholas J.; HIGHAM, Nicholas J.. Reducing the influence of tiny normwise relative errors on performance profiles. 39(4):24:1–24:11, 2013.

[34] DRISCOLL, Tobin A., HALE, Nicholas, and TREFETHEN, Lloyd N. *Chebfun Guide*. Pafnuty Publications, Oxford, 2014.

[35] DUFF, I. S.; ERISMAN, A. M.; REID, J. K. *Direct Methods for Sparse Matrices*. 1986.

[36] EWING, R.E. Society for Industrial, and Applied Mathematics. *The Mathematics of Reservoir Simulation*. Frontiers in Applied Mathematics. Society for Industrial and Applied Mathematics, 1983.

[37] FLORES, J. M. *Problemas resueltos de matemáticas para la edificaccón e otras ingenierías*. Editora Parainfo, Asturias, Espanha, 1 edition, 2011.

[38] FRANCO, N. B. and CANALE, Raymond. *Cálculo Numérico*. Pearson Prentice Hall, Rio de Janeiro, Brasil, 5 edition, 2007.

[39] GEAR, C. W. and SKEEL, R. D.. The development of ODE methods: A symbiosis between hardware and numerical analysis. In Stephen G. Nash, editor, *A History of Scientific Computing*, pages 88–105., 1990.

[40] GEMAN, Stuart. The spectral radius of large random matrices. *Ann. Probab.*, 14(4):1318–1328, 1986.

[41] GILBERT, John R., MOLER, Cleve B., and SCHREIBER, Robert S.. Sparse matrices in MATLAB: Design and implementation. 13(1):333–356, 1992.

[42] GILL, Philip E., MURRAY, Walter, and WRIGHT, Margaret H.. *Practical optimization*. Academic Press Inc., London, 1981.

[43] GOLDEN SECTION SEARCH. Technique for Unimodal Optimization. Available in: http://www.maplesoft.com/applications/.

[44] GOLUB, Gene H. and LOAN, Charles F. Van. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.

[45] GOLUB, Gene H. and LOAN, Charles F. Van. *Matrix Computations*. Fourth edition, 2013.

[46] GREENBAUM, Anne. *Iterative Methods for Solving Linear Systems*. 1997.

[47] GRIFFITHS, David F., DOLD, John W., and SILVESTER, David J.. *Essential Partial Differential Equations: Analytical and Computational Aspects*. 2015.

[48] GRIFFITHS, David F. and HIGHAM, Desmond J. *Numerical Methods for Ordinary Differential Equations*. 2010.

[49] GRINDROD, Peter. *Mathematical Underpinnings of Analytics. Theory and Applications*. 2015.

[50] HIGHAM, Nicholas J. Color spaces and digital imaging. In Nicholas J. Higham, Mark R. Dennis, Paul Glendinning, Paul A. Martin, Fadil Santosa, and Jared Tanner, editors, *The Princeton Companion to Applied Mathematics*, pages 808–813. 2015.

[51] HIGHAM, Nicholas J. Programming languages: An applied mathematics view. In Nicholas J. Higham, Mark R. Dennis, Paul Glendinning, Paul A. Martin, Fadil Santosa, and Jared Tanner, editors, *The Princeton Companion to Applied Mathematics*, pages 828–839. 2015.

[52] LASZLO, Michael and MUKHERJEE, Sumitra. Iterated local search for microaggregation. *J. Syst. Softw.*, 100(C):15–26, Feb 2015.

[53] MOLER, Cleve B. and MORRISON, Donald. Replacing square roots by Pythagorean sums. *IBM J. Res. Develop.*, 27(6):577–581, 1983.

[54] MOLER, Cleve B. Yet another look at the FFT. *The MathWorks Newsletter*, Spring 1992.

[55] MOLER, Cleve B. MATLAB's magical mystery tour. *The MathWorks Newsletter*, 7(1):8–9, 1993.

[56] MOLER, Cleve B. Objectively speaking. OOPS is not an apology. *MATLAB News and Notes*, pages 6–7, 1999.

[57] MURRAY, J. D. *Mathematical Biology I. An Introduction*. 2002.

[58] NEFTCI, Salih N. *An Introduction to the Mathematics of Financial Derivatives*. Second edition, 2000.

[59] NEIDINGER, Richard D. Introduction to automatic differentiation and MATLAB object-oriented programming. 52(3):545–563, 2010.

[60] NEWMAN, M. E. J., MOORE, C., and WATTS, D. J.. Mean-field solution of the small-world network model. *Physical Review Letters*, 84:3201–3204, 2000.

[61] QUARTERONI, A. and SALERI, F. *Cálculo Científico - Com MATLAB e Octave*. Springer-Verlag, São Paulo, Brasil, 3 edition, 2007.

[62] SHAMPINE, Lawrence F., KIERZENKA, Jacek A., and REICHELT, Mark W.. Solving boundary value problems for ordinary differential equations in MATLAB, 2000.

[63] SHAMPINE, Lawrence F. and REICHELT, Mark W. The MATLAB ODE suite. 18(1):1–22, 1997.

[64] SHAMPINE, Lawrence F. and THOMPSON, S.. Solving DDEs in MATLAB. 37:441–458, 2001.

[65] STEWART, G. W. *Matrix Algorithms. Volume I: Basic Decompositions*. 1998.

[66] STEWART, G. W. *Matrix Algorithms. Volume II: Eigensystems*. 2001.

[67] SUBASI, Murat, YILDIRIM, Necmettin, and YILDIZ, Bennyamin. An improvement on Fibonacci Search Method in Optimization Theory. *Applied Mathematics and Computation*, 147:893–901, 01 2004.