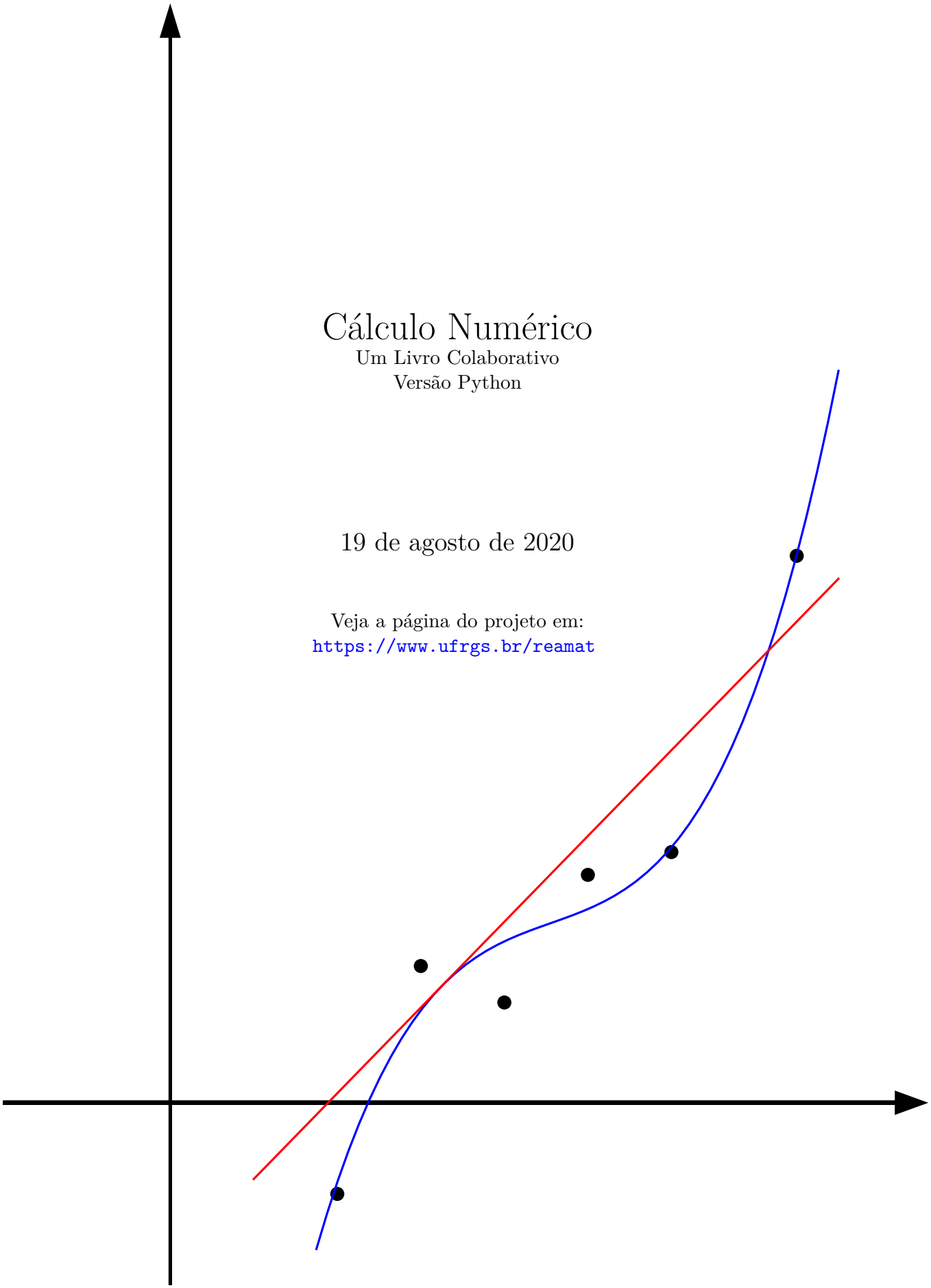
Cálculo Numérico

Um Livro Colaborativo

Versão Python

19 de agosto de 2020

Veja a página do projeto em:
https://www.ufrgs.br/reamat



Organizadores

Dagoberto Adriano Rizzotto Justo - UFRGS

Esequia Sauter - UFRGS

Fabio Souto de Azevedo - UFRGS

Leonardo Fernandes Guidi - UFRGS

Pedro Henrique de Almeida Konzen - UFRGS

Colaboradores

Este material é fruto da escrita colaborativa. Veja a lista de colaboradores em:

https://github.com/reamat/CalculoNumerico/graphs/contributors

Para saber mais como participar, visite o site oficial do projeto:

https://www.ufrgs.br/reamat/CalculoNumerico

ou comece agora mesmo visitando nosso repositório GitHub:

https://github.com/reamat/CalculoNumerico

Licença

Nota dos organizadores

Nosso objetivo é de fomentar o desenvolvimento de materiais didáticos pela colaboração entre professores e alunos de universidades, institutos de educação e demais interessados no estudo e aplicação de cálculo numérico nos mais diversos ramos da ciência e tecnologia.

Para tanto, disponibilizamos em repositório público GitHub todo o código-fonte dos materiais em desenvolvimento sob licença Creative Commons Atribuição-CompartilhaIgual 3.0 Não Adaptada (CC-BY-SA-3.0). Ou seja, você pode copiar, redistribuir, alterar e construir um novo material para qualquer uso, inclusive comercial. Leia a licença para maiores informações.

O sucesso do projeto depende da colaboração! Participe diretamenta da escrita dos recursos educacionais, dê sugestões ou nos avise de erros e imprecisões. Toda a colaboração é bem vinda. Veja mais sobre o projeto em:

https://www.ufrgs.br/reamat/CalculoNumerico

Desejamos-lhe ótimas colaborações!

Prefácio

Este livro busca abordar os tópicos de um curso de introdução ao cálculo numérico moderno oferecido a estudantes de matemática, física, engenharias e outros. A ênfase é colocada na formulação de problemas, implementação em computador da resolução e interpretação de resultados. Pressupõe-se que o estudante domine conhecimentos e habilidades típicas desenvolvidas em cursos de graduação de cálculo, álgebra linear e equações diferenciais. Conhecimentos prévios em linguagem de computadores é fortemente recomendável, embora apenas técnicas elementares de programação sejam realmente necessárias.

Nesta versão do livro, fazemos ênfase na utilização da linguagem computacional Python para a implementação dos métodos numéricos abordados. Recomendamos ao leitor ter à sua disposição um computador com o interpretador `Python 2.7` (ou superior) e o conjunto de biblioteca SciPy instalados. Não é necessário estar familiarizado com esta linguagem, mas recomendamos a leitura do Apêndice A, no qual apresentamos uma rápida introdução a esta linguagem com ênfase naquilo que é mais essencial para a leitura do livro. Alternativamente, existem algumas soluções em nuvem que fornecem acesso a consoles *online* Python. Veja, por exemplo, o CoCalc.

Os códigos computacionais dos métodos numéricos apresentados no livro são implementados em uma abordagem didática. Isto é, temos o objetivo de que a implementação em linguagem computacional venha a auxiliar o leitor no aprendizado das técnicas numéricas que são apresentadas no livro. Implementações computacionais eficientes de técnicas de cálculo numérico podem ser obtidas na série de livros "Numerical Recipes", veja [10].

Sumário

Capítulo 1

Introdução

Cálculo numérico é a disciplina que estuda as técnicas para a solução aproximada de problemas matemáticos. Estas técnicas são de natureza analítica e computacional. As principais preocupações normalmente envolvem exatidão e desempenho.

Aliado ao aumento contínuo da capacidade de computação disponível, o desenvolvimento de métodos numéricos tornou a simulação computacional de problemas matemáticos uma prática usual nas mais diversas áreas científicas e tecnológicas. As então chamadas simulações numéricas são constituídas de um arranjo de vários esquemas numéricos dedicados a resolver problemas específicos como, por exemplo: resolver equações algébricas, resolver sistemas de equações lineares, interpolar e ajustar pontos, calcular derivadas e integrais, resolver equações diferenciais ordinárias etc. Neste livro, abordamos o desenvolvimento, a implementação, a utilização e os aspectos teóricos de métodos numéricos para a resolução desses problemas.

Trabalharemos com problemas que abordam aspectos teóricos e de utilização dos métodos estudados, bem como com problemas de interesse na engenharia, na física e na matemática aplicada.

A necessidade de aplicar aproximações numéricas decorre do fato de que esses problemas podem se mostrar intratáveis se dispomos apenas de meios puramente analíticos, como aqueles estudados nos cursos de cálculo e álgebra linear. Por exemplo, o teorema de Abel-Ruffini nos garante que não existe uma fórmula algébrica, isto é, envolvendo apenas operações aritméticas e radicais, para calcular as raízes de uma equação polinomial de qualquer grau, mas apenas casos particulares:

- Simplesmente isolar a incógnita para encontrar a raiz de uma equação do primeiro grau;
- Fórmula de Bhaskara para encontrar raízes de uma equação do segundo grau;
- Fórmula de Cardano para encontrar raízes de uma equação do terceiro grau;

- Existe expressão para equações de quarto grau;
- Casos simplificados de equações de grau maior que 4 onde alguns coeficientes são nulos também podem ser resolvidos.

Equações não polinomiais podem ser ainda mais complicadas de resolver exatamente, por exemplo:

$$\cos(x) = x \qquad \text{ou} \qquad xe^x = 10 \tag{1.1}$$

Para resolver o problema de valor inicial

$$\begin{aligned} y' + xy &= x, \\ y(0) &= 2, \end{aligned} \tag{1.2}$$

podemos usar o método de fator integrante e obtemos $y = 1 + e^{-x^2/2}$. No entanto, não parece possível encontrar uma expressão fechada em termos de funções elementares para a solução exata do problema de valor inicial dado por

$$\begin{aligned} y' + xy &= e^{-y}, \\ y(0) &= 2, \end{aligned} \quad . \tag{1.3}$$

Da mesma forma, resolvemos a integral

$$\int_1^2 xe^{-x^2} dx \tag{1.4}$$

pelo método da substituição e obtemos $\frac{1}{2}(e^{-1} - e^{-2})$. Porém a integral

$$\int_1^2 e^{-x^2} dx \tag{1.5}$$

não pode ser expressa analiticamente em termos de funções elementares, como uma consequência do teorema de Liouville.

A maioria dos problemas envolvendo fenômenos reais produzem modelos matemáticos cuja solução analítica é difícil (ou impossível) de obter, mesmo quando provamos que a solução existe. Nesse curso propomos calcular aproximações numéricas para esses problemas, que apesar de, em geral, serem diferentes da solução exata, mostraremos que elas podem ser bem próximas.

Para entender a construção de aproximações é necessário estudar como funciona a aritmética implementada nos computadores e erros de arredondamento. Como computadores, em geral, usam uma base binária para representar números, começaremos falando em mudança de base.

Capítulo 2

Representação de números e aritmética de máquina

Neste capítulo, abordaremos formas de representar números reais em computadores. Iniciamos com uma discussão sobre representação posicional e mudança de base. Então, enfatizaremos a representação de números com quantidade finita de dígitos, mais especificamente, as representações de números inteiros, ponto fixo e ponto flutuante em computadores.

A representação de números e a aritmética em computadores levam aos chamados erros de arredondamento e de truncamento. Ao final deste capítulo, abordaremos os efeitos do erro de arredondamento na computação científica.

Ao longo do capítulo, faremos alguns comentários usando códigos em `Python 2.7`. Nestes, assumiremos que os seguintes módulos estão carregados:

```
>>> from __future__ import division
>>> import numpy as np
```

A primeira instrução garante que divisões de números inteiros sejam computadas em ponto flutuante (`double`) e a segunda carrega a biblioteca de computação científica numpy.

2.1 Sistema de numeração e mudança de base

Usualmente, utilizamos o sistema de numeração decimal para representar números. Esse é um sistema de numeração posicional onde a posição do dígito indica a potência de 10 que o dígito representa.

Exemplo 2.1.1. O número 293 é decomposto como

$$\begin{aligned} 293 &= 2 \text{ centenas} + 9 \text{ dezenas} + 3 \text{ unidades} \\ &= 2 \cdot 10^2 + 9 \cdot 10^1 + 3 \cdot 10^0. \end{aligned} \tag{2.1}$$

O sistema de numeração posicional também pode ser usado com outras bases. Vejamos a seguinte definição.

Definição 2.1.1 (Sistema de numeração de base b)**.** *Dado um número natural $b > 1$ e o conjunto de símbolos $\{\pm, \mathbf{0}, \mathbf{1}, \mathbf{2}, \ldots, \boldsymbol{b-1}\}$[1], a sequência de símbolos*

$$(d_n d_{n-1} \cdots d_1 d_0{,}d_{-1} d_{-2} \cdots)_b \tag{2.2}$$

representa o número positivo

$$d_n \cdot b^n + d_{n-1} \cdot b^{n-1} + \cdots + d_0 \cdot b^0 + d_{-1} \cdot b^{-1} + d_{-2} \cdot b^{-2} + \cdots \tag{2.3}$$

Para representar números negativos usamos o símbolo $-$ a esquerda do numeral[2]*.*

Observação 2.1.1 ($b \geq 10$)**.** Para sistemas de numeração com base $b \geq 10$ é usual utilizar as seguintes notações:

- No sistema de numeração decimal ($b = 10$), costumamos representar o número sem os parênteses e o subíndice, ou seja,

$$\pm\, d_n d_{n-1} \ldots d_1 d_0{,}d_{-1} d_{-2} \ldots := \pm(d_n d_{n-1} \ldots d_1 d_0{,}d_{-1} d_{-2} \ldots)_{10}. \tag{2.4}$$

- Se $b > 10$, usamos as letras $A, B, C, \cdots$ para denotar os algarismos: $A = 10$, $B = 11$, $C = 12$, $D = 13$, $E = 14$, $F = 15$.

Exemplo 2.1.2 (Sistema binário)**.** O sistema de numeração em base dois é chamado de binário e os algarismos binários são conhecidos como *bits* (do inglês **binary digits**). Um *bit* pode assumir dois valores distintos: 0 ou 1. Por exemplo:

$$\begin{aligned} x &= (1001{,}101)_2 \\ &= 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 1 \cdot 2^{-1} + 0 \cdot 2^{-2} + 1 \cdot 2^{-3} \\ &= 8 + 0 + 0 + 1 + 0{,}5 + 0 + 0{,}125 = 9{,}625. \end{aligned} \tag{2.5}$$

Ou seja, $(1001{,}101)_2$ é igual a 9,625 no sistema decimal.

Em `Python` podemos converter o número $(1001{,}101)_2$ para a base decimal computando

```
>>> 1*2**3 + 0*2**2 + 0*2**1 + 1*2**0 + 1*2**-1 + 0*2**-2 + 1*2**-3
9.625
```

[1] Para $b > 10$, veja a Observação 2.1.1.

[2] O uso do símbolo $+$ é opcional na representação de números positivos.

Exemplo 2.1.3 (Sistema quaternário)**.** No sistema quaternário a base b é igual a 4 e, portanto, temos o seguinte conjunto de algarismos $\{0, 1, 2, 3\}$. Por exemplo:

$$(301{,}2)_4 = 3 \cdot 4^2 + 0 \cdot 4^1 + 1 \cdot 4^0 + 2 \cdot 4^{-1} = 49{,}5. \tag{2.6}$$

Verifique no computador!

Exemplo 2.1.4 (Sistema octal)**.** No sistema octal a base é $b = 8$. Por exemplo:

$$\begin{aligned}(1357{,}24)_8 &= 1 \cdot 8^3 + 3 \cdot 8^2 + 5 \cdot 8^1 + 7 \cdot 8^0 + 2 \cdot 8^{-1} + 4 \cdot 8^{-2} \\ &= 512 + 192 + 40 + 7 + 0{,}25 + 0{,}0625 = 751{,}3125.\end{aligned} \tag{2.7}$$

Verifique no computador!

Exemplo 2.1.5 (Sistema hexadecimal)**.** O sistema de numeração cuja a base é $b = 16$ é chamado de sistema hexadecimal. Neste, temos o conjunto de algarismos $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F\}$. Convertendo o número $(E2AC)_{16}$ para a base 10 temos

$$\begin{aligned}(E2AC)_{16} &= 14 \cdot 16^3 + 2 \cdot 16^2 + 10 \cdot 16^1 + 12 \cdot 16^0 \\ &= 57344 + 512 + 160 + 12 = 58028.\end{aligned} \tag{2.8}$$

Verifique no computador!

Observação 2.1.2. `Python` tem algumas sintaxes para representar números em algumas bases. Por exemplo, temos:

```
>>> print(0b1001) #bin -> dec
9
>>> print(0o157) #oct -> dec
111
>>> print(0xbeba) #hex -> dec
48826
```

Nos exemplos acima vimos como converter números representados em um sistema de numeração de base b para o sistema decimal. Agora, vamos estudar como fazer o processo inverso. Isto é, dado um número decimal $(X)_{10}$ queremos escrevê-lo em uma outra base b, isto é, queremos obter a seguinte representação:

$$\begin{aligned}(X)_{10} &= (d_n d_{n-1} \cdots d_0{,}d_{-1} \cdots)_b \\ &= d_n \cdot b^n + d_{n-1} \cdot b^{n-1} + \cdots + d_0 \cdot b^0 + d_{-1} \cdot b^{-1} + d_{-2} \cdot b^{-2} + \cdots\end{aligned} \tag{2.9}$$

Separando as partes inteira e fracionária de X, isto é, $X = X^{\mathrm{i}} + X^{\mathrm{f}}$, temos

$$X^{\mathrm{i}} = d_n \cdot b^n + \cdots + d_{n-1} b^{n-1} \cdots + d_1 \cdot b^1 + d_0 \cdot b^0 \tag{2.10}$$

e

$$X^{\mathrm{f}} = \frac{d_{-1}}{b^1} + \frac{d_{-2}}{b^2} + \cdots \tag{2.11}$$

Nosso objetivo é determinar os algarismos $\{d_n, d_{n-1}, ...\}$.

Primeiramente, vejamos como tratar a parte inteira X^{i}. Calculando sua divisão de X^{i} por b, temos:

$$\frac{X^{\mathrm{i}}}{b} = \frac{d_0}{b} + d_1 + d_2 \cdot b^1 + \cdots + d_{n-1} \cdot b^{n-2} + d_n \cdot b^{n-1}. \tag{2.12}$$

Observe que d_0 é o resto da divisão de X^{i} por b, pois $d_1 + d_2 \cdot b^1 + \cdots + d_{n-1} \cdot b^{n-2} + d_n \cdot b^{n-1}$ é inteiro e $\frac{d_0}{b}$ é uma fração com $d_0 < b$. Da mesma forma, o resto da divisão de $d_1 + d_2 \cdot b^1 + \cdots + d_{n-1} \cdot b^{n-2} + d_n \cdot b^{n-1}$ por b é d_1. Ou seja, repetindo este processo encontramos os algarismos d_0, d_1, d_2, ..., d_n.

Vamos, agora, converter a parte fracionária X^{f} do número decimal X para o sistema de base b. Multiplicando X^{f} por b, temos

$$bX^{\mathrm{f}} = d_{-1} + \frac{d_{-2}}{b} + \frac{d_{-3}}{b^2} + \cdots \tag{2.13}$$

Observe que a parte inteira desse produto é d_{-1} e $\frac{d_{-2}}{b} + \frac{d_{-3}}{b^2} + \cdots$ é a parte fracionária. Quando multiplicamos $\frac{d_{-2}}{b} + \frac{d_{-3}}{b^2} + \cdots$ por b novamente, encontramos d_{-2}. Repetindo este processo encontramos os demais algarismos.

Exemplo 2.1.6. Vamos converter o número 9,625 para a base binária ($b = 2$). Primeiramente, decompomos 9,625 na soma de suas partes inteira e fracionária.

$$9{,}625 = 9 + 0{,}625. \tag{2.14}$$

Conversão da parte inteira. Para converter a parte inteira, fazemos sucessivas divisões por $b = 2$ obtendo

$$\begin{aligned} 9 &= 4 \cdot 2 + 1 && (2.15)\\ &= (2 \cdot 2 + 0) \cdot 2 + 1 && (2.16)\\ &= 2^3 + 1. && (2.17) \end{aligned}$$

Ou seja, temos que $9 = (1001)_2$. Em `Python`, podemos usar os comandos `int` (truncamento) e a operação `%` (resto da divisão) para computar esta conversão da seguinte forma

```
>>> x = 9
>>> d0 = x%2; x = int(x/2); print("d0 = %d, x = %d" % (d0,x))
d0 = 1, x = 4
```

```
>>> d1 = x%2; x = int(x/2); print("d1 = %d, x = %d" % (d1,x))
d1 = 0, x = 2
>>> d2 = x%2; x = int(x/2); print("d2 = %d, x = %d" % (d2,x))
d2 = 0, x = 1
>>> d3 = x%2; x = int(x/2); print("d3 = %d, x = %d" % (d3,x))
d3 = 1, x = 0
```

Conversão da parte fracionária. Para converter a parte fracionária, fazemos sucessivas multiplicações por $b = 2$ obtendo

$$0{,}625 = 1{,}25 \cdot 2^{-1} = 1 \cdot 2^{-1} + 0{,}25 \cdot 2^{-1} \tag{2.18}$$
$$= 1 \cdot 2^{-1} + (0{,}5 \cdot 2^{-1}) \cdot 2^{-1} = 1 \cdot 2^{-1} + 0{,}5 \cdot 2^{-2} \tag{2.19}$$
$$= 1 \cdot 2^{-1} + (1 \cdot 2^{-1}) \cdot 2^{-2} = 1 \cdot 2^{-1} + 1 \cdot 2^{-3} \tag{2.20}$$

Ou seja, temos que $0{,}625 = (0{,}101)_2$. Em `Python`, podemos computar esta conversão da parte fracionária da seguinte forma

```
>>> x=0.625
>>> d = int(2*x); x = 2*x - d; print("d = %d, x = %f" % (d,x))
d = 1, x = 0.250000
>>> d = int(2*x); x = 2*x - d; print("d = %d, x = %f" % (d,x))
d = 0, x = 0.500000
>>> d = int(2*x); x = 2*x - d; print("d = %d, x = %f" % (d,x))
d = 1, x = 0.000000
```

Conclusão. Da conversão das partes inteira e fracionária de 9,625, obtemos $9 = (1001)_2$ e $0{,}625 = (0{,}101)_2$. Logo, concluímos que $9{,}625 = (1001{,}101)_2$.

Observação 2.1.3. `Python` oferece algumas funções para a conversão de números inteiros em base decimal para uma base dada. Por exemplo, temos:

```
>>> print(bin(9))
0b1001
>>> print(oct(111)) #a saída será "0o157", no Python 3 ou superior
0157
>>> print(hex(48826))
0xbeba
```

Observação 2.1.4. Uma maneira de converter um número dado em uma base b_1 para uma base b_2 é fazer em duas partes: primeiro converter o número dado na base b_2 para base decimal e depois converter para a base b_1.

Exercícios resolvidos

Esta seção carece de exercícios resolvidos. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

ER 2.1.1. Obtenha a representação do número $125{,}58\overline{3}$ na base 6.

Solução. Decompomos $125{,}58\overline{3}$ nas suas partes inteira 125 e fracionária $0{,}58\overline{3}$. Então, convertemos cada parte.

Conversão da parte inteira. Vamos escrever o número 125 na base 6. Para tanto, fazemos sucessivas divisões por 6 como segue:

$$\begin{aligned} 125 &= 20 \cdot 6 + 5 \quad (125 \text{ dividido por } 6 \text{ é igual a } 20 \text{ e resta } 5) \\ &= (3 \cdot 6 + 2) \cdot 6 + 5 = 3 \cdot 6^2 + 2 \cdot 6 + 5, \end{aligned} \tag{2.21}$$

logo $125 = (325)_6$.

Estes cálculos podem ser feitos em `Python` com o auxílio do operador `%` e da função `int`. Com o primeiro calculamos o resto da divisão entre dois números, enquanto que a segunda retorna a parte inteira de um número dado. No nosso exemplo, temos:

```
>>> q = 125; d0 = (q % 6); print(q,d0)
>>> q = int(q/6); d1 = (q % 6); print(q,d1)
>>> q = int(q/6); d2 = (q % 6); print(q,d2)
```

Verifique!

Conversão da parte fracionária. Para converter $0{,}58\overline{3}$ para a base 6, fazemos sucessivas multiplicações por 6 como segue:

$$\begin{aligned} 0{,}58\overline{3} &= 3{,}5 \cdot 6^{-1} \quad (0{,}58\overline{3} \text{ multiplicado por } 6 \text{ é igual a } 3{,}5) \\ &= 3 \cdot 6^{-1} + 0{,}5 \cdot 6^{-1} \\ &= 3 \cdot 6^{-1} + (3 \cdot 6^{-1}) \cdot 6^{-1} \\ &= 3 \cdot 6^{-1} + 3 \cdot 6^{-2}, \end{aligned} \tag{2.22}$$

logo $0{,}58\overline{3} = (0{,}33)_6$. Em `Python`, é possível computar esta conversão de valores fracionários da seguinte maneira:

```
>>> x = 0.58 + 1/3/100; print("x = {}".format(x))
x = 0.5833333333333333
>>> d = int(6*x); x = round(6*x,1) - d; print("d = {}, x = {}".format(d, x))
d = 3, x = 0.5
>>> d = int(6*x); x = round(6*x,1) - d; print("d = {}, x = {}".format(d, x))
d = 3, x = 0.0
```

◊

ER 2.1.2. Obtenha a representação na base 4 do número $(101{,}01)_2$.

Solução. Começamos convertendo $(101{,}01)_2$ para a base decimal:

$$(1001{,}101)_2 = 1 \cdot 2^2 + 1 \cdot 2^0 + 1 \cdot 2^{-2} = 5{,}25. \tag{2.23}$$

Então, convertemos 5,25 para a base 4. Para sua parte inteira, temos

$$5 = 1 \cdot 4 + 1 = (11)_4. \tag{2.24}$$

Para sua parte fracionária, temos

$$0{,}25 = 1 \cdot 4^{-1} = (0{,}1)_4. \tag{2.25}$$

Logo, $(101{,}01)_2 = (11{,}1)_4$. Verifique estas contas no computador! ◊

Exercícios

E 2.1.1. Converta para base decimal cada um dos seguintes números:

a) $(100)_2$

b) $(100)_3$

c) $(100)_b$

d) $(12)_5$

e) $(AA)_{16}$

f) $(7{,}1)_8$

g) $(3{,}12)_5$

E 2.1.2. Escreva os números abaixo na base decimal.

a) $(25{,}13)_8$

b) $(101{,}1)_2$

c) $(12F{,}4)_{16}$

d) $(11{,}2)_3$

E 2.1.3. Escreva o número 5,5 em base binária.

E 2.1.4. Escreva o número 17,109375 em base hexadecimal ($b = 16$).

E 2.1.5. Escreva cada número decimal na base b.

a) $7{,}\overline{6}$ na base $b = 5$

b) $29{,}1\overline{6}$ na base $b = 6$

E 2.1.6. Escreva $(12.4)_8$ em base decimal e binária.

E 2.1.7. Escreva cada número dado para a base b.

a) $(45{,}1)_8$ para a base $b = 2$

b) $(21{,}2)_8$ para a base $b = 16$

c) $(1001{,}101)_2$ para a base $b = 8$

d) $(1001{,}101)_2$ para a base $b = 16$

E 2.1.8. Quantos algarismos são necessários para representar o número 937163832173947 em base binária? E em base 7? Dica: Qual é o menor e o maior inteiro que pode ser escrito em dada base com N algarismos?

2.2 Notação científica e notação normalizada

Como vimos, no sistema posicional usual um número x na base b é representado por

$$x = \pm(d_n d_{n-1} \cdots d_0{,}d_{-1}d_{-2}d_{-3}\cdots)_b, \tag{2.26}$$

onde $d_n \neq 0$ e $d_i \in \{0, 1, \ldots, b-1\}$ é o dígito da i-ésima posição. Alternativamente, é costumeiro usarmos a chamada notação científica. Nesta, o número x é representado como

$$x = \pm(M)_b \times b^e, \tag{2.27}$$

onde $(M)_b = (d_m d_{m-1} \cdots d_0{,}d_{-1}d_{-2}d_{-3}\cdots)_b$ é chamada de mantissa e $e \in \mathbb{Z}$ é chamado de expoente de x.

Exemplo 2.2.1. a) O número 602,2141 em notação científica pode ser escrito como

$$602{,}2141 \times 10^0 = 60{,}22141 \times 10^1 = 0{,}6022141 \times 10^3. \tag{2.28}$$

b) O número $(1010{,}10)_2$ pode ser escrito em notação científica como $(10{,}1010)_2 \times 2^2$.

Observamos que um número pode ser representado de várias formas equivalentes em notação científica. Para termos uma representação única introduzimos o conceito de notação normalizada.

Definição 2.2.1. *Um número x na base b é dito estar representado em notação (científica) normalizada quando está escrito na forma*

$$x = (-1)^s (M)_b \times b^E, \tag{2.29}$$

onde $(M)_b = (d_0{,}d_{-1}d_{-2}d_{-3}\cdots)_b$, com $d_0 \neq 0$[34]*, s é 0 para positivo e 1 para negativo, E é o expoente.*

Exemplo 2.2.2. Vejamos os seguintes casos:

a) O número 602,2141 em notação (científica) normalizada é representado por $6{,}022141 \times 10^2$.

b) O número $(1010{,}10)_2$ escrito em notação normalizada é $(1{,}01010)_2 \times 2^3$.

Observação 2.2.1. Em `Python`, podemos controlar a impressão de números usando o comando `print`. Por exemplo:

```
>>> print("%1.5f" % -np.pi)
-3.14159
>>> print("%1.5e" % -np.pi)
-3.14159e+00
```

No primeiro caso, obtemos a representação em ponto flutuante decimal com 6 dígitos do número $-\pi$. No segundo caso, obtemos a representação em notação científica normalizada com 6 dígitos.

Exercícios resolvidos

Esta seção carece de exercícios resolvidos. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

[3] Em algumas referências é usado $M_b = (0{,}d_{-1}d_{-2}d_{-3}\cdot)_b$.

[4] No caso de $x = 0$, $M_b = (0{,}00\cdots)_b$.

Exercícios

Esta seção carece de exercícios. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

E 2.2.1. Represente os seguintes números em notação científica normalizada:

$$\begin{array}{ll} a)\ 299792{,}458 & b)\ 66{,}2607 \times 10^{-35} \\ c)\ 0{,}6674 \times 10^{-7} & d)\ 9806{,}65 \times 10^{1} \end{array} \tag{2.30}$$

(2.31)

E 2.2.2. Use o computador para verificar as respostas do Exercício 2.2.1.

2.3 Representação decimal finita

Em computadores, é usual representarmos números usando uma quantidade de dígitos finita. A quantidade a ser usada normalmente depende da precisão com que as computações estão sendo feitas. Ocorre que quando restringimos a representação a um número finito de dígitos, muitos números não podem ser representado de forma exata, por exemplo, as dízimas infinitas e os números irracionais. Este fenômeno nos leva aos conceitos de número de dígitos significativos e de arredondamento.

Definição 2.3.1 (Número de dígitos significativos)**.** *Um número decimal* $x = \pm d_0{,}d_{-1}\cdots d_{-i}d_{-i-1}\cdots d_{-i-n}d_{-i-n-1}\cdot\times 10^E$ *é dito ter* n *dígitos significativos quando* $d_j = 0$ *para* $j \geq -i$ *e* $j \leq -i-n-1$.

Exemplo 2.3.1. O número $0{,}0602100 \times 10^{-3}$ tem 4 dígitos significativos.

2.3.1 Arredondamento de números

Quando representamos um número x com uma quantidade de dígitos menor que a de dígitos significativos acabamos com uma aproximação deste. Este procedimento é chamado arredondamento de um número. Mais precisamente, seja dado

$$x = \pm d_0{,}d_1d_2\ldots d_{k-1}d_kd_{k+1}\ldots d_n \times 10^e \tag{2.33}$$

em notação normalizada, isto é, $d_0 \neq 0$[5]. Podemos representar x com k dígitos da seguinte forma:

1. **Arredondamento por truncamento** (ou corte): aproximamos x por

$$\bar{x} = \pm d_0{,}d_1d_2\ldots d_k \times 10^e \tag{2.34}$$

simplesmente descartando os dígitos d_j com $j > k$.

2. **Arredondamento por proximidade**[6]: se $d_{k+1} < 5$ aproximamos x por

$$\bar{x} = \pm d_0{,}d_1d_2\ldots d_k \times 10^e \tag{2.35}$$

senão aproximamos x por[7]

$$\bar{x} = \pm(d_0{,}d_1d_2\ldots d_k + 10^{-k}) \times 10^e \tag{2.37}$$

3. **Arredondamento por proximidade com desempate par**: se $d_{k+1} < 5$ aproximamos x por

$$\bar{x} = \pm d_0{,}d_1d_2\ldots d_k \times 10^e. \tag{2.38}$$

Se $d_{k+1}{,}d_{d+2}d_{k+3}\ldots > 5$ aproximamos x por

$$\bar{x} = \pm(d_0{,}d_1d_2\ldots d_k + 10^{-k}) \times 10^e. \tag{2.39}$$

Agora, no caso de empate, i.e. $d_{k+1}{,}d_{k+2}d_{k+3}\ldots = 5$, então x é aproximado por

$$\bar{x} = \pm d_0{,}d_1d_2\ldots d_k \times 10^e \tag{2.40}$$

caso d_k seja par e, caso contrário, por

$$\bar{x} = \pm(d_0{,}d_1d_2\ldots d_k + 10^{-k}) \times 10^e. \tag{2.41}$$

Observação 2.3.1. O arredondamento por proximidade é frequentemente empregado para arredondamentos de números reais para inteiros. Por exemplo:

- $x = 1{,}49$ arredonda-se para o inteiro 1.

[5] caso $x \neq 0$.

[6] com desempate infinito.

[7] Note que essas duas opções são equivalentes a somar 5 no dígito a direita do corte e depois arredondar por corte, ou seja, arredondar por corte

$$\pm\left(d_0{,}d_1d_2\ldots d_kd_{k+1} + 5 \times 10^{-(k+1)}\right) \times 10^e \tag{2.36}$$

- $x = 1{,}50$ arredonda-se para o inteiro 2.
- $x = 2{,}50$ arredonda-se para o inteiro 3.

```
>>> round(1.49)
1.0
>>> round(1.50)
2.0
>>> round(2.50)
3.0
```

Exemplo 2.3.2. Represente os números $x_1 = 0{,}567$, $x_2 = 0{,}233$, $x_3 = -0{,}675$ e $x_4 = 0{,}314159265\ldots \times 10^1$ com dois dígitos significativos por truncamento e arredondamento.

Solução. Vejamos cada caso:

- Por truncamento:

$$x_1 = 0{,}56, \quad x_2 = 0{,}23, \quad x_3 = -0{,}67 \quad \text{e} \quad x_4 = 3{,}1. \tag{2.42}$$

  Em `Python`, podemos obter a representação de $x_3 = -0{,}675$ fazendo:

```
>>> int(-0.675*1e2)/1e2
```

  e, em notação normalizada, temos:

```
>>> print("%1.1e" % (int(-0.675*1e2)/1e2))
-6.7e-01
```

- Por arredondamento:

$$x_1 = 0{,}57; \quad x_2 = 0{,}23; \quad x_3 = -0{,}68 \quad \text{e} \quad x_4 = 3{,}1. \tag{2.43}$$

  Em `Python`, a representação de números por arredondamento é o padrão. Assim, para obtermos a representação desejada de $x_3 = 0{,}675$ fazemos:

```
>>> print("%1.2f" % (-0.675))
-0.68
```

  e, em notação normalizada:

```
>>> print("%1.1e" % (-0.675))
-6.8e-01
```

$\Diamond$

Observação 2.3.2. Observe que o arredondamento pode mudar todos os dígitos e o expoente da representação em ponto flutuante de um número dado. Por exemplo, o arredondamento de $0{,}9999 \times 10^{-1}$ com 3 dígitos significativos é $0{,}1 \times 10^0$.

Exercícios resolvidos

Esta seção carece de exercícios resolvidos. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

Exercícios

E 2.3.1. Aproxime os seguintes números para 2 dígitos significativos por arredondamento por truncamento.

(a) 1,159

(b) 7,399

(c) −5,901

E 2.3.2. Aproxime os seguintes números para 2 dígitos significativos por arredondamento por proximidade com desempate par.

(a) 1,151

(b) 1,15

(c) 2,45

(d) −2,45

E 2.3.3. O `Python` usa arredondamento por proximidade com desempate par como padrão. Assim sendo, por exemplo

```
>>> print('%1.1e\n' % 1.25)
1.2e+00
```

Agora:

```
>>> print('%1.1e\n' % 2.45)
2.5e+00
```

Não deveria ser 2.4? Explique o que está ocorrendo.

2.4 Representação de números em máquina

Os computadores, em geral, usam a base binária para representar os números, onde as posições, chamadas de bits, assumem as condições "verdadeiro" ou "falso", ou seja, 1 ou 0, respectivamente. Os computadores representam os números com uma quantidade fixa de bits, o que se traduz em um conjunto finito de números representáveis. Os demais números são tomados por proximidade àqueles conhecidos, gerando erros de arredondamento. Por exemplo, em aritmética de computador, o número 2 tem representação exata, logo $2^2 = 4$, mas $\sqrt{3}$ não tem representação finita, logo $(\sqrt{3})^2 \neq 3$.

Veja isso em `Python`:

```
>>> 2**2 == 4
True
>>> np.sqrt(3)**2 == 3
False
```

2.4.1 Números inteiros

Tipicamente, um número inteiro é armazenado em um computador como uma sequência de dígitos binários de comprimento fixo denominado **registro**.

Representação sem sinal

Um registro com n bits da forma

d_{n-1}	d_{n-2}	$\cdots$	d_1	d_0

representa o número $(d_{n-1}d_{n-2}...d_1d_0)_2$.

Assim, é possível representar números inteiros entre $2^n - 1$ e 0, sendo

$$\begin{aligned} [111\ldots111] &= 2^{n-1} + 2^{n-2} + \cdots + 2^1 + 2^0 = 2^n - 1, \\ &\vdots \\ [000\ldots011] &= 3, \\ [000\ldots010] &= 2, \\ [000\ldots001] &= 1, \\ [000\ldots000] &= 0. \end{aligned} \tag{2.44}$$

Representação com bit de sinal

O bit mais significativo (o primeiro à esquerda) representa o sinal: por convenção, 0 significa positivo e 1 significa negativo. Um registro com n bits da forma

s	d_{n-2}	$\cdots$	d_1	d_0

representa o número $(-1)^s(d_{n-2}\dots d_1d_0)_2$. Assim, é possível representar números inteiros entre -2^{n-1} e 2^{n-1}, com duas representações para o zero: $(1000\dots000)_2$ e $(00000\dots000)_2$.

Exemplo 2.4.1. Em um registro com 8 bits, teremos os números

$$\begin{aligned}
[11111111] &= -(2^6+\cdots+2+1) = -127,\\
&\vdots\\
[10000001] &= -1,\\
[10000000] &= -0,\\
[01111111] &= 2^6+\cdots+2+1 = 127,\\
&\vdots\\
[00000010] &= 2,\\
[00000001] &= 1,\\
[00000000] &= 0.
\end{aligned} \tag{2.45}$$

Representação complemento de dois

O bit mais significativo (o primeiro à esquerda) representa o coeficiente de -2^{n-1}. Um registro com n bits da forma:

d_{n-1}	d_{n-2}	$\cdots$	d_1	d_0

representa o número $-d_{n-1}2^{n-1}+(d_{n-2}\dots d_1d_0)_2$.

Observação 2.4.1. Note que todo registro começando com 1 será um número negativo.

Exemplo 2.4.2. O registro com 8 bits [01000011] representa o número:

$$-0(2^7)+(1000011)_2 = 2^6+2+1 = 67. \tag{2.46}$$

Agora, o registro [10111101] representa:

$$-1(2^7)+(0111101)_2 = -128+2^5+2^4+2^3+2^2+1 = -67. \tag{2.47}$$

Note que podemos obter a representação de -67 invertendo os dígitos de 67 em binário e somando 1.

Exemplo 2.4.3. Em um registro com 8 bits, teremos os números

$$\begin{aligned}
[11111111] &= -2^7 + 2^6 + \cdots + 2 + 1 = -1 \\
&\vdots \\
[10000001] &= -2^7 + 1 = -127 \\
[10000000] &= -2^7 = -128 \\
[01111111] &= 2^6 + \cdots + 2 + 1 = 127 \\
&\vdots \\
[00000010] &= 2 \\
[00000001] &= 1 \\
[00000000] &= 0
\end{aligned} \tag{2.48}$$

2.4.2 Sistema de ponto fixo

O sistema de ponto fixo representa as partes inteira e fracionária do número com uma quantidade fixas de dígitos.

Exemplo 2.4.4. Em um computador de 32 bits que usa o sistema de ponto fixo, o registro

d_{31}	d_{30}	d_{29}	$\cdots$	d_1	d_0

pode representar o número

- $(-1)^{d_{31}}(d_{30}d_{29}\cdots d_{17}d_{16}, d_{15}d_{14}\cdots d_1d_0)_2$ se o sinal for representado por um dígito. Observe que, neste caso, o zero possui duas representações possíveis:

  $$[10000000000000000000000000000000] \tag{2.49}$$

  e

  $$[00000000000000000000000000000000] \tag{2.50}$$

- $(d_{30}d_{29}\cdots d_{17}d_{16})_2 - d_{31}(2^{15} - 2^{-16}) + (0{,}d_{15}d_{14}\cdots d_1d_0)_2$ se o sinal do número estiver representado por uma implementação em complemento de um. Observe que o zero também possui duas representações possíveis:

  $$[11111111111111111111111111111111] \tag{2.51}$$

  e

  $$[00000000000000000000000000000000] \tag{2.52}$$

- $(d_{30}d_{29}\cdots d_{17}d_{16})_2 - d_{31}2^{15} + (0{,}d_{15}d_{14}\cdots d_1d_0)_2$ se o sinal do número estiver representado por uma implementação em complemento de dois. Nesse caso o zero é unicamente representado por

$$[00000000000000000000000000000000] \tag{2.53}$$

Observe que 16 dígitos são usados para representar a parte fracionária, 15 são para representar a parte inteira e um dígito, o d_{31}, está relacionado ao sinal do número.

2.4.3 Sistema de ponto flutuante

O sistema de ponto flutuante não possui quantidade fixa de dígitos para as partes inteira e fracionária do número.

Podemos definir uma máquina F em ponto flutuante de dois modos:

$$F(\beta,|M|,|E|,BIAS) \text{ ou } F(\beta,|M|,E_{MIN},E_{MAX}) \tag{2.54}$$

onde

- β é a base (em geral 2 ou 10),
- $|M|$ é o número de dígitos da mantissa,
- $|E|$ é o número de dígitos do expoente,
- $BIAS$ é um valor de deslocamento do expoente (veja a seguir),
- E_{MIN} é o menor expoente,
- E_{MAX} é o maior expoente.

Considere uma máquina com um registro de 64 bits e base $\beta = 2$. Pelo padrão IEEE754, 1 bit é usado para o sinal, 11 bits para o expoente e 52 bits são usados para o significando tal que

s	c_{10}	c_9	$\cdots$	c_0	m_1	m_2	$\cdots$	m_{51}	m_{52}

represente o número (o $BIAS = 1023$ por definição)

$$x = (-1)^s M \times 2^{c-BIAS}, \tag{2.55}$$

onde a **característica** é representada por

$$c = (c_{10}c_9\cdots c_1c_0)_2 = c_{10}2^{10} + \cdots + c_1 2^1 + c_0 2^0 \tag{2.56}$$

e o significando por

$$M = (1.m_1m_2\cdots m_{51}m_{52})_2. \tag{2.57}$$

Observação 2.4.2. Em base 2 não é necessário armazenar o primeiro dígito (por quê?).

Exemplo 2.4.5. O registro

$$[0|100\ 0000\ 0000|1010\ 0000\ 0000 \dots 0000\ 0000] \tag{2.58}$$

representa o número

$$(-1)^0(1 + 2^{-1} + 2^{-3}) \times 2^{1024-1023} = (1 + 0.5 + 0.125)2 = 3.25. \tag{2.59}$$

O expoente deslocado

Uma maneira de representar os expoentes inteiros é deslocar todos eles uma mesma quantidade. Desta forma permitimos a representação de números negativos e a ordem deles continua crescente. O expoente é representado por um inteiro sem sinal do qual é deslocado o **BIAS**.

Tendo $|E|$ dígitos para representar o expoente, geralmente o $BIAS$ é predefinido de tal forma a dividir a tabela ao meio de tal forma que o expoente *um* seja representado pelo sequência $[100\dots000]$.

Exemplo 2.4.6. Com 64 bits, pelo padrão $IEEE754$, temos que $|E| := 11$. Assim, $(100\ 0000\ 0000)_2 = 2^{10} = 1024$. Como queremos que esta sequência represente o 1, definimos $BIAS := 1023$, pois

$$1024 - BIAS = 1. \tag{2.60}$$

Com 32 bits, temos $|E| := 8$ e $BIAS := 127$. E com 128 bits, temos $|E| := 15$ e $BIAS := 16383$.

Com $|E| = 11$ temos

$$\begin{aligned}
[111\ 1111\ 1111] &= \text{reservado} \\
[111\ 1111\ 1110] &= 2046 - BIAS = 1023_{10} = E_{MAX} \\
&\vdots = \\
[100\ 0000\ 0001] &= 2^{10} + 1 - BIAS = 2_{10} \\
[100\ 0000\ 0000] &= 2^{10} \quad\ - BIAS = 1_{10} \\
[011\ 1111\ 1111] &= 1023 \quad - BIAS = 0_{10} \\
[011\ 1111\ 1110] &= 1022 \quad - BIAS = -1_{10} \\
&\vdots = \\
[000\ 0000\ 0001] &= 1 - BIAS = -1022 = E_{MIN} \\
[000\ 0000\ 0000] &= \text{reservado}
\end{aligned} \tag{2.61}$$

O maior expoente é dado por $E_{MAX} = 1023$ e o menor expoente é dado por $E_{MIN} = -1022$.

O menor número representável positivo é dado pelo registro

$$[0|000\ 0000\ 0001|0000\ 0000\ 0000\ \ldots\ 0000\ 0000] \tag{2.62}$$

quando $s = 0$, $c = 1$ e $M = (1.000...000)_2$, ou seja,

$$MINR = (1+0)_2 \times 2^{1-1023} \approx 0.2225 \times 10^{-307}. \tag{2.63}$$

O maior número representável é dado por

$$[0|111\ 1111\ 1110|1111\ 1111\ \cdots\ 1111\ 1111] \tag{2.64}$$

quando $s = 0$, $c = 2046$ e $M = (1.1111\ 1111 \cdots\ 1111)_2 = 2 - 2^{-52}$, ou seja,

$$MAXR = (2 - 2^{-52}) \times 2^{2046-1023} \approx 2^{1024} \approx 0.17977 \times 10^{309}. \tag{2.65}$$

Observação 2.4.3. Em `Python`, podemos podemos obter o maior e o menor `double` positivo não nulo com os comandos:

```
>>> import sys
>>> sys.float_info.max
1.7976931348623157e+308
>>> sys.float_info.min
2.2250738585072014e-308
```

Outras informações sobre a representação em ponto flutuante podem ser obtidas com `sys.float_info`.

Casos especiais

O **zero** é um caso especial representado pelo registro

$$[0|000\ 0000\ 0000|0000\ 0000\ 0000...0000\ 0000] \tag{2.66}$$

Os expoentes **reservados** são usados para casos especiais:

- $c = [0000...0000]$ é usado para representar o zero (se $m = 0$) e os números subnormais (se $m \neq 0$).
- $c = [1111...1111]$ é usado para representar o infinito (se $m = 0$) e NaN (se $m \neq 0$).

Os números subnormais[8] tem a forma

$$x = (-1)^s (0.m_1 m_2 \cdots m_{51} m_{52})_2 \times 2^{1-BIAS}. \tag{2.67}$$

[8] Note que poderíamos definir números um pouco menores que o $MINR$.

2.4.4 Precisão e épsilon de máquina

A **precisão** p de uma máquina é o número de dígitos significativos usado para representar um número. Note que $p = |M| + 1$ em binário e $p = |M|$ para outras bases.

O **épsilon de máquina**, $\epsilon_{mach} = \epsilon$, é definido de forma que $1 + \epsilon$ seja o menor número representável maior que 1, isto é, $1 + \epsilon$ é representável, mas não existem números representáveis em $(1, 1 + \epsilon)$.

Exemplo 2.4.7. Com 64 bits, temos que o épsilon será dado por

$$\begin{aligned} 1 \rightarrow \quad & (1.0000\ 0000....0000)_2 \times 2^0 \\ \epsilon \rightarrow + & (0.0000\ 0000....0001)_2 \times 2^0 = 2^{-52} \\ & (1.0000\ 0000....0001)_2 \times 2^0 \neq 1 \end{aligned} \tag{2.68}$$

Assim, $\epsilon = 2^{-52}$.

Em `Python`, podemos obter o épsilon de máquina com os comandos:

```
>>> import sys
>>> sys.float_info.epsilon
2.220446049250313e-16
```

Observe, também, os seguintes resultados:

```
>>> eps = sys.float_info.epsilon
>>> 1 + 1e-16 == 1
True
>>> 1 + eps == 1
False
```

2.4.5 Distribuição dos números

Utilizando uma máquina em ponto flutuante, temos um número finito de números que podemos representar.

Um número muito pequeno geralmente é aproximado por zero (**underflow**) e um número muito grande (**overflow**) geralmente faz o cálculo parar. Além disso, os números não estão uniformemente espaçados no eixo real. Números pequenos estão bem próximos enquanto que números com expoentes grandes estão bem distantes.

Se tentarmos armazenar um número que não é representável, devemos utilizar o número mais próximo, gerando os erros de arredondamento.

Exercícios

E 2.4.1. Usando a representação complemento de dois de números inteiros com 8 **bits**, escreva o número decimal que corresponde aos seguintes barramentos:

a) [01100010].

b) [00011101].

c) [10000000].

d) [11100011].

e) [11111111]

E 2.4.2. Usando a representação complemento de dois de números inteiros com 16 **bits**, escreva o número decimal que corresponde aos seguintes barramentos:

a) [0110001001100010].

b) [0001110100011101].

c) [1110001011100011].

d) [1111111111111111].

E 2.4.3. Usando a representação de ponto flutuante com 64 **bits**, escreva o número decimal que corresponde aos seguintes barramentos:

a) [0|10000000000|111000...0].

b) [1|10000000001|0111000...0].

E 2.4.4. Explique a diferença entre o sistema de ponto fixo e ponto flutuante.

E 2.4.5. Considere a seguinte rotina escrita para ser usada no `Python`:

```
x=1.0
while x+1>x:
    x=x+1
```

Explique se esta rotina finaliza em tempo finito, em caso afirmativo calcule a ordem de grandeza do tempo de execução supondo que cada passo do laço demore $10^{-7}s$. Justifique sua reposta.

2.5 Tipos de erros

Em geral, os números não são representados de forma exata nos computadores. Isto nos leva ao chamado erro de arredondamento. Quando resolvemos problemas com técnicas numéricas, estamos sujeitos a este e outros tipos de erros. Nesta seção, veremos quais são estes erros e como controlá-los, quando possível.

Quando fazemos aproximações numéricas, os erros são gerados de várias formas, sendo as principais delas as seguintes:

1. **Incerteza dos dados** são devidos aos erros nos dados de entrada. Quando o modelo matemático é oriundo de um problema físico, existe incerteza nas medidas feitas pelos instrumentos de medição, que possuem acurácia finita.

2. **Erros de Arredondamento** são aqueles relacionados com as limitações existentes na forma de representar números em máquina.

3. **Erros de Truncamento** surgem quando aproximamos um conceito matemático formado por uma sequência infinita de passos por de um procedimento finito. Por exemplo, a definição de integral é dada por um processo de limite de somas. Numericamente, aproximamos por um soma finita. O erro de truncamento deve ser estudado analiticamente para cada método empregado e normalmente envolve matemática mais avançada que a estudado em um curso de graduação.

Uma questão fundamental é a quantificação dos erros imbricados na computação da solução de um dado problema. Para tanto, precisamos definir medidas de erros (ou de exatidão). As medidas de erro mais utilizadas são o **erro absoluto** e o **erro relativo**.

Definição 2.5.1 (Erro absoluto e relativo)**.** *Seja x um número real e $\overline{x}$, sua aproximação. O* ***erro absoluto*** *da aproximação $\overline{x}$ é definido como*

$$|x - \overline{x}|. \tag{2.69}$$

O ***erro relativo*** *da aproximação $\overline{x}$ é definido como*

$$\frac{|x - \overline{x}|}{|x|}, \quad x \neq 0. \tag{2.70}$$

Observação 2.5.1. Observe que o erro relativo é adimensional e, muitas vezes, é expresso em porcentagens. Mais precisamente, o erro relativo em porcentagem da aproximação $\overline{x}$ é dado por

$$\frac{|x - \bar{x}|}{|x|} \times 100\%. \tag{2.71}$$

Exemplo 2.5.1. Sejam $x = 123456{,}789$ e sua aproximação $\bar{x} = 123000$. O erro absoluto é

$$|x - \bar{x}| = |123456{,}789 - 123000| = 456{,}789 \tag{2.72}$$

e o erro relativo é

$$\frac{|x - \bar{x}|}{|x|} = \frac{456{,}789}{123456{,}789} \approx 0{,}00369999 \text{ ou } 0{,}36\% \tag{2.73}$$

Exemplo 2.5.2. Sejam $y = 1{,}23456789$ e $\bar{y} = 1{,}13$. O erro absoluto é

$$|y - \bar{y}| = |1{,}23456789 - 1{,}13| = 0{,}10456789 \tag{2.74}$$

que parece pequeno se compararmos com o exemplo anterior. Entretanto o erro relativo é

$$\frac{|y - \bar{y}|}{|y|} = \frac{0{,}10456789}{1{,}23456789} \approx 0{,}08469999 \text{ ou } 8{,}4\% \tag{2.75}$$

Note que o erro relativo leva em consideração a escala do problema.

Exemplo 2.5.3. Observe os erros absolutos e relativos em cada caso a seguir:

x	$\bar{x}$	Erro absoluto	Erro relativo
$0{,}\overline{3} \times 10^{-2}$	$0{,}3 \times 10^{-2}$	$0{,}\overline{3} \times 10^{-3}$	10%
$0{,}\overline{3}$	$0{,}3$	$0{,}\overline{3} \times 10^{-2}$	10%
$0{,}\overline{3} \times 10^{2}$	$0{,}3 \times 10^{2}$	$0{,}\overline{3} \times 10^{1}$	10%

Outra forma de medir a exatidão de uma aproximação numérica é contar o **número de dígitos significativos corretos** em relação ao valor exato.

Definição 2.5.2 (Número de dígitos significativos corretos)**.** *A aproximação $\overline{x}$ de um número x tem s* ***dígitos significativos corretos*** *quando*[9]

$$\frac{|x - \overline{x}|}{|x|} < 5 \times 10^{-s}. \tag{2.77}$$

[9]Esta definição é apresentada em [3]. Não existe uma definição única na literatura para o conceito de dígitos significativos corretos, embora não precisamente equivalentes, elas transmitem o mesmo conceito. Uma maneira de interpretar essa regra é: calcula-se o erro relativo na forma normalizada e a partir da ordem do expoente temos o número de dígitos significativos corretos. Como queremos o expoente, podemos estimar s por

$$DIGSE(x,\bar{x}) = s \approx int\left|\log_{10}\frac{|x - \overline{x}|}{|x|}\right|. \tag{2.76}$$

Exemplo 2.5.4. Vejamos os seguintes casos:

a) A aproximação de $x = 0{,}333333$ por $\overline{x} = 0{,}333$ tem 3 dígitos significativos corretos, pois

$$\frac{|x - \overline{x}|}{|x|} = \frac{0{,}000333}{0{,}333333} \approx 0{,}000999 \leq 5 \times 10^{-3}. \tag{2.78}$$

b) Considere as aproximações $\bar{x}_1 = 0{,}666$ e $\bar{x}_2 = 0{,}667$ de $x = 0{,}666888$. Os erros relativos são

$$\frac{|x - \bar{x}_1|}{|x|} = \frac{|0{,}666888 - 0{,}666|}{0{,}666888} \approx 0{,}00133... < 5 \times 10^{-3}. \tag{2.79}$$

$$\frac{|x - \bar{x}_2|}{|x|} = \frac{|0{,}666888 - 0{,}667|}{0{,}666888} \approx 0{,}000167... < 5 \times 10^{-4}. \tag{2.80}$$

Note que $\bar{x}_1$ possui 3 dígitos significativos corretos e $\bar{x}_2$ possui 4 dígitos significativos (o quarto dígito é o dígito 0 que não aparece a direita, i.e, $\bar{x}_2 = 0.6670$. Isto também leva a conclusão que x_2 aproxima melhor o valor de x do que x_1 pois está mais próximo de x.

c) $\overline{x} = 9{,}999$ aproxima $x = 10$ com 4 dígitos significativos corretos, pois

$$\frac{|x - \overline{x}|}{|x|} = \frac{|10 - 9{,}999|}{10} \approx 0{,}0000999... < 5 \times 10^{-4}. \tag{2.81}$$

d) Considere as aproximações $\overline{x}_1 = 1{,}49$ e $\overline{x}_2 = 1{,}5$ de $x = 1$. Da definição, temos que 1,49 aproxima 1 com um dígito significativo correto (verifique), enquanto 1,5 tem zero dígito significativo correto, pois:

$$\frac{|1 - 1{,}5|}{|1|} = 5 \times 10^{-1} < 5 \times 10^{0}. \tag{2.82}$$

Exercícios

E 2.5.1. Calcule os erros absoluto e relativo das aproximações $\bar{x}$ para x em cada caso:

a) $x = \pi = 3{,}14159265358979\ldots$ e $\bar{x} = 3{,}141$

b) $x = 1{,}00001$ e $\bar{x} = 1$

c) $x = 100001$ e $\bar{x} = 100000$

E 2.5.2. Arredonde os seguintes números para cinco algarismos significativos:

a) 1,7888544

b) 1788,8544

c) 0,0017888544

d) 0,004596632

e) $2{,}1754999 \times 10^{-10}$

f) $2{,}1754999 \times 10^{10}$

E 2.5.3. Represente os seguintes números com três dígitos significativos usando arredondamento por truncamento e arredondamento por proximidade.

a) 3276.

b) 42,55.

c) 0,00003331.

E 2.5.4. Usando a Definição 2.5.2, verifique quantos são os dígitos significativos corretos na aproximação de x por $\bar{x}$.

a) $x = 2{,}5834$ e $\bar{x} = 2{,}6$

b) $x = 100$ e $\bar{x} = 99$

E 2.5.5. Resolva a equação $0{,}1x - 0{,}01 = 12$ usando arredondamento com três dígitos significativos em cada passo e compare com o resultado exato.

E 2.5.6. Calcule o erro relativo e absoluto envolvido nas seguintes aproximações e expresse as respostas com três algarismos significativos corretos.

a) $x = 3{,}1415926535898$ e $\tilde{x} = 3{,}141593$

b) $x = \frac{1}{7}$ e $\tilde{x} = 1{,}43 \times 10^{-1}$

2.6 Erros nas operações elementares

O erro relativo presente nas operações elementares de adição, subtração, multiplicação e divisão é da ordem do épsilon de máquina. Se estivermos usando uma máquina com 64 bits, temos que $\epsilon = 2^{-52} \approx 2{,}22E-16$.

Este erro é bem pequeno para a maioria das aplicações! Assumindo que x e y são representados com todos dígitos corretos, temos aproximadamente 15 dígitos significativos corretos quando fazemos uma das operações $x+y$, $x-y$, $x \times y$ ou x/y.

Mesmo que fizéssemos, por exemplo, 1000 operações elementares sucessivas em ponto flutuante, teríamos, no pior dos casos, acumulado todos esses erros e perdido 3 casas decimais ($1000 \times 10^{-15} \approx 10^{-12}$).

Entretanto, quando subtraímos números muito próximos, o erro pode se propagar de forma catastrófico.

2.7 Cancelamento catastrófico

Quando fazemos subtrações com números muito próximos entre si, ocorre o que chamamos de "cancelamento catastrófico", onde podemos perder vários dígitos de precisão em uma única subtração.

Exemplo 2.7.1. Efetue a operação

$$0{,}987624687925 - 0{,}987624 = 0{,}687925 \times 10^{-6} \tag{2.86}$$

usando arredondamento com seis dígitos significativos e observe a diferença se comparado com resultado sem arredondamento.

Solução. Os números arredondados com seis dígitos para a mantissa resultam na seguinte diferença

$$0{,}987625 - 0{,}987624 = 0{,}100000 \times 10^{-5} \tag{2.87}$$

Observe que os erros relativos entre os números exatos e aproximados no lado esquerdo são bem pequenos,

$$\frac{|0{,}987624687925 - 0{,}987625|}{|0{,}987624687925|} = 0{,}00003159 \tag{2.88}$$

e

$$\frac{|0{,}987624 - 0{,}987624|}{|0{,}987624|} = 0\%, \tag{2.89}$$

enquanto no lado direito o erro relativo é enorme:

$$\frac{|0{,}100000 \times 10^{-5} - 0{,}687925 \times 10^{-6}|}{0{,}687925 \times 10^{-6}} = 45{,}36\%. \tag{2.90}$$

$\Diamond$

Exemplo 2.7.2. Considere o problema de encontrar as raízes da equação de segundo grau

$$x^2 + 300x - 0{,}014 = 0, \tag{2.91}$$

usando seis dígitos significativos.

Aplicando a fórmula de Bhaskara com $a = 0{,}100000 \times 10^{1}$, $b = 0{,}300000 \times 10^{3}$ e $c = 0{,}140000 \times 10^{-1}$, temos o discriminante:

$$\begin{aligned}
\Delta &= b^2 - 4 \cdot a \cdot c && (2.92)\\
&= 0{,}300000 \times 10^{3} \times 0{,}300000 \times 10^{3} && (2.93)\\
&+ \ 0{,}400000 \times 10^{1} \times 0{,}100000 \times 10^{1} \times 0{,}140000 \times 10^{-1} && (2.94)\\
&= 0{,}900000 \times 10^{5} + 0{,}560000 \times 10^{-1} && (2.95)\\
&= 0{,}900001 \times 10^{5} && (2.96)
\end{aligned}$$

e as raízes:

$$\begin{aligned}
x_1,x_2 &= \frac{-0{,}300000 \times 10^{3} \pm \sqrt{\Delta}}{0{,}200000 \times 10^{1}} && (2.97)\\
&= \frac{-0{,}300000 \times 10^{3} \pm \sqrt{0{,}900001 \times 10^{5}}}{0{,}200000 \times 10^{1}} && (2.98)\\
&= \frac{-0{,}300000 \times 10^{3} \pm 0{,}300000 \times 10^{3}}{0{,}200000 \times 10^{1}} && (2.99)\\
& && (2.100)
\end{aligned}$$

Então, as duas raízes obtidas com erros de arredondamento, são:

$$\begin{aligned}
\tilde{x}_1 &= \frac{-0{,}300000 \times 10^{3} - 0{,}300000 \times 10^{3}}{0{,}200000 \times 10^{1}} \\
&= -\frac{0{,}600000 \times 10^{3}}{0{,}200000 \times 10^{1}} = -0{,}300000 \times 10^{3}
\end{aligned} \tag{2.101}$$

e

$$\tilde{x}_2 = \frac{-0{,}300000 \times 10^{3} + 0{,}300000 \times 10^{3}}{0{,}200000 \times 10^{1}} = 0{,}000000 \times 10^{0} \tag{2.102}$$

No entanto, os valores das raízes com seis dígitos significativos livres de erros de arredondamento, são:

$$x_1 = -0{,}300000 \times 10^{3} \quad \text{e} \quad x_2 = 0{,}466667 \times 10^{-4}. \tag{2.103}$$

Observe que a primeira raiz apresenta seis dígitos significativos corretos, mas a segunda não possui nenhum dígito significativo correto.

Observe que isto acontece porque b^2 é muito maior que $4ac$, ou seja, $b \approx \sqrt{b^2-4ac}$, logo a diferença

$$-b+\sqrt{b^2-4ac} \tag{2.104}$$

estará próxima de zero. Uma maneira de evitar o cancelamento catastrófico é aplicar procedimentos analíticos na expressão para eliminar essa diferença. Um técnica padrão consiste usar uma expansão em série de Taylor em torno da origem, tal como:

$$\sqrt{1-x} = 1 - \frac{1}{2}x + O(x^2). \tag{2.105}$$

Substituindo esta aproximação na fórmula de Bhaskara, temos:

$$\begin{aligned} x &= \frac{-b \pm \sqrt{b^2-4ac}}{2a} && (2.106)\\ &= \frac{-b \pm b\sqrt{1-\frac{4ac}{b^2}}}{2a} && (2.107)\\ &\approx \frac{-b \pm b\left(1-\frac{4ac}{2b^2}\right)}{2a} && (2.108)\\ & && (2.109) \end{aligned}$$

Observe que $\frac{4ac}{b^2}$ é um número pequeno e por isso a expansão faz sentido. Voltamos no exemplo anterior e calculamos as duas raízes com o nova expressão

$$\begin{aligned} \tilde{x}_1 &= \frac{-b-b+\frac{4ac}{2b}}{2a} = -\frac{b}{a}+\frac{c}{b} && (2.110)\\ &= -\frac{0{,}300000\times 10^3}{0{,}100000\times 10^1} - \frac{0{,}140000\times 10^{-1}}{0{,}300000\times 10^3} && (2.111)\\ &= -0{,}300000\times 10^3 - 0{,}466667\times 10^{-4} && (2.112)\\ &= -0{,}300000\times 10^3 && (2.113) \end{aligned}$$

$$\begin{aligned} \tilde{x}_2 &= \frac{-b+b-\frac{4ac}{2b}}{2a} && (2.114)\\ &= -\frac{4ac}{4ab} && (2.115)\\ &= -\frac{c}{b} = -\frac{-0{,}140000\times 10^{-1}}{0{,}300000\times 10^3} = 0{,}466667\times 10^{-4} && (2.116)\\ & && (2.117) \end{aligned}$$

Observe que o efeito catastrófico foi eliminado.

Observação 2.7.1. O cancelamento catastrófico também poderia ter sido evitado através do seguinte truque analítico:

$$x_2 = \frac{-b+\sqrt{b^2-4ac}}{2a} = \frac{-b+\sqrt{b^2-4ac}}{2a} \cdot \frac{-b-\sqrt{b^2-4ac}}{-b-\sqrt{b^2-4ac}} \tag{2.118}$$

$$= \frac{b^2-(b^2-4ac)}{2a\left(-b-\sqrt{b^2-4ac}\right)} = \frac{4ac}{2a\left(-b-\sqrt{b^2-4ac}\right)} \tag{2.119}$$

$$= -\frac{2c}{\left(b+\sqrt{b^2-4ac}\right)} \tag{2.120}$$

2.8 Condicionamento de um problema

Nesta seção, utilizaremos a seguinte descrição abstrata para o conceito de "resolver um problema": dado um conjunto de dados de entrada, encontrar os dados de saída. Se denotamos pela variável x os dados de entrada e pela variável y os dados de saída, resolver o problema significa encontrar y dado x. Em termos matemáticos, a resolução de um problema é realizada pelo mapeamento $f : x \to y$, ou simplesmente $y = f(x)$.

É certo que, na maioria das aplicações, os dados de entrada do problema — isto é, x — não são conhecidos com total exatidão, devido a diversas fontes de erros, como incertezas na coleta dos dados e erros de arredondamento. O conceito de condicionamento está relacionado à forma como os erros nos dados de entrada influenciam os dados de saída.

Para fins de análise, denotaremos por x, os dados de entrada com precisão absoluta e por x^*, os dados com erro. Definiremos também a solução y^*, do problema com dados de entrada x^*, ou seja, $y^* = f(x^*)$.

Estamos interessados em saber se os erros cometidos na entrada $\Delta x = x - x^*$ influenciaram na saída do problema $\Delta y = y - y^*$. No caso mais simples, temos que $x \in \mathbb{R}$ e $y \in \mathbb{R}$. Assumindo que f seja diferenciável, a partir da série de Taylor

$$f(x+\Delta x) \approx f(x) + f'(x)\Delta x \tag{2.121}$$

obtemos (subtraindo $f(x)$ dos dois lados)

$$\Delta y = f(x+\Delta x) - f(x) \approx f'(x)\Delta x \tag{2.122}$$

Para relacionarmos os erros relativos, dividimos o lado esquerdo por y, o lado direito por $f(x) = y$ e obtemos

$$\frac{\Delta y}{y} \approx \frac{f'(x)}{f(x)}\frac{x\Delta x}{x} \tag{2.123}$$

sugerindo a definição de número de condicionamento de um problema.

Definição 2.8.1. *Seja f uma função diferenciável. O* ***número de condicionamento*** *de um problema é definido como*

$$\kappa_f(x) := \left|\frac{xf'(x)}{f(x)}\right| \tag{2.124}$$

e fornece uma estimativa de quanto os erros relativos na entrada $\left|\frac{\Delta x}{x}\right|$ serão amplificados na saída $\left|\frac{\Delta y}{y}\right|$.

De modo geral, quando f depende de várias variáveis, podemos obter

$$\delta_f = |f(x_1,x_2,...,x_n) - f(\bar{x}_1,\bar{x}_2,...,\bar{x}_n)| \approx \sum_{i=1}^{n} \left|\frac{\partial f}{\partial x_i}(x_1,x_2,...,x_n)\right| \delta_{x_i} \tag{2.125}$$

Uma matriz de números de condicionamento também poderia ser obtida como em [5].

Exemplo 2.8.1. Considere o problema de calcular $\sqrt{x}$ em $x = 2$. Se usarmos $x^* = 1{,}999$, quanto será o erro relativo na saída? O erro relativo na entrada é

$$\left|\frac{\Delta x}{x}\right| = \left|\frac{2 - 1{,}999}{2}\right| = 0{,}0005 \tag{2.126}$$

O número de condicionamento do problema calcular a raiz é

$$\kappa_f(x) := \left|\frac{xf'(x)}{f(x)}\right| = \left|\frac{x\frac{1}{2\sqrt{x}}}{\sqrt{x}}\right| = \frac{1}{2} \tag{2.127}$$

Ou seja, os erros na entrada serão diminuídos pela metade. De fato, usando $y = \sqrt{2} = 1{,}4142136...$ e $y^* = \sqrt{1{,}999} = 1{,}41386...$, obtemos

$$\frac{\Delta y}{y} = \frac{\sqrt{2} - \sqrt{1{,}999}}{\sqrt{2}} \approx 0{,}000250031... \tag{2.128}$$

Exemplo 2.8.2. Considere a função $f(x) = \frac{10}{1-x^2}$ e $x^* = 0{,}9995$ com um erro absoluto na entrada de 0,0001.

Calculando $y^* = f(x^*)$ temos

$$y^* = \frac{10}{1 - (0{,}9995)^2} \approx 10002{,}500625157739705173 \tag{2.129}$$

Mas qual é a estimativa de erro nessa resposta? Quantos dígitos significativos temos nessa resposta?

Sabendo que $f'(x) = -20x/(1-x^2)^2$, o número de condicionamento é

$$\kappa_f(x) := \left|\frac{xf'(x)}{f(x)}\right| = \left|\frac{2x^2}{1-x^2}\right| \tag{2.130}$$

o que nos fornece para $x^* = 0{,}9995$,

$$\kappa_f(0{,}9995) \approx 1998{,}5 \tag{2.131}$$

Como o erro relativo na entrada é

$$\left|\frac{\Delta x}{x}\right| = \left|\frac{0{,}0001}{0{,}9995}\right| \approx 0{,}00010005... \tag{2.132}$$

temos que o erro na saída será aproximadamente

$$\left|\frac{\Delta y}{y}\right| \approx \kappa_f(x)\left|\frac{\Delta x}{x}\right| \approx 1998{,}5 \times 0{,}00010005... \approx 0{,}1999 \tag{2.133}$$

ou seja um erro relativo de aproximadamente 19,99%.

Note que se usarmos $x_1 = 0{,}9994$ e $x_2 = 0{,}9996$ (ambos no intervalo do erro absoluto da entrada) encontramos

$$y_1^* \approx 8335{,}83 \tag{2.134}$$
$$y_2^* \approx 12520{,}50 \tag{2.135}$$

confirmando a estimativa de 19,99%.

Exemplo 2.8.3. Seja $f(x) = x\exp(x)$. Calcule o erro absoluto ao calcular $f(x)$ sabendo que $x = 2 \pm 0{,}05$.

Solução. Temos que $x \approx 2$ com erro absoluto de $\delta_x = 0{,}05$. Neste caso, calculamos δ_f, isto é, o erro absoluto ao calcular $f(x)$, por:

$$\delta_f = |f'(x)|\delta_x. \tag{2.136}$$

Como $f'(x) = (1+x)e^x$, temos:

$$\delta_f = |(1+x)e^x| \cdot \delta_x \tag{2.137}$$
$$= |3e^2| \cdot 0{,}05 = 1{,}1084. \tag{2.138}$$

Portanto, o erro absoluto ao calcular $f(x)$ quando $x = 2 \pm 0{,}05$ é de 1,1084. ◊

Exemplo 2.8.4. Calcule o erro relativo ao medir $f(x,y) = \frac{x^2+1}{x^2}e^{2y}$ sabendo que $x \approx 3$ é conhecido com 10% de erro e $y \approx 2$ é conhecido com 3% de erro.

Solução. Calculamos as derivadas parciais de f:

$$\frac{\partial f}{\partial x} = \frac{2x^3 - (2x^3 + 2x)}{x^4} e^{2y} = -\frac{2e^{2y}}{x^3} \tag{2.139}$$

e

$$\frac{\partial f}{\partial y} = 2\frac{x^2 + 1}{x^2} e^{2y} \tag{2.140}$$

Calculamos o erro absoluto em termos do erro relativo:

$$\frac{\delta_x}{|x|} = 0{,}1 \Rightarrow \delta_x = 3 \cdot 0{,}1 = 0{,}3 \tag{2.141}$$

$$\frac{\delta_y}{|y|} = 0{,}03 \Rightarrow \delta_y = 2 \cdot 0{,}03 = 0{,}06 \tag{2.142}$$

Aplicando a expressão para estimar o erro em f temos

$$\delta_f = \left|\frac{\partial f}{\partial x}\right| \delta_x + \left|\frac{\partial f}{\partial y}\right| \delta_y \tag{2.143}$$

$$= \frac{2e^4}{27} \cdot 0{,}3 + 2\frac{9+1}{9} e^4 \cdot 0{,}06 = 8{,}493045557 \tag{2.144}$$

Portanto, o erro relativo ao calcular f é estimado por

$$\frac{\delta f}{|f|} = \frac{8{,}493045557}{\frac{9+1}{9} e^4} = 14\% \tag{2.145}$$

$\Diamond$

Exemplo 2.8.5. No exemplo anterior, reduza o erro relativo em x pela metade e calcule o erro relativo em f. Depois, repita o processo reduzindo o erro relativo em y pela metade.

Solução. Na primeira situação temos $x = 3$ com erro relativo de 5% e $\delta_x = 0{,}05 \cdot 3 = 0{,}15$. Calculamos $\delta_f = 7{,}886399450$ e o erro relativo em f de 13%. Na segunda situação, temos $y = 2$ com erro de 1,5% e $\delta_y = 2 \cdot 0{,}015 = 0{,}03$. Calculamos $\delta_f = 4{,}853168892$ e o erro relativo em f de 8%. Observe que mesma o erro relativo em x sendo maior, o erro em y é mais significante na função. $\Diamond$

Exemplo 2.8.6. Considere um triângulo retângulo onde a hipotenusa e um dos catetos são conhecidos a menos de um erro: hipotenusa $a = 3 \pm 0{,}01$ metros e cateto $b = 2 \pm 0{,}01$ metros. Calcule o erro absoluto ao calcular a área dessa triângulo.

Solução. Primeiro vamos encontrar a expressão para a área em função da hipotenusa a e um cateto b. A tamanho de segundo cateto c é dado pelo teorema de Pitágoras, $a^2 = b^2 + c^2$, ou seja, $c = \sqrt{a^2 - b^2}$. Portanto a área é

$$A = \frac{bc}{2} = \frac{b\sqrt{a^2 - b^2}}{2}. \tag{2.146}$$

Agora calculamos as derivadas

$$\frac{\partial A}{\partial a} = \frac{ab}{2\sqrt{a^2-b^2}}, \tag{2.147}$$

$$\frac{\partial A}{\partial b} = \frac{\sqrt{a^2-b^2}}{2} - \frac{b^2}{2\sqrt{a^2-b^2}}, \tag{2.148}$$

e substituindo na estimativa para o erro δ_A em termos de $\delta_a = 0{,}01$ e $\delta_b = 0{,}01$:

$$\delta_A \approx \left|\frac{\partial A}{\partial a}\right|\delta_a + \left|\frac{\partial A}{\partial b}\right|\delta_b \tag{2.149}$$

$$\approx \frac{3\sqrt{5}}{5}\cdot 0{,}01 + \frac{\sqrt{5}}{10}\cdot 0{,}01 = 0{,}01565247584 \tag{2.150}$$

Em termos do erro relativo temos erro na hipotenusa de $\frac{0{,}01}{3} \approx 0{,}333\%$, erro no cateto de $\frac{0{,}01}{2} = 0{,}5\%$ e erro na área de

$$\frac{0{,}01565247584}{\frac{2\sqrt{3^2-2^2}}{2}} = 0{,}7\% \tag{2.151}$$

◊

Exercícios

E 2.8.1. Considere que a variável $x \approx 2$ é conhecida com um erro relativo de 1% e a variável $y \approx 10$ com um erro relativo de 10%. Calcule o erro relativo associado a z quando:

$$z = \frac{y^4}{1+y^4}e^x. \tag{2.152}$$

Suponha que você precise conhecer o valor de z com um erro de 0,5%. Você propõe uma melhoria na medição da variável x ou y? Explique.

E 2.8.2. A corrente I em ampères e a tensão V em volts em uma lâmpada se relacionam conforme a seguinte expressão:

$$I = \left(\frac{V}{V_0}\right)^{\alpha}, \tag{2.153}$$

onde α é um número entre 0 e 1 e V_0 é tensão nominal em volts. Sabendo que $V_0 = 220 \pm 3\%$ e $\alpha = -0{,}8 \pm 4\%$, calcule a corrente e o erro relativo associado quando a tensão vale $220 \pm 1\%$.
Obs:. Este problema pode ser resolvido de duas formas distintas: usando a expressão aproximada para a propagação de erro e inspecionando os valores máximos e mínimos que a expressão pode assumir. Pratique os dois métodos.**Dica:** lembre que $x^{\alpha} = e^{\alpha \ln(x)}$

2.9 Exemplos selecionados de cancelamento catastrófico

Exemplo 2.9.1. Considere o seguinte processo iterativo:

$$x^{(1)} = \frac{1}{3} \tag{2.154}$$

$$x^{(n+1)} = 4x^{(n)} - 1, \quad n = 1, 2, \ldots \tag{2.155}$$

Observe que $x^{(1)} = \frac{1}{3}$, $x^{(2)} = 4 \cdot \frac{1}{3} - 1 = \frac{1}{3}$, $x^{(3)} = \frac{1}{3}$, ou seja, temos uma sequência constante igual a $\frac{1}{3}$. No entanto, ao calcularmos no computador, usando o sistema de numeração `double`, a sequência obtida não é constante e, de fato, diverge. Faça o teste em `Python`, colocando:

```
>>> x = 1/3; x
```

e itere algumas vezes a linha de comando:

```
>>> x = 4*x-1; x
```

Para compreender o que acontece, devemos levar em consideração que o número $\frac{1}{3} = 0{,}\overline{3}$ possui um representação infinita tanto na base decimal quanto na base binária. Logo, sua representação de máquina inclui um erro de arredondamento. Seja ϵ a diferença entre o valor exato de $\frac{1}{3}$ e sua representação de máquina, isto é, $\tilde{x}^{(1)} = \frac{1}{3} + \epsilon$. A sequência efetivamente calculada no computador é:

$$\tilde{x}^{(1)} = \frac{1}{3} + \epsilon \tag{2.156}$$

$$\tilde{x}^{(2)} = 4x^{(1)} - 1 = 4\left(\frac{1}{3} + \epsilon\right) - 1 = \frac{1}{3} + 4\epsilon \tag{2.157}$$

$$\tilde{x}^{(3)} = 4x^{(2)} - 1 = 4\left(\frac{1}{3} + 4\epsilon\right) - 1 = \frac{1}{3} + 4^2\epsilon \tag{2.158}$$

$$\vdots \tag{2.159}$$

$$\tilde{x}^{(n)} = \frac{1}{3} + 4^{(n-1)}\epsilon \tag{2.160}$$

Portanto o limite da sequência diverge,

$$\lim_{x \to \infty} |\tilde{x}^{(n)}| = \infty \tag{2.161}$$

Qual o número de condicionamento desse problema?

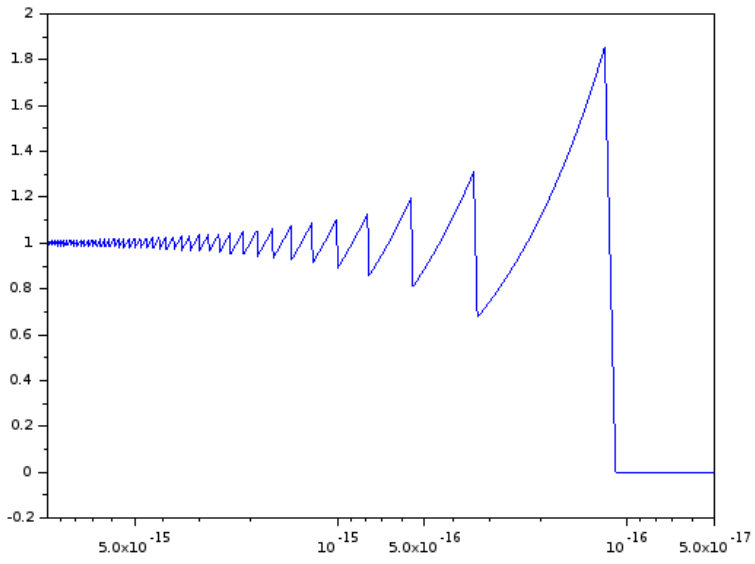


Figura 2.1: Gráfico na função do Exemplo 2.9.2.

Exemplo 2.9.2. Observe a seguinte identidade

$$f(x) = \frac{(1+x)-1}{x} = 1 \tag{2.162}$$

Calcule o valor da expressão à esquerda para $x = 10^{-12}$, $x = 10^{-13}$, $x = 10^{-14}$, $x = 10^{-15}$, $x = 10^{-16}$ e $x = 10^{-17}$. Observe que quando x se aproxima do ϵ de máquina a expressão perde o significado. Veja a Figura 2.1 com o gráfico de $f(x)$ em escala logarítmica.

Exemplo 2.9.3. Neste exemplo, estamos interessados em compreender mais detalhadamente o comportamento da expressão

$$\left(1+\frac{1}{n}\right)^n \tag{2.163}$$

quando n é um número grande ao computá-la em sistemas de numeral de ponto flutuante com acurácia finita. Um resultado bem conhecido do cálculo nos diz que o limite de (2.163) quando n tende a infinito é o número de Euler:

$$\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n = e = 2{,}718281828459... \tag{2.164}$$

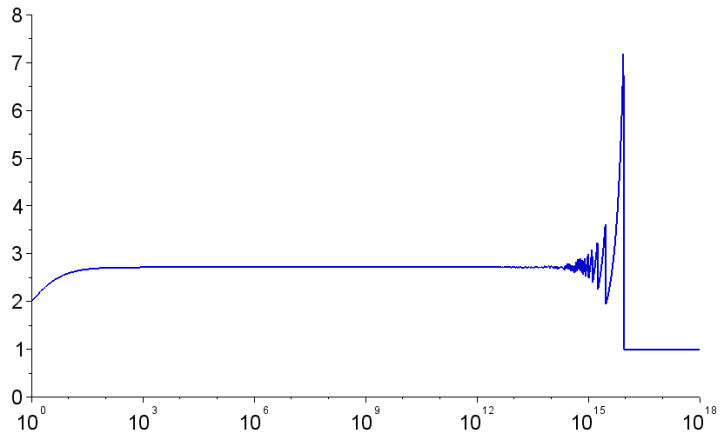


Figura 2.2: Gráfico de $\left(1+\frac{1}{n}\right)^n$ em função de n em escala linear-logarítmica variando de 10^0 até 10^{18}. Veja o Exemplo 2.9.3.

Sabemos também que a sequência produzida por (2.163) é crescente, isto é:

$$\left(1+\frac{1}{1}\right)^1 < \left(1+\frac{1}{2}\right)^2 < \left(1+\frac{1}{3}\right)^3 < \cdots \tag{2.165}$$

No entanto, quando calculamos essa expressão no `Python`, nos defrontamos com o seguinte resultado:

n	$\left(1+\frac{1}{n}\right)^n$		n	$\left(1+\frac{1}{n}\right)^n$
1	2,0000000000000		10^2	2,7048138294215
2	2,2500000000000		10^4	2,7181459268249
3	2,3703703703704		10^6	2,7182804690957
4	2,4414062500000		10^8	2,7182817983391
5	2,4883200000000		10^{10}	2,7182820532348
6	2,5216263717421		10^{12}	2,7185234960372
7	2,5464996970407		10^{14}	2,7161100340870
8	2,5657845139503		10^{16}	1,0000000000000
9	2,5811747917132		10^{18}	1,0000000000000
10	2,5937424601000		10^{20}	1,0000000000000

Podemos resumir esses dados no gráfico de $\left(1+\frac{1}{n}\right)^n$ em função de n, veja a Figura 2.9.

Observe que quando n se torna grande, da ordem de 10^{15}, o gráfico da função deixa de ser crescente e apresenta oscilações. Observe também que a expressão se torna identicamente igual a 1 depois de um certo limiar. Tais fenômenos não são intrínsecos da função $f(n) = (1 + 1/n)^n$, mas **oriundas de erros de arredondamento**, isto é, são resultados numéricos espúrios. A fim de pôr o comportamento numérico de tal expressão, apresentamos abaixo o gráfico da mesma função, porém restrito à região entre 10^{14} e 10^{16}.

Para compreendermos melhor por que existe um limiar N que, quando atingido torna a expressão do exemplo acima identicamente igual a 1, observamos a sequência de operações realizadas pelo computador:

$$n \;\to\; 1/n \;\to\; 1+1/n \;\to\; (1+1/n)^n \tag{2.166}$$

Devido ao limite de precisão da representação de números em ponto flutuante, existe um menor número representável que é maior do que 1. Este número é 1+`eps`, onde `eps` é chamado de **épsilon de máquina** e é o menor número que somado a 1 produz um resultado superior a 1 no sistema de numeração usado. O épsilon de máquina no sistema de numeração **double** vale aproximadamente $2{,}22 \times 10^{-16}$. Em `Python` podemos obter o épsilon de máquina com o seguinte comando `numpy`:

```
>>> eps = np.finfo(float).eps
>>> print(eps)
2.22044604925e-16
>>> 1+eps == 1
False
>>> 1+eps
1.0000000000000002
```

Quando somamos a 1 um número positivo inferior ao épsilon de máquina, obtemos o número 1. Dessa forma, o resultado obtido pela operação de ponto flutuante $1 + n$ para $0 < n < 2{,}22 \times 10^{-16}$ é 1.

Portanto, quando realizamos a sequência de operações dada em (2.166), toda informação contida no número n é perdida na soma com 1 quando $1/n$ é menor que o épsilon de máquina, o que ocorre quando $n > 5 \times 10^{15}$. Assim, $(1 + 1/n)$ é aproximado para 1 e a última operação se resume a 1^n, o que é igual a 1 mesmo quando n é grande.

Um erro comum é acreditar que o perda de significância se deve ao fato de $1/n$ ser muito pequeno para ser representado e é aproximando para 0. Isto é falso, o sistema de ponto de flutuante permite representar números de magnitude muito inferior ao épsilon de máquina. O problema surge da limitação no tamanho da mantissa. Observe como a seguinte sequência de operações não perde significância para números positivos x muito menores que o épsilon de máquina:

$$n \;\to\; 1/n \;\to\; 1/(1/n) \tag{2.167}$$

compare o desempenho numérico desta sequência de operações para valores pequenos de n com o da seguinte sequência:

$$n \;\to\; 1+n \;\to\; (1+n)-1. \tag{2.168}$$

Finalmente, notamos que quando tentamos calcular $\left(1+\frac{1}{n}\right)^n$ para n grande, existe perda de significância no cálculo de $1+1/n$. Para entendermos isso melhor, vejamos o que acontece no `Python` quando $n = 7 \times 10^{13}$:

```
>>> n=7e13; print("%1.15e" % n)
7.000000000000000e+13
>>> n=7e13; print("%1.20e" % n)
7.00000000000000000000e+13
>>> print("%1.20e" % (1/n))
1.42857142857142843451e-14
>>> y=1+1/n; print("%1.20e" % y)
1.00000000000001421085e+00
```

Observe a perda de informação ao deslocar a mantissa de $1/n$. Para evidenciar o fenômenos, observamos o que acontece quando tentamos recalcular n subtraindo 1 de $1 + 1/n$ e invertendo o resultado:

```
>>> print("%1.20e" % (y-1))
1.42108547152020037174e-14
>>> print("%1.20e" % (1/(y-1)))
7.03687441776640000000e+13
```

Exemplo 2.9.4 (Analogia da balança)**.** Observe a seguinte comparação interessante que pode ser feita para ilustrar os sistemas de numeração com ponto fixo e flutuante: o sistema de ponto fixo é como uma balança cujas marcas estão igualmente espaçadas; o sistema de ponto flutuante é como uma balança cuja distância entre as marcas é proporcional à massa medida. Assim, podemos ter uma balança de ponto fixo cujas marcas estão sempre distanciadas de 100g (100g, 200g, 300g, ..., 1Kg, 1,1Kg,...) e outra balança de ponto flutuante cujas marcas estão distanciadas sempre de aproximadamente um décimo do valor lido (100g, 110g, 121g, 133g, ..., 1Kg, 1,1Kg, 1,21Kg, ...) A balança de ponto fixo apresenta uma resolução baixa para pequenas medidas, porém uma resolução alta para grandes medidas. A balança de ponto flutuante distribui a resolução de forma proporcional ao longo da escala.

Seguindo nesta analogia, o fenômeno de perda de significância pode ser interpretado como a seguir: imagine que você deseje obter o peso de um gato (aproximadamente 4Kg). Dois processos estão disponíveis: colocar o gato diretamente na balança ou medir seu peso com o gato e, depois, sem o gato. Na balança de ponto flutuante, a incerteza associada à medida do peso do gato (sozinho) é aproximadamente 10% de 4Kg, isto é, 400g. Já a incerteza associada à medida da uma pessoa (aproximadamente 70Kg) com o gato é de 10% do peso total, isto é, aproximadamente 7Kg. Esta incerteza é da mesma ordem de grandeza da medida a ser realizada, tornado o processo impossível de ser realizado, já que teríamos uma incerteza da ordem de 14Kg (devido à dupla medição) sobre uma grandeza de 4Kg.

Exercícios resolvidos

ER 2.9.1. Deseja-se medir a concentração de dois diferentes oxidantes no ar. Três sensores eletroquímicos estão disponíveis para a medida e apresentam a seguintes respostas:

$$v_1 = 270[A] + 30[B], \quad v_2 = 140[A] + 20[B] \quad \text{e} \quad v_3 = 15[A] + 200[B] \qquad (2.169)$$

as tensões v_1, v_2 e v_3 são dadas em mV e as concentrações em $milimol/l$.

a) Encontre uma expressão para os valores de $[A]$ e $[B]$ em termos de v_1 e v_2 e, depois, em termos de v_1 e v_3. Dica: Se $ad \neq bc$, então a matriz A dada por

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{2.170}$$

é inversível e sua inversa é dada por

$$A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}. \tag{2.171}$$

b) Sabendo que incerteza relativa associada às sensibilidades dos sensores 1 e 2 é de 2% e que a incerteza relativa associada às sensibilidades do sensor 3 é 10%, verifique a incerteza associada à medida feita com o par $1-2$ e o par $1-3$. Use $[A] = [B] = 10 milimol/l$. Dica: Você deve diferenciar as grandezas $[A]$ e $[B]$ em relação aos valores das tensões.

Solução. Em ambos casos, temos a seguinte estrutura:

$$\begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix} \begin{bmatrix} [A] \\ [B] \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \tag{2.172}$$

De forma que

$$\begin{bmatrix} [A] \\ [B] \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix}^{-1} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \frac{1}{S_{11}S_{22} - S_{12}S_{21}} \begin{bmatrix} S_{22} & -S_{12} \\ -S_{21} & S_{11} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \tag{2.173}$$

Portanto

$$[A] = \frac{S_{22}v_1 - S_{12}v_2}{S_{11}S_{22} - S_{12}S_{21}} \tag{2.174}$$

$$[B] = \frac{-S_{21}v_1 + S_{11}v_2}{S_{11}S_{22} - S_{12}S_{21}} \tag{2.175}$$

Usando derivação logarítmica, temos

$$\frac{1}{[A]}\frac{\partial [A]}{\partial S_{11}} = -\frac{S_{22}}{S_{11}S_{22} - S_{12}S_{21}} \tag{2.176}$$

$$\frac{1}{[A]}\frac{\partial [A]}{\partial S_{12}} = -\frac{v_2}{S_{22}v_1 - S_{12}v_2} + \frac{S_{21}}{S_{11}S_{22} - S_{12}S_{21}} = -\frac{[A]}{[B]} \cdot \frac{S_{22}}{S_{11}S_{22} - S_{12}S_{21}} \tag{2.177}$$

$$\frac{1}{[A]}\frac{\partial [A]}{\partial S_{21}} = \frac{S_{12}}{S_{11}S_{22} - S_{12}S_{21}} \tag{2.178}$$

$$\frac{1}{[A]}\frac{\partial [A]}{\partial S_{22}} = \frac{v_1}{S_{22}v_1 - S_{12}v_2} - \frac{S_{11}}{S_{11}S_{22} - S_{12}S_{21}} = \frac{[A]}{[B]} \cdot \frac{S_{12}}{S_{11}S_{22} - S_{12}S_{21}} \tag{2.179}$$

e

$$\frac{1}{[B]}\frac{\partial[B]}{\partial S_{11}} = \frac{v_2}{-S_{21}v_1+S_{11}v_2} - \frac{S_{22}}{S_{11}S_{22}-S_{12}S_{21}} = \frac{[B]}{[A]}\frac{S_{21}}{S_{11}S_{22}-S_{12}S_{21}} \quad (2.180)$$

$$\frac{1}{[B]}\frac{\partial[B]}{\partial S_{12}} = \frac{S_{21}}{S_{11}S_{22}-S_{12}S_{21}} \quad (2.181)$$

$$\frac{1}{[B]}\frac{\partial[B]}{\partial S_{21}} = -\frac{v_1}{-S_{21}v_1+S_{11}v_2} + \frac{S_{21}}{S_{11}S_{22}-S_{12}S_{21}} = -\frac{[B]}{[A]}\frac{S_{11}}{S_{11}S_{22}-S_{12}S_{21}} \quad (2.182)$$

$$\frac{1}{[B]}\frac{\partial[B]}{\partial S_{22}} = -\frac{S_{11}}{S_{11}S_{22}-S_{12}S_{21}} \quad (2.183)$$

$$(2.184)$$

E o erro associado às medidas pode ser aproximado por

$$\frac{1}{[A]}\delta_{[A]} = \left|\frac{1}{[A]}\frac{\partial[A]}{\partial S_{11}}\right|\delta_{S_{11}} + \left|\frac{1}{[A]}\frac{\partial[A]}{\partial S_{12}}\right|\delta_{S_{12}} + \left|\frac{1}{[A]}\frac{\partial[A]}{\partial S_{21}}\right|\delta_{S_{21}} + \left|\frac{1}{[A]}\frac{\partial[A]}{\partial S_{22}}\right|\delta_{S_{22}} \quad (2.185)$$

$$= \frac{1}{|\det S|}\left[S_{22}\delta_{S_{11}} + \frac{[A]}{[B]}S_{22}\delta_{S_{12}} + S_{12}\delta_{S_{21}} + \frac{[A]}{[B]}S_{12}\delta_{S_{22}}\right] \quad (2.186)$$

Analogamente, temos:

$$\frac{1}{[B]}\delta_{[B]} = \frac{1}{|\det S|}\left[\frac{[B]}{[A]}S_{21}\delta_{S_{11}} + S_{21}\delta_{S_{11}} + \frac{[B]}{[A]}S_{11}\delta_{S_{21}} + S_{11}\delta_{S_{22}}\right] \quad (2.187)$$

onde não se indicou $|S_{ij}|$ nem $|[.]|$ pois são todos positivos.

Fazemos agora a aplicação numérica:

Caso do par 1-2:

$$\det S = \begin{vmatrix} 270 & 30 \\ 140 & 20 \end{vmatrix} = 1200 \quad (2.188)$$

$$\frac{1}{[A]}\delta_{[A]} = \frac{1}{1200}\left[20\times270\times2\% + 20\times30\times2\% + 30\times140\times2\% + 30\times20\times2\%\right] \quad (2.189)$$

$$= \frac{216}{1200} = 0.18 = 18\% \quad (2.190)$$

$$\frac{1}{[B]}\delta_{[B]} = \frac{1}{1200}\left[140\times270\times2\% + 140\times30\times2\% + 270\times140\times2\% + 270\times20\times2\%\right] \quad (2.191)$$

$$= \frac{426}{1200} = 0.355 = 35.5\% \quad (2.192)$$

Caso do par 1-3:

$$\det S = \begin{vmatrix} 270 & 30 \\ 15 & 200 \end{vmatrix} = 53550 \quad (2.193)$$

$$\frac{1}{[A]}\delta_{[A]} = \frac{1}{53550}\left[200\times270\times2\% + 200\times30\times2\% + 30\times15\times10\% + 30\times200\times2\%\right] \quad (2.194)$$

$$= \frac{1804{,}6}{52550} \approx 0.0337 = 3.37\% \quad (2.195)$$

$$\frac{1}{[B]}\delta_{[B]} = \frac{1}{53550}\left[15\times270\times2\% + 15\times30\times2\% + 270\times15\times10\% + 270\times200\times2\%\right] \quad (2.196)$$

$$= \frac{5895}{53550} \approx 0.11 = 11\% \quad (2.197)$$

Conclusão, apesar de o sensor 3 apresentar uma incerteza cinco vezes maior na sensibilidade, a escolha do sensor 3 para fazer par ao sensor 1 parece mais adequada. ♢

Exercícios

E 2.9.1. Considere as expressões:

$$\frac{\exp(1/\mu)}{1+\exp(1/\mu)} \quad (2.198)$$

e

$$\frac{1}{\exp(-1/\mu)+1} \quad (2.199)$$

com $\mu > 0$. Verifique que elas são idênticas como funções reais. Teste no computador cada uma delas para $\mu = 0{,}1$, $\mu = 0{,}01$ e $\mu = 0{,}001$. Qual dessas expressões é mais adequada quando μ é um número pequeno? Por quê?

E 2.9.2. Encontre expressões alternativas para calcular o valor das seguintes funções quando x é próximo de zero.

a) $f(x) = \frac{1-\cos(x)}{x^2}$

b) $g(x) = \sqrt{1+x} - 1$

c) $h(x) = \sqrt{x+10^6} - 10^3$

d) $i(x) = \sqrt{1+e^x} - \sqrt{2}$ Dica: Faça $y = e^x - 1$

E 2.9.3. Use uma identidade trigonométrica adequada para mostrar que:

$$\frac{1-\cos(x)}{x^2} = \frac{1}{2}\left(\frac{\operatorname{sen}(x/2)}{x/2}\right)^2. \quad (2.200)$$

Analise o desempenho destas duas expressões no computador quando x vale 10^{-5}, 10^{-6}, 10^{-7}, 10^{-8}, 10^{-9}, 10^{-200} e 0. Discuta o resultado. **Dica:** Para $|x| < 10^{-5}$, $f(x)$ pode ser aproximada por $1/2 - x^2/24$ com erro de truncamento inferior a 10^{-22}.

E 2.9.4. Reescreva as expressões:

$$\sqrt{e^{2x}+1} - e^x \qquad \text{e} \qquad \sqrt{e^{2x}+x^2} - e^x \tag{2.201}$$

de modo que seja possível calcular seus valores para $x = 100$ utilizando a aritmética de ponto flutuante ("Double") no computador.

E 2.9.5. Na teoria da relatividade restrita, a energia cinética de uma partícula e sua velocidade se relacionam pela seguinte fórmula:

$$E = mc^2 \left(\frac{1}{\sqrt{1-(v/c)^2}} - 1 \right), \tag{2.205}$$

onde E é a energia cinética da partícula, m é a massa de repouso, v o módulo da velocidade e c a velocidade da luz no vácuo dada por $c = 299792458m/s$. Considere que a massa de repouso $m = 9{,}10938291 \times 10^{-31}Kg$ do elétron seja conhecida com erro relativo de 10^{-9}. Qual é o valor da energia e o erro relativo associado a essa grandeza quando $v = 0{,}1c$, $v = 0{,}5c$, $v = 0{,}99c$ e $v = 0{,}999c$ sendo que a incerteza relativa na medida da velocidade é 10^{-5}?

Capítulo 3

Solução de equações de uma variável

Neste capítulo, construiremos aproximações numéricas para a solução de **equações algébricas em uma única variável real**. Observamos que obter uma solução para uma dada equação é equivalente a encontrar um **zero de uma função real** apropriada. Com isso, iniciamos este capítulo discutindo condições de existência e unicidade de raízes de funções de uma variável real. Então, apresentamos o **método da bisseção** como uma primeira abordagem numérica para a solução de tais equações.

Em seguida, exploramos outra abordagem via **iteração do ponto fixo**. Desta, obtemos o **método de Newton**[1], para o qual estudamos aplicações e critérios de convergência. Por fim, apresentamos o **método das secantes** como uma das possíveis variações do método de Newton.

Ao longo do capítulo, apresentamos algumas computações com `Python`. Nestas, assumiremos o que os seguintes módulos estão carregados:

```
>>> from __future__ import division
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> import scipy as sci
>>> from scipy import optimize
```

A segunda instrução carrega a biblioteca de computação científica numpy e a terceira carrega a biblioteca gráfica matplotlib.

[1] Sir Isaac Newton, 1642 - 1727, matemático e físico inglês.

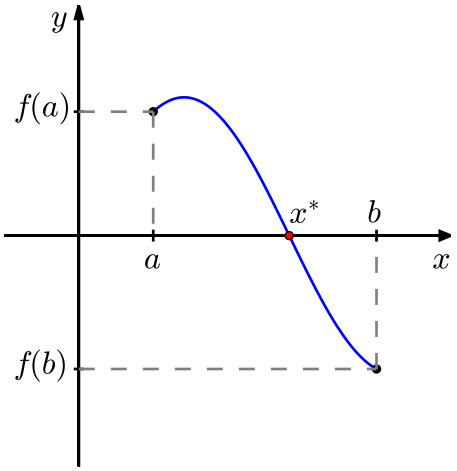


Figura 3.1: Teorema de Bolzano.

3.1 Existência e unicidade

O **teorema de Bolzano**[2] nos fornece condições suficientes para a existência do zero de uma função. Este é uma aplicação direta do **teorema do valor intermediário**.

Teorema 3.1.1 (Teorema de Bolzano)**.** *Se $f : [a, b] \to \mathbb{R}$, $y = f(x)$, é uma função contínua tal que $f(a) \cdot f(b) < 0$*[3]*, então existe $x^* \in (a, b)$ tal que $f(x^*) = 0$.*

Demonstração. O resultado é uma consequência imediata do teorema do valor intermediário que estabelece que dada uma função contínua $f : [a, b] \to \mathbb{R}$, $y = f(x)$, tal que $f(a) < f(b)$ (ou $f(b) < f(a)$), então para qualquer $d \in (f(a), f(b))$ (ou $k \in (f(b), f(a))$) existe $x^* \in (a, b)$ tal que $f(x^*) = k$. Ou seja, nestas notações, se $f(a) \cdot f(b) < 0$, então $f(a) < 0 < f(b)$ (ou $f(b) < 0 < f(a)$). Logo, tomando $k = 0$, temos que existe $x^* \in (a, b)$ tal que $f(x^*) = k = 0$. □

Em outras palavras, se $f(x)$ é uma função contínua em um dado intervalo no qual ela troca de sinal, então ela tem pelo menos um zero neste intervalo (veja a Figura 3.1).

Exemplo 3.1.1. Mostre que existe pelo menos uma solução da equação $e^x = x+2$ no intervalo $(-2,0)$.

[2] Bernhard Placidus Johann Gonzal Nepomuk Bolzano, 1781 - 1848, matemático do Reino da Boêmia.

[3] Esta condição é equivalente a dizer que a função troca de sinal no intervalo.

Solução. Primeiramente, observamos que resolver a equação $e^x = x + 2$ é equivalente a resolver $f(x) = 0$ com $f(x) = e^x - x - 2$. Agora, como $f(-2) = e^{-2} > 0$ e $f(0) = -2 < 0$, temos do teorema de Bolzano que existe pelo menos um zero de $f(x)$ no intervalo $(-2, 0)$. E, portanto, existe pelo menos uma solução da equação dada no intervalo $(-2, 0)$.

Podemos usar `Python` para estudarmos esta função. Por exemplo, podemos definir a função $f(x)$ e computá-la nos extremos do intervalo dado com os seguintes comandos:

```
>>> def f(x): return np.exp(x)-x-2
...
>>> f(-2),f(0)
(0.13533528323661281, -1.0)
```

Alternativamente (e com maior precisão), podemos verificar diretamente o sinal da função nos pontos desejados com a função numpy.sign:

```
>>> np.sign(f(-2)*f(0))
-1.0
```

♢

Quando procuramos aproximações para zeros de funções, é aconselhável isolar cada raiz em um intervalo. Desta forma, gostaríamos de poder garantir a existência e a unicidade da raiz dentro de um dado intervalo. A seguinte proposição nos fornece condições suficientes para tanto.

Proposição 3.1.1. *Se $f : [a, b] \to \mathbb{R}$ é um função diferenciável, $f(a) \cdot f(b) < 0$ e $f'(x) > 0$ (ou $f'(x) < 0$) para todo $x \in (a, b)$, então existe um único $x^* \in (a, b)$ tal que $f(x^*) = 0$.*

Em outras palavras, para garantirmos que exista um único zero de uma dada função diferenciável em um intervalo, é suficiente que ela troque de sinal e seja monótona neste intervalo.

Exemplo 3.1.2. No Exemplo 3.1.1, mostramos que existe pelo menos um zero de $f(x) = e^x - x - 2$ no intervalo $(-2,0)$, pois $f(x)$ é contínua e $f(-2) \cdot f(0) < 0$. Agora, observamos que, além disso, $f'(x) = e^x - 1$ e, portanto, $f'(x) < 0$ para todo $x \in (-2,0)$. Logo, da Proposição 3.1.1, temos garantida a existência de um único zero no intervalo dado.

Podemos inspecionar o comportamento da função $f(x) = e^x - x - 2$ e de sua derivada fazendo seus gráficos no Python. Para tanto, podemos usar o seguinte código `Python`:

```
>>> def f(x): return np.exp(x)-x-2
...
>>> xx = np.linspace(-2,0)
>>> plt.plot(xx,f(xx))
>>> plt.grid(); plt.show()

>>> def fl(x): return np.exp(x)-1
...
>>> plt.plot(xx,fl(xx))
>>> plt.grid(); plt.show()
```

A discussão feita nesta seção, especialmente o teorema de Bolzano, nos fornece os fundamentos para o método da bisseção, o qual discutimos na próxima seção.

Exercícios

E 3.1.1. Mostre que $\cos x = x$ tem solução no intervalo $[0, \pi/2]$.

E 3.1.2. Mostre que $\cos x = x$ tem uma única solução no intervalo $[0, \pi/2]$.

E 3.1.3. Interprete a equação $\cos(x) = kx$ como o problema de encontrar a intersecção da curva $y = \cos(x)$ com $y = kx$. Encontre o valor positivo k para o qual essa equação admite exatamente duas raízes positivas distintas.

E 3.1.4. Mostre que a equação:

$$\ln(x) + x^3 - \frac{1}{x} = 10 \tag{3.1}$$

possui uma única solução positiva.

E 3.1.5. Use o teorema de Bolzano para mostrar que o erro absoluto ao aproximar o zero da função $f(x) = e^x - x - 2$ por $\overline{x} = -1{,}841$ é menor que 10^{-3}.

E 3.1.6. Mostre que o erro absoluto associado à aproximação $\overline{x} = 1{,}962$ para a solução exata x^* de:

$$e^x + \operatorname{sen}(x) + x = 10 \tag{3.2}$$

é menor que 10^{-4}.

E 3.1.7. Mostre que a equação

$$\ln(x) + x - \frac{1}{x} = v \tag{3.3}$$

possui uma solução para cada v real e que esta solução é única.

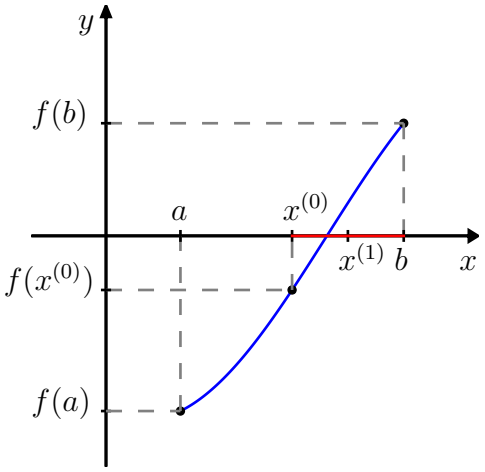


Figura 3.2: Método da bisseção.

3.2 Método da bisseção

O **método da bisseção** explora o fato de que uma função contínua $f : [a, b] \to \mathbb{R}$ com $f(a) \cdot f(b) < 0$ tem um zero no intervalo (a, b) (veja o teorema de Bolzano 3.1.1). Assim, a ideia para aproximar o zero de uma tal função $f(x)$ é tomar, como primeira aproximação, o ponto médio do intervalo $[a, b]$, isto é:

$$x^{(0)} = \frac{(a+b)}{2}. \tag{3.4}$$

Pode ocorrer de $f(x^{(0)}) = 0$ e, neste caso, o zero de $f(x)$ é $x^* = x^{(0)}$. Caso contrário, se $f(a) \cdot f(x^{(0)}) < 0$, então $x^* \in (a, x^{(0)})$. Neste caso, tomamos como segunda aproximação do zero de $f(x)$ o ponto médio do intervalo $[a, x^{(0)}]$, isto é, $x^{(1)} = (a + x^{(0)})/2$. No outro caso, temos $f(x^{(0)}) \cdot f(b) < 0$ e, então, tomamos $x^{(1)} = (x^{(0)} + b)/2$. Repetimos este procedimento até obtermos a aproximação desejada (veja Figura 3.2).

De forma mais precisa, suponha que queiramos calcular uma aproximação com uma certa precisão TOL para um zero x^* de uma dada função contínua $f : [a, b] \to \mathbb{R}$ tal que $f(a) \cdot f(b) < 0$. Iniciamos, tomando $n = 0$ e:

$$a^{(n)} = a, \quad b^{(n)} = b \quad \text{e} \quad x^{(n)} = \frac{a^{(n)} + b^{(n)}}{2}. \tag{3.5}$$

Verificamos o **critério de parada**, isto é, se $f(x^{(n)}) = 0$ ou:

$$\frac{|b^{(n)} - a^{(n)}|}{2} < TOL, \tag{3.6}$$

Tabela 3.1: Iteração do método da bisseção para o Exemplo 3.2.1.

n	$a^{(n)}$	$b^{(n)}$	$x^{(n)}$	$f(a^{(n)})f(x^{(n)})$	$\frac{\lvert b^{(n)} - a^{(n)}\rvert}{2}$
0	-2	0	-1	< 0	1
1	-2	-1	$-1{,}5$	< 0	0,5
2	-2	$-1{,}5$	$-1{,}75$	< 0	0,25
3	-2	$-1{,}75$	$-1{,}875$	> 0	0,125
4	$-1{,}875$	$-1{,}75$	$-1{,}8125$	< 0	0,0625

então $x^{(n)}$ é a aproximação desejada. Caso contrário, preparamos a próxima iteração $n+1$ da seguinte forma: se $f(a^{(n)}) \cdot f(x^{(n)}) < 0$, então definimos $a^{(n+1)} = a^{(n)}$ e $b^{(n+1)} = x^{(n)}$; no outro caso, se $f(x^{(n)}) \cdot f(b^{(n)}) < 0$, então definimos $a^{(n+1)} = x^{(n)}$ e $b^{(n+1)} = b^{(n)}$. Trocando n por $n+1$, temos a nova aproximação do zero de $f(x)$ dada por:

$$x^{(n+1)} = \frac{a^{(n+1)} + b^{(n+1)}}{2}. \tag{3.7}$$

Voltamos a verificar o critério de parada acima e, caso não satisfeito, iteramos novamente. Iteramos até obtermos a aproximação desejada ou o número máximo de iterações ter sido atingido.

Exemplo 3.2.1. Use o método da bisseção para calcular uma solução de $e^x = x+2$ no intervalo $[-2, 0]$ com precisão $TOL = 10^{-1}$.

Solução. Primeiramente, observamos que resolver a equação dada é equivalente a calcular o zero de $f(x) = e^x - x - 2$. Além disso, temos $f(-2) \cdot f(0) < 0$. Desta forma, podemos iniciar o método da bisseção tomando o intervalo inicial $[a^{(0)}, b^{(0)}] = [-2, 0]$ e:

$$x^{(0)} = \frac{a^{(0)} + b^{(0)}}{2} = -1. \tag{3.8}$$

Apresentamos as iterações na Tabela 3.1. Observamos que a precisão $TOL = 10^{-1}$ foi obtida na quarta iteração com o zero de $f(x)$ sendo aproximado por $x^{(4)} = -1{,}8125$.

Usando `Python` neste exemplos, temos:

```
>>> def f(x): return np.exp(x) - x - 2
...
>>> a=-2; b=0; x = (a+b)/2; [a,b,x]
[-2, 0, -1.0]
```

```
>>> [(b-a)/2, np.sign(f(a)*f(x))]
[1.0, -1.0]
>>> b=x; x=(a+b)/2; [a,b,x]
[-2, -1.0, -1.5]
>>> [(b-a)/2, np.sign(f(a)*f(x))]
```

e, assim, sucessivamente. Veja o código completo na Seção 3.2.1. ◊

Vamos agora discutir sobre a **convergência** do método da bisseção, que é garantida pelo Teorema 3.2.1.

Teorema 3.2.1 (Convergência do método da bisseção)**.** *Sejam $f : [a, b] \to \mathbb{R}$ uma função contínua tal que $f(a) \cdot f(b) < 0$ e x^* o único zero de $f(x)$ no intervalo (a, b). Então, a sequência $\{x^{(n)}\}_{n\geq 0}$ do método da bisseção satisfaz:*

$$|x^{(n)} - x^*| < \frac{b-a}{2^{n+1}}, \quad \forall n \geq 0, \tag{3.9}$$

isto é, $x^{(n)} \to x^*$ quando $n \to \infty$.

Demonstração. Notemos que, a cada iteração, a distância entre a aproximação $x^{(n)}$ e o zero x^* da função é menor ou igual que a metade do tamanho do intervalo $[a^{(n)}, b^{(n)}]$ (veja Figura 3.2), isto é:

$$|x^{(n)} - x^*| \leq \frac{b^{(n)} - a^{(n)}}{2}. \tag{3.10}$$

Por construção do método, temos $[a^{(n)}, b^{(n)}] \subset [a^{(n-1)}, b^{(n-1)}]$ e:

$$b^{(n)} - a^{(n)} = \frac{b^{(n-1)} - a^{(n-1)}}{2}. \tag{3.11}$$

Desta forma:

$$|x^{(n)} - x^*| \leq \frac{b^{(n)} - a^{(n)}}{2} = \frac{b^{(n-1)} - a^{(n-1)}}{2^2} = \cdots = \frac{b^{(0)} - a^{(0)}}{2^{n+1}}, \quad \forall n \geq 1. \tag{3.12}$$

Logo, vemos que:

$$|x^{(n)} - x^*| \leq \frac{b-a}{2^{n+1}}, \quad \forall n \geq 0. \tag{3.13}$$

□

Observamos que a hipótese de que $f(x)$ tenha um único zero no intervalo não é realmente necessária. Se a função tiver mais de um zero no intervalo inicial, as iterações ainda convergem para um dos zeros. Veja o Exercício 3.2.3.

Observação 3.2.1. O Teorema 3.2.1 nos fornece uma estimativa para a convergência do método da bisseção. Aproximadamente, temos:

$$|x^{(n+1)} - x^*| \lesssim \frac{1}{2}|x^{(n)} - x^*|. \tag{3.14}$$

Isto nos leva a concluir que o método da bisseção tem **taxa de convergência** linear.

Exemplo 3.2.2. No Exemplo 3.2.1, precisamos de 4 iterações do método da bisseção para computar uma aproximação com precisão de 10^{-1} do zero de $f(x) = e^x - x - 2$ tomando como intervalo inicial $[a, b] = [-2, 0]$. Poderíamos ter estimado o número de iterações **a priori**, pois, como vimos acima:

$$|x^{(n)} - x^*| \leq \frac{b-a}{2^{n+1}}, \quad n \geq 0. \tag{3.15}$$

Logo, temos:

$$|x^{(n)} - x^*| < \frac{b-a}{2^{n+1}} = \frac{2}{2^{n+1}} \tag{3.16}$$

$$= 2^{-n} < 10^{-1} \Rightarrow n > -\log_2 10^{-1} \approx 3{,}32. \tag{3.17}$$

O que está de acordo com o experimento numérico realizado naquele exemplo.

O método da bisseção tem a boa propriedade de garantia de convergência, bem como de fornecer uma simples estimativa do erro na aproximação calculada. Entretanto, a taxa de convergência linear é superada por outros métodos. A construção de tais métodos está, normalmente, associada à iteração do ponto fixo, a qual exploramos na próxima seção.

3.2.1 Código Python: método da bisseção

O seguinte código é uma implementação em `Python` do algoritmo da bisseção. As variáveis de entrada são:

- `f` - função objetivo
- `a` - extremo esquerdo do intervalo de inspeção $[a, b]$
- `b` - extremo direito do intervalo de inspeção $[a, b]$
- `TOL` - tolerância (critério de parada)
- `N` - número máximo de iterações

A variável de saída é:

- `p` - aproximação da raiz de `f`, isto é, $f(p) \approx 0$.

```
from __future__ import division

def bissecao(f, a, b, TOL, N):
    i = 1
    fa = f(a)
    while (i <= N):
        #iteracao da bissecao
        p = a + (b-a)/2
        fp = f(p)
        #condicao de parada
        if ((fp == 0) or ((b-a)/2 < TOL)):
            return p
        #bissecta o intervalo
        i = i+1
        if (fa * fp > 0):
            a = p
            fa = fp
        else:
            b = p

    raise NameError('Num. max. de iter. excedido!');
```

Exercícios

E 3.2.1. Considere a equação $\sqrt{x} = \cos(x)$. Use o método da bisseção com intervalo inicial $[a, b] = [0, 1]$ e $x^{(1)} = (a + b)/2$ para calcular a aproximação $x^{(4)}$ da solução desta equação.

E 3.2.2. Trace o gráfico e isole as três primeiras raízes positivas da função:

$$f(x) = 5\,\mathrm{sen}\,(x^2) - \exp\left(\frac{x}{10}\right) \tag{3.18}$$

em intervalos de comprimento 0,1. Então, use o método da bisseção para obter aproximações dos zeros desta função com precisão de 10^{-5}.

E 3.2.3. O polinômio $p(x) = -4+8x-5x^2+x^3$ tem raízes $x_1 = 1$ e $x_2 = x_3 = 2$ no intervalo $[1/2, 3]$.

a) Se o método da bisseção for usando com o intervalo inicial $[1/2, 3]$, para qual raiz as iterações convergem?

b) É possível usar o método da bisseção para a raiz $x = 2$? Justifique sua resposta.

E 3.2.4. O polinômio $f(x) = x^4 - 4x^2 + 4$ possui raízes duplas em $\sqrt{2}$ e $-\sqrt{2}$. O método da bisseção pode ser aplicado a f? Explique.

E 3.2.5. Mostre que a equação do Problema 3.1.7 possui uma solução no intervalo $[1, v+1]$ para todo v positivo. Dica: defina $f(x) = \ln(x) + x - \frac{1}{x} - v$ e considere a seguinte estimativa:

$$f(v+1) = f(1) + \int_1^{v+1} f'(x)dx \geq -v + \int_1^{v+1} dx = 0. \tag{3.19}$$

Use esta estimativa para iniciar o método de bisseção e obtenha o valor da raiz com pelo menos 6 algarismos significativos para $v = 1, 2, 3, 4$ e 5.

E 3.2.6. (Estática) Considere o seguinte problema físico: uma plataforma está fixa a uma parede através de uma dobradiça cujo momento é dado por:

$$\tau = k\theta, \tag{3.20}$$

onde θ é angulo da plataforma com a horizontal e k é uma constante positiva. A plataforma é feita de material homogêneo, seu peso é P e sua largura é l. Modele a relação entre o ângulo θ e o peso P próprio da plataforma. Encontre o valor de θ quando $l = 1$ m, $P = 200$ N, $k = 50$ Nm/rad, sabendo que o sistema está em equilíbrio. Use o método da bisseção e expresse o resultado com 4 algarismos significativos.

E 3.2.7. Considere a equação de Lambert dada por:

$$xe^x = t, \tag{3.21}$$

onde t é um número real positivo. Mostre que esta equação possui uma única solução x^* que pertence ao intervalo $[0, t]$. Usando esta estimativa como intervalo inicial, quantos passos são necessário para obter o valor numérico de x^* com erro absoluto inferior a 10^{-6} quando $t = 1$, $t = 10$ e $t = 100$ através do método da bisseção? Obtenha esses valores.

E 3.2.8. (Eletrônica) O desenho abaixo mostra um circuito não linear envolvendo uma fonte de tensão constante, um diodo retificador e um resistor. Sabendo

que a relação entre a corrente (I_d) e a tensão (v_d) no diodo é dada pela seguinte expressão:

$$I_d = I_R \left(\exp\left(\frac{v_d}{v_t}\right) - 1 \right), \tag{3.22}$$

onde I_R é a corrente de condução reversa e v_t, a tensão térmica dada por $v_t = \frac{kT}{q}$ com k, a constante de Boltzmann, T a temperatura de operação e q, a carga do elétron. Aqui $I_R = 1pA = 10^{-12}$ A, $T = 300$ K. Escreva o problema como uma equação na incógnita v_d e, usando o método da bisseção, resolva este problema com 3 algarismos significativos para os seguintes casos:

a) $V = 30$ V e $R = 1$ kΩ.

b) $V = 3$ V e $R = 1$ kΩ.

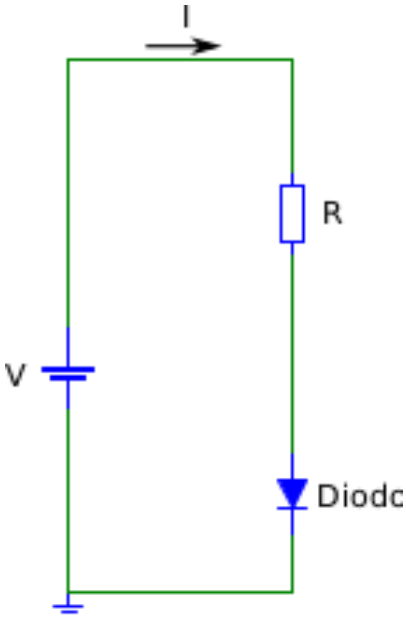
c) $V = 3$ V e $R = 10$ kΩ.

d) $V = 300$ mV e $R = 1$ kΩ.

e) $V = -300$ mV e $R = 1$ kΩ.

f) $V = -30$ V e $R = 1$ kΩ.

g) $V = -30$ V e $R = 10$ kΩ.



Dica: $V = RI_d + v_d$.

E 3.2.9. (Propagação de erros) Obtenha os valores de I_d no Problema 3.2.8. Lembre que existem duas expressões disponíveis:

$$I_d = I_R \left(\exp\left(\frac{v_d}{v_t}\right) - 1 \right) \tag{3.23}$$

e

$$I_d = \frac{v - v_d}{R} \tag{3.24}$$

Faça o estudo da propagação do erro e decida qual a melhor expressão em cada caso.

3.3 Iteração de ponto fixo

Nesta seção, discutimos a abordagem da **iteração do ponto fixo** para a solução numérica de equações de uma variável real. Observamos que sempre podemos

reescrever uma equação da forma $f(x) = 0$ (problema de encontrar os zeros de uma função) em uma equação equivalente na forma $g(x) = x$ (**problema de ponto fixo**). Um ponto $x = x^*$ tal que $g(x^*) = x^*$ é chamado de **ponto fixo** da função $g(x)$. Geometricamente, um ponto fixo de uma função é um ponto de interseção entre a reta $y = x$ com o gráfico da função $g(x)$ (veja Figura 3.3).

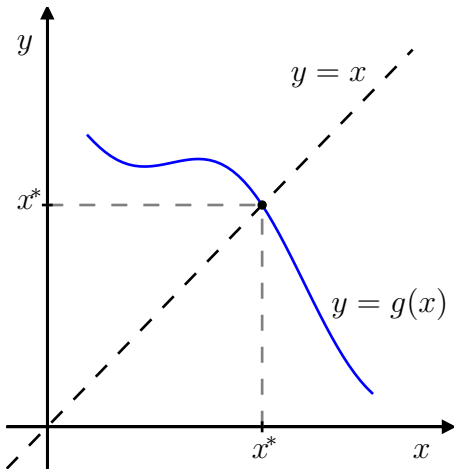


Figura 3.3: Ponto fixo $g(x^*) = x^*$.

Exemplo 3.3.1. Resolver a equação $e^x = x + 2$ é equivalente a resolver $f(x) = 0$, com $f(x) = e^x - x - 2$. Estes são equivalentes a resolver $g(x) = x$, com $g(x) = e^x - 2$, isto é:

$$e^x = x + 2 \Leftrightarrow e^x - x - 2 = 0 \Leftrightarrow e^x - 2 = x \tag{3.25}$$

Dada uma função $g(x)$, a **iteração do ponto fixo** consiste em computar a seguinte sequência recursiva:

$$x^{(n+1)} = g(x^{(n)}), \quad n \geq 1, \tag{3.26}$$

onde $x^{(1)}$ é uma aproximação inicial do ponto fixo.

Exemplo 3.3.2 (Método babilônico)**.** O método babilônico[4] é de uma iteração de ponto fixo para extrair a raiz quadrada de um número positivo A, isto é, resolver a equação $x^2 = A$.

Seja $r > 0$ uma aproximação para $\sqrt{A}$. Temos três possibilidades:

- $r > \sqrt{A} \Longrightarrow \frac{A}{r} < \sqrt{A} \Longrightarrow \sqrt{A} \in \left(\frac{A}{r}, r\right)$;

[4]Heron de Alexandria, 10 d.C. - 70 d.C., matemático grego.

- $r = \sqrt{A} \Longrightarrow \frac{A}{r} = \sqrt{A}$;
- $r < \sqrt{A} \Longrightarrow \frac{A}{r} > \sqrt{A} \Longrightarrow \sqrt{A} \in \left(r, \frac{A}{r}\right)$.

Ou seja, $\sqrt{A}$ sempre está no intervalo entre r e $\frac{A}{r}$, no qual podemos buscar uma nova aproximação como, por exemplo, pelo ponto médio:

$$x = \frac{r + \frac{A}{r}}{2}. \tag{3.27}$$

Aplicando esse método repetidas vezes, podemos construir a iteração (de ponto fixo):

$$x^{(1)} = r \tag{3.28}$$

$$x^{(n+1)} = \frac{x^{(n)}}{2} + \frac{A}{2x^{(n)}}, \quad n = 1,2,3,... \tag{3.29}$$

Por exemplo, para obter uma aproximação para $\sqrt{5}$, podemos iniciar com a aproximação inicial $r = 2$ e $A = 5$. Então, tomamos $x^{(1)} = 2$ e daí seguem as aproximações:

$$x^{(2)} = \frac{2}{2} + \frac{2{,}5}{2} = 2{,}25 \tag{3.30}$$

$$x^{(3)} = \frac{2{,}25}{2} + \frac{2{,}5}{2{,}25} = 2{,}2361111 \tag{3.31}$$

$$x^{(4)} = \frac{2{,}2361111}{2} + \frac{2{,}5}{2{,}2361111} = 2{,}236068 \tag{3.32}$$

$$x^{(5)} = \frac{2{,}236068}{2} + \frac{2{,}5}{2{,}236068} = 2{,}236068 \tag{3.33}$$

O método babilônico sugere que a iteração do ponto fixo pode ser uma abordagem eficiente para a solução de equações. Ficam, entretanto, as seguintes perguntas:

1. Será que a iteração do ponto fixo é convergente?
2. Caso seja convergente, será que o limite da sequência produzida, isto é, $x^* := \lim_{n\to\infty} x^{(n)}$ é um ponto fixo?
3. Caso seja convergente, qual é a taxa de convergência?

A segunda pergunta é a mais fácil de ser respondida. No caso de $g(x)$ ser contínua, se $x^{(n)} \to x^* \in \mathrm{Dom}\,(g)$, então:

$$x^* = \lim_{n\to\infty} x^{(n)} = \lim_{n\to\infty} g(x^{(n-1)}) = g\left(\lim_{n\to\infty} x^{(n-1)}\right) = g(x^*). \tag{3.34}$$

Antes de respondermos as outras perguntas acima, vejamos mais um exemplo.

Tabela 3.2: Iterações do ponto fixo para o Exemplo 3.3.3.

n	$x_1^{(n)}$	$x_2^{(n)}$
1	1,700	1,700
2	2,047	1,735
3	−0,8812	1,743
4	4,3013	1,746
5	−149,4	1,746

Exemplo 3.3.3. Considere o problema de encontrar o zero da função $f(x) = xe^x - 10$. Uma maneira geral de construir um problema de ponto fixo equivalente é o seguinte:

$$f(x) = 0 \Rightarrow \alpha f(x) = 0 \Rightarrow x - \alpha f(x) = x, \tag{3.35}$$

para qualquer parâmetro $\alpha \neq 0$. Consideremos, então, as seguintes duas funções:

$$g_1(x) = x - 0{,}5f(x) \quad \text{e} \quad g_2(x) = x - 0{,}05f(x). \tag{3.36}$$

Notamos que o ponto fixo destas duas funções coincide com o zero de $f(x)$. Construindo as iterações do ponto fixo:

$$x_1^{(n+1)} = g_1(x_1^{(n)}) \quad \text{e} \quad x_2^{(n+1)} = g_2(x_2^{(n)}), \tag{3.37}$$

tomando $x_1^{(1)} = x_2^{(1)} = 1{,}7$, obtemos os resultados apresentados na Tabela 3.2. Observamos que, enquanto, a iteração do ponto fixo com a função $g_1(x)$ ($\alpha = 0{,}5$) parece divergir, a iteração com a função $g_2(x)$ ($\alpha = 0{,}05$) parece convergir.

Em `Python`, podemos computar as iterações do ponto fixo $x^{(n+1)} = g_1(x^{(n)})$ com o seguinte código:

```
>>> def f(x): return x*np.exp(x)-10
...
>>> def g1(x): return x-0.5*f(x)
...
>>> x=1.7
>>> x=g1(x);x
2.0471447170318804
>>> x=g1(x);x
-0.88119413893725618
```

e, assim, sucessivamente. Itere com a função $g_2(x)$ e verifique a convergência!

A fim de estudarmos a convergência da iteração do ponto fixo, apresentamos o teorema do ponto fixo.

3.3.1 Teorema do ponto fixo

O teorema do ponto fixo nos fornece condições suficientes para a existência e unicidade do ponto fixo, bem como para a convergência das iterações do método.

Definição 3.3.1. *Uma **contração** é uma função real $g : [a, b] \to [a, b]$ tal que:*

$$|g(x) - g(y)| \leq \beta|x - y|, \quad 0 \leq \beta < 1. \tag{3.38}$$

Observação 3.3.1. Seja $g : [a, b] \to [a, b]$, y=g(x).

- Se $g(x)$ é uma contração, então $g(x)$ função contínua.
- Se $|g'(x)| < k$, $0 < k < 1$, para todo $x \in [a, b]$, então $g(x)$ é uma contração.

Teorema 3.3.1 (Teorema do ponto fixo)**.** *Se $g : [a,b] \to [a,b]$ é uma contração, então existe um único ponto $x^* \in [a, b]$ tal que $g(x^*) = x^*$, isto é, x^* é ponto fixo de $g(x)$. Além disso, a sequência $\{x^{(n)}\}_{n\in\mathbb{N}}$ dada por:*

$$x^{(n+1)} = g(x^{(n)}) \tag{3.39}$$

converge para x^* para qualquer $x^{(1)} \in [a, b]$.

Demonstração. Começamos demonstrando que existe pelo menos um ponto fixo. Para tal definimos a função $f(x) = x - g(x)$ e observamos que:

$$f(a) = a - g(a) \leq a - a = 0 \tag{3.40}$$

e

$$f(b) = b - g(b) \geq b - b = 0 \tag{3.41}$$

Se $f(a) = a$ ou $f(b) = b$, então o ponto fixo existe. Caso contrário, as desigualdades são estritas e a $f(x)$ muda de sinal no intervalo. Como esta função é contínua, pelo teorema de Bolzano 3.1.1, existe um ponto x^* no intervalo (a, b) tal que $f(x^*) = 0$, ou seja, $g(x^*) = x^*$. Isto mostra a existência.

Para provar que o ponto fixo é único, observamos que se x^* e x^{**} são pontos fixos, eles devem ser iguais, pois:

$$|x^* - x^{**}| = |g(x^*) - g(x^{**})| \leq \beta|x^* - x^{**}|. \tag{3.42}$$

A desigualdade $|x^* - x^{**}| \leq \beta|x^* - x^{**}|$ com $0 \leq \beta < 1$ implica $|x^* - x^{**}| = 0$.

Para demonstrar a convergência da sequência, observamos que:

$$|x^{(n+1)} - x^*| = |g(x^{(n)}) - x^*| = |g(x^{(n)}) - g(x^*)| \leq \beta|x^{(n)} - x^*|. \tag{3.43}$$

Daí, temos:

$$|x^{(n)} - x^*| \leq \beta|x^{(n-1)} - x^*| \leq \beta^2|x^{(n-2)} - x^*| \leq \cdots \leq \beta^n|x^{(0)} - x^*|. \tag{3.44}$$

Portanto, como $0 \leq \beta < 1$, temos:

$$\lim_{n\to\infty} |x^{(n)} - x^*| = 0, \tag{3.45}$$

ou seja, $x^{(n)} \to x^*$ quando $n \to \infty$. □

Observação 3.3.2. Do teorema do ponto fixo, temos que se $g(x)$ é uma contração com constante $0 \leq \beta < 1$, então:

$$|x^{(n+1)} - x^*| \leq \beta|x^{(n)} - x^*|, \quad n \geq 1. \tag{3.46}$$

Isto é, as iterações do ponto fixo têm taxa de convergência linear.

Exemplo 3.3.4. Mostre que o teorema do ponto fixo se aplica a função $g(x) = \cos(x)$ no intervalo $[1/2, 1]$, isto é, a iteração de ponto fixo converge para a solução da equação $\cos x = x$. Então, calcule as iterações do ponto fixo com aproximação inicial $x^{(1)} = 0{,}7$, estime o erro absoluto da aproximação e verifique a taxa de convergência.

Solução. Basta mostrarmos que:

a) $g([1/2,1]) \subseteq [1/2,1]$;

b) $|g'(x)| < \beta, \quad 0 < \beta < 1, \quad \forall x \in [1/2,1]$.

Para provar a), observamos que $g(x)$ é decrescente no intervalo, pelo que temos:

$$0{,}54 < \cos(1) \leq \cos(x) \leq \cos(1/2) < 0{,}88 \tag{3.47}$$

Como [0,54, 0,88] $\subseteq$ [0,5, 1], temos o item a).

Para provar o item b), observamos que:

$$g'(x) = -\operatorname{sen}(x). \tag{3.48}$$

Da mesma forma, temos a estimativa:

$$-0{,}85 < -\operatorname{sen}(1) \leq -\operatorname{sen}(x) \leq -\operatorname{sen}(1/2) < -0{,}47. \tag{3.49}$$

Assim, $|g'(x)| < 0{,}85$ e temos a desigualdade com $\beta = 0{,}85 < 1$.

A Tabela 3.3 apresenta o comportamento numérico da iteração do ponto fixo:

$$x^{(1)} = 0{,}7 \tag{3.50}$$

$$x^{(n+1)} = \cos(x^{(n)}), \quad n \geq 1. \tag{3.51}$$

n	$x^{(n)}$	$\epsilon_n := \lvert x^{(n)} - x^* \rvert$
1	0,70000	3,9E−02
2	0,76484	2,6E−02
3	0,72149	1,8E−02
4	0,75082	1,2E−02
5	0,73113	8,0E−03
6	0,74442	5,3E−03
7	0,73548	3,6E−03

Tabela 3.3: Iteração do ponto fixo para o Exemplo 3.3.4.

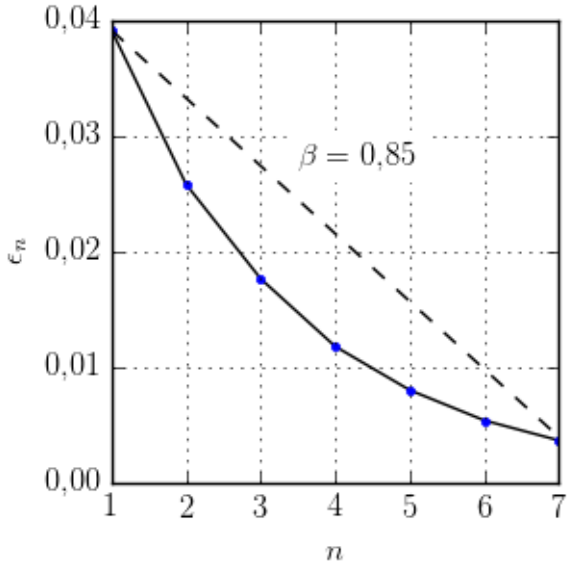


Figura 3.4: Decaimento do erro $\epsilon_n = |x^{(n)} - x^*|$ da iteração do ponto fixo estudada no Exemplo 3.3.4.

Para estimar o erro, consideramos $x^* = 0{,}7390851605$. A Figura 3.4 mostrar o decaimento do erro $\epsilon_n = |x^{(n)} - x^*|$ comparado com a taxa de convergência linear com $\beta = 0{,}85$.

Em `Python`, podemos computar estas iterações, o erro absoluto com o seguinte código:

```
#funcao do pto. fixo
def g(x):
    return np.cos(x)

#est. da solucao
xe = sci.optimize.fixed_point(g, 0.7)

#aprox. inicial
x0 = 0.7
eps = np.fabs(x0-xe)
print("x1 =  %.5f, eps =~ %.1e" % (x0, eps))

for i in np.arange(7):
    x = g(x0);
    eps = np.fabs(x-xe);
    print("%s =~ %.5f, eps =~ %.1e" % (('x'+str(i+2)), x, eps))
    x0 = x
```

◊

3.3.2 Teste de convergência

Seja $g : [a,b] \to \mathbb{R}$ uma função $C^0[a,b]$ e $x^* \in (a,b)$ um ponto fixo de g. Então x^* é dito estável se existe uma região $(x^* - \delta, x^* + \delta)$ chamada bacia de atração tal que $x^{(n+1)} = g(x^{(n)})$ é convergente sempre que $x^{(0)} \in (x^* - \delta, x^* + \delta)$.

Proposição 3.3.1 (Teste de convergência)**.** *Se $g \in C^1[a,b]$ e $|g'(x^*)| < 1$, então x^* é estável. Se $|g'(x^*)| > 1$ é instável e o teste é inconclusivo quando $|g'(x^*)| = 1$.*

Exemplo 3.3.5. No Exemplo 3.3.3, observamos que a função $g_1(x)$ nos forneceu uma iteração divergente, enquanto que a função $g_2(x)$ forneceu uma iteração convergente (veja a Figura 3.5. Estes comportamentos são explicados pelo teste da convergência. Com efeito, sabemos que o ponto fixo destas funções está no intervalo $[1{,}6, 1{,}8]$ e temos:

$$|g_1'(x)| = |1 - 0{,}5(x+1)e^x| > 4{,}8, \quad \forall x \in [1{,}6, 1{,}8], \tag{3.52}$$

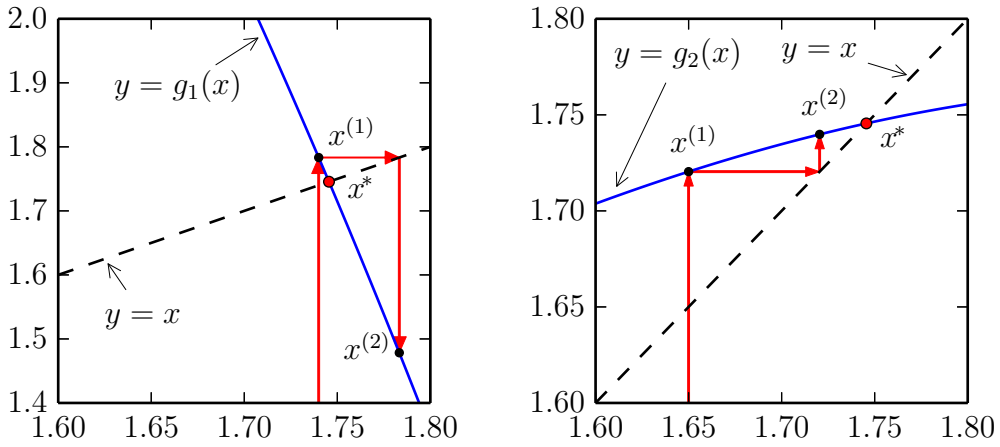


Figura 3.5: Ilustração das iterações do ponto fixo para: (esquerda) $y = g_1(x)$ e (direita) $y = g_2(x)$. Veja Exemplo 3.3.5.

enquanto:

$$|g_2'(x)| = |1 - 0{,}05(x+1)e^x| < 0{,}962, \quad \forall x \in [1{,}6, 1{,}8]. \tag{3.53}$$

3.3.3 Estabilidade e convergência

A fim de compreendermos melhor os conceitos de estabilidade e convergência, considere uma função $\Phi(x)$ com um ponto fixo $x^* = g(x^*)$ e analisemos o seguinte processo iterativo:

$$x^{(n+1)} = g\left(x^{(n)}\right) \tag{3.54}$$

$$x^{(0)} = x \tag{3.55}$$

Vamos supor que a função $g(x)$ pode ser aproximada por seu polinômio de Taylor em torno do ponto fixo:

$$g(x) = g(x^*) + (x - x^*)g'(x^*) + O\left((x - x^*)^2\right), n \geq 0 \tag{3.56}$$

$$= x^* + (x - x^*)g'(x^*) + O\left((x - x^*)^2\right) \tag{3.57}$$

$$\approx x^* + (x - x^*)g'(x^*) \tag{3.58}$$

Substituindo na relação de recorrência, temos

$$x^{(n+1)} = g\left(x^{(n)}\right) \approx x^* + (x^{(n)} - x^*)g'(x^*) \tag{3.59}$$

Ou seja:

$$\left(x^{(n+1)} - x^*\right) \approx (x^{(n)} - x^*)g'(x^*) \tag{3.60}$$

Tomando módulos, temos:

$$\underbrace{\left|x^{(n+1)} - x^*\right|}_{\epsilon_{n+1}} \approx \underbrace{\left|x^{(n)} - x^*\right|}_{\epsilon_n} \left|g'(x^*)\right|, \tag{3.61}$$

onde $\epsilon_n = \left|x^{(n)} - x^*\right|$.

Observação 3.3.3. Da análise acima, concluímos:

- Se $|g'(x^*)| < 1$, então, a distância de $x^{(n)}$ até o ponto fixo x^* está diminuindo a cada passo.
- Se $|g'(x^*)| > 1$, então, a distância de $x^{(n)}$ até o ponto fixo x^* está aumentando a cada passo.
- Se $|g'(x^*)| = 1$, então, nossa aproximação de primeira ordem não é suficiente para compreender o comportamento da sequência.

3.3.4 Erro absoluto e tolerância

Na prática, quando se aplica uma iteração como esta, não se conhece de antemão o valor do ponto fixo x^*. Assim, o erro $\epsilon_n = \left|x^{(n)} - x^*\right|$ precisa ser estimado com base nos valores calculados $x^{(n)}$. Uma abordagem frequente é analisar a evolução da diferença entre dois elementos da sequência:

$$\Delta_n = \left|x^{(n+1)} - x^{(n)}\right| \tag{3.62}$$

A pergunta natural é: Será que o erro $\epsilon_n = \left|x^{(n)} - x^*\right|$ será pequeno quando $\Delta_n = \left|x^{(n+1)} - x^{(n)}\right|$ for pequeno?

Para responder a esta pergunta, observamos que

$$x^* = \lim_{n\to\infty} x^{(n)} \tag{3.63}$$

portanto:

$$\begin{aligned} x^* - x^{(N)} &= \left(x^{(N+1)} - x^{(N)}\right) + \left(x^{(N+2)} - x^{(N+1)}\right) + \left(x^{(N+3)} - x^{(N+2)}\right) + \quad (3.64)\\ &= \sum_{k=0}^{\infty} \left(x^{(N+k+1)} - x^{(N+k)}\right) \quad (3.65) \end{aligned}$$

Usamos também as expressões:

$$\begin{aligned} x^{(n+1)} &\approx x^* + (x^{(n)} - x^*)g'(x^*) \quad (3.66)\\ x^{(n)} &\approx x^* + (x^{(n-1)} - x^*)g'(x^*) \quad (3.67) \end{aligned}$$

Subtraindo uma da outra, temos:

$$x^{(n+1)} - x^{(n)} \approx (x^{(n)} - x^{(n-1)})g'(x^*) \tag{3.68}$$

Portanto:

$$x^{(N+k+1)} - x^{(N+k)} \approx (x^{(N+1)} - x^{(N)})\left(g'(x^*)\right)^k \tag{3.69}$$

E temos:

$$\begin{aligned} x^* - x^{(N)} &= \sum_{k=0}^{\infty}\left(x^{(N+k+1)} - x^{(N+k)}\right) && (3.70)\\ &\approx \sum_{k=0}^{\infty}(x^{(N+1)} - x^{(N)})\left(g'(x^*)\right)^k && (3.71)\\ &= (x^{(N+1)} - x^{(N)})\frac{1}{1-g'(x^*)}, \quad |g'(x^*)| < 1 && (3.72) \end{aligned}$$

Tomando módulo, temos:

$$\begin{aligned} \left|x^* - x^{(N)}\right| &\approx \left|x^{(N+1)} - x^{(N)}\right|\frac{1}{1-g'(x^*)} && (3.73)\\ \epsilon_N &\approx \frac{\Delta_N}{1-g'(x^*)} && (3.74) \end{aligned}$$

Observação 3.3.4. Tendo em mente a relação $x^{(n+1)} - x^{(n)} \approx (x^{(n)} - x^{(n-1)})g'(x^*)$, concluímos:

- Quando $g'(x^*) < 0$, o esquema é alternante, isto é, o sinal do erro se altera a cada passo. O erro ϵ_N pode ser estimado diretamente da diferença Δ_N, pois o denominador $1 - g'(x^*) > 1$.
- Quando $0 < g'(x^*) < 1$, o esquema é monótono e $\frac{1}{1-g'(x^*)} > 1$, pelo que o erro ϵ_N é maior que a diferença Δ_N. A relação será tão mais importante quando mais próximo da unidade for $g'(x^*)$, ou seja, quando mais lenta for a convergência. Para estimar o erro em função da diferença Δ_N, observamos que $g'(x^*) \approx \frac{x^{(n+1)} - x^{(n)}}{x^{(n)} - x^{(n-1)}}$ e

  $$|g'(x^*)| \approx \frac{\Delta_n}{\Delta_{n-1}} \tag{3.75}$$

  e portanto

  $$\epsilon_N \approx \frac{\Delta_N}{1 - \frac{\Delta_n}{\Delta_{n-1}}}. \tag{3.76}$$

Exercícios

E 3.3.1. Resolver a equação $e^x = x + 2$ é equivalente a calcular os pontos fixos da função $g(x) = e^x - 2$ (veja o Exemplo 3.3.1). Use a iteração do ponto fixo $x^{(n+1)} = g(x^n)$ com $x^{(1)} = -1{,}8$ para obter uma aproximação de uma das soluções da equação dada com 8 dígitos significativos.

E 3.3.2. Mostre que a equação:

$$\cos(x) = x \tag{3.77}$$

possui uma única solução no intervalo $[0, 1]$. Use a iteração do ponto fixo e encontre uma aproximação para esta solução com 4 dígitos significativos.

E 3.3.3. Mostre que a equação $xe^x = 10$ é equivalente às seguintes equações:

$$x = \ln\left(\frac{10}{x}\right) \quad \text{e} \quad x = 10e^{-x}. \tag{3.78}$$

Destas, considere as seguintes iterações de ponto fixo:

a) $x^{(n+1)} = \ln\left(\dfrac{10}{x^{(n)}}\right)$

b) $x^{(n+1)} = 10e^{-x^{(n)}}$

Tomando $x^{(1)} = 1$, verifique se estas sequências são convergentes.

E 3.3.4. Verifique (analiticamente) que a única solução real da equação:

$$xe^x = 10 \tag{3.92}$$

é ponto fixo das seguintes funções:

a) $g(x) = \ln\left(\frac{10}{x}\right)$

b) $g(x) = x - \frac{xe^x - 10}{15}$

c) $g(x) = x - \frac{xe^x - 10}{10 + e^x}$

Implemente o processo iterativo $x^{(n+1)} = g(x^{(n)})$ para $n \geq 0$ e compare o comportamento. Discuta os resultados com base na teoria estudada.

E 3.3.5. Verifique (analiticamente) que a única solução real da equação:

$$\cos(x) = x \tag{3.93}$$

é ponto fixo das seguintes funções:

a) $g(x) = \cos(x)$

b) $g(x) = 0{,}4x + 0{,}6\cos(x)$

c) $g(x) = x + \frac{\cos(x)-x}{1+\text{sen}\,(x)}$

Implemente o processo iterativo $x^{(n+1)} = g(x^{(n)})$ para $n \geq 0$ e compare o comportamento. Discuta os resultados com base na teoria estudada.

E 3.3.6. Encontre a solução de cada equação com erro absoluto inferior a 10^{-6}.

a) $e^x = x + 2$ no intervalo $(-2{,}0)$.

b) $x^3 + 5x^2 - 12 = 0$ no intervalo $(1{,}2)$.

c) $\sqrt{x} = \cos(x)$ no intervalo $(0{,}1)$.

E 3.3.7. Encontre numericamente as três primeiras raízes positivas da equação dada por:

$$\cos(x) = \frac{x}{10 + x^2} \tag{3.94}$$

com erro absoluto inferior a 10^{-6}.

E 3.3.8. Considere os seguintes processos iterativos:

$$a \begin{cases} x^{(n+1)} &= \cos(x^{(n)}) \\ x^{(1)} &= .5 \end{cases}$$
$$\text{e} \tag{3.95}$$
$$b \begin{cases} x^{(n+1)} &= .4x^{(n)} + .6\cos(x^{(n)}) \\ x^{(1)} &= .5 \end{cases}$$

Use o teorema do ponto fixo para verificar que cada um desses processos converge para a solução da equação x^* de $\cos(x) = x$. Observe o comportamento numérico dessas sequências. Qual estabiliza mais rápido com cinco casas decimais? Discuta.

Dica: Verifique que $\cos([0.5{,}1]) \subseteq [0.5{,}1]$ e depois a mesma identidade para a função $f(x) = 0{,}4x + 0{,}6\cos(x)$.

E 3.3.9. Use o teorema do ponto fixo aplicado a um intervalo adequado para mostrar que a função $g(x) = \ln(100 - x)$ possui um ponto fixo estável.

E 3.3.10. (Fluidos) Na hidráulica, o fator de atrito de Darcy é dado pela implicitamente pela equação de Colebrook-White:

$$\frac{1}{\sqrt{f}} = -2\log_{10}\left(\frac{\varepsilon}{14.8R_h} + \frac{2.51}{\mathrm{Re}\sqrt{f}}\right) \tag{3.96}$$

onde f é o fator de atrito, ε é a rugosidade do tubo em metros, R_h é o raio hidráulico em metros e Re é o número de Reynolds. Considere $\varepsilon = 2mm$, $R_h = 5cm$ e $Re = 10000$ e obtenha o valor de f pela iteração:

$$x^{(n+1)} = -2\log_{10}\left(\frac{\varepsilon}{14.8R_h} + \frac{2.51x^{(n)}}{\mathrm{Re}}\right) \tag{3.97}$$

E 3.3.11. Encontre uma solução aproximada para a equação algébrica

$$180 - 100x = 0.052\,\mathrm{senh}^{\,-1}(10^{13}x) \tag{3.98}$$

com erro absoluto inferior a 10^{-3} usando um método iterativo. Estime o erro associado ao valor de $v = 180 - 100x = 0.052\,\mathrm{senh}^{\,-1}(10^{13}x)$, usando cada uma dessas expressões. Discuta sucintamente o resultado obtido. Dica: Este caso é semelhante ao Problema 3.2.8.

E 3.3.12. Considere que x_n satisfaz a seguinte relação de recorrência:

$$x_{n+1} = x_n - \beta\,(x_n - x^*) \tag{3.99}$$

onde β e x^* são constantes. Prove que

$$x_n - x^* = (1-\beta)^{n-1}(x_1 - x^*). \tag{3.100}$$

Conclua que $x_n \to x^*$ quando $|1-\beta| < 1$.

E 3.3.13. (Convergência lenta) Considere o seguinte esquema iterativo:

$$x^{(n+1)} = x_n + q^n, \tag{3.101}$$
$$x^{(0)} = 0, \tag{3.102}$$

onde $q = 1 - 10^{-6}$.

a) Calcule o limite

$$x_\infty = \lim_{n\to\infty} x^{(n)} \tag{3.103}$$

analiticamente.

b) Considere que o problema de obter o limite da sequência numericamente usando como critério de parada que $|x^{(n+1)} - x^{(n)}| < 10^{-5}$. Qual o valor é produzido pelo esquema numérico? Qual o desvio entre o valor obtido pelo esquema numérico e o valor do limite obtido no item a? Discuta. (Dica: Você não deve implementar o esquema iterativo, obtendo o valor de $x^{(n)}$ analiticamente)

c) Qual deve ser a tolerância especificada para obter o resultado com erro relativo inferior a 10^{-2}?

E 3.3.14. (Convergência sublinear) Considere o seguinte esquema iterativo:

$$x^{(n+1)} = x^{(n)} - [x^{(n)}]^3,\ x^{(n)} \geq 0 \tag{3.104}$$

com $x^{(0)} = 10^{-2}$. Prove que $\{x^{(n)}\}$ é sequência de número reais positivos convergindo para zero. Verifique que são necessários mais de mil passos para que $x^{(n)}$ se torne menor que $0.9x^{(0)}$.

E 3.3.15. (Taxa de convergência)

a) Use o teorema do ponto fixo para mostrar que a função $g(x) = 1 - \operatorname{sen}(x)$ possui um único ponto fixo estável o intervalo $[\frac{1}{10},1]$. Construa um método iterativo $x^{(n+1)} = g(x^{(n)})$ para encontrar esse ponto fixo. Use o computador para encontrar o valor numérico do ponto fixo.

b) Verifique que função $\psi(x) = \frac{1}{2}[x + 1 - \operatorname{sen}(x)]$ possui um ponto fixo x^* que também é o ponto fixo da função g do item a. Use o computador para encontrar o valor numérico do ponto fixo através da iteração $x^{(n+1)} = \psi(x^{(n)})$. Qual método é mais rápido?

E 3.3.16. (Esquemas oscilantes)(*Esquemas oscilantes*)

a) Considere a função $g(x)$ e a função composta $\psi(x) = g \circ g = g(g(x))$. Verifique todo ponto fixo de g também é ponto fixo de ψ.

b) Considere a função

$$g(x) = 10\exp(-x) \tag{3.105}$$

e a função composta $\psi(x) = g \circ g = g(g(x))$. Mostre que ψ possui dois pontos fixos que não são pontos fixos de g.

c) No problema anterior, o que acontece quando o processo iterativo $x^{(n+1)} = g(x^{(n)})$ é inicializado com um ponto fixo de ψ que não é ponto fixo de g?

E 3.3.17. (Aceleração de convergência - introdução ao método de Newton) Mostre que se $f(x)$ possui uma raiz x^* então a x^* é um ponto fixo de $\phi(x) = x + \gamma(x)f(x)$. Encontre uma condição em $\gamma(x)$ para que o ponto fixo x^* de ϕ seja estável. Encontre uma condição em $\gamma(x)$ para que $\phi'(x^*) = 0$.

E 3.3.18. (Aceleração de convergência - introdução ao método de Newton) Considere que $x^{(n)}$ satisfaz a seguinte relação de recorrência:

$$x^{(n+1)} = x^{(n)} - \gamma f(x^{(n)}) \tag{3.106}$$

onde γ é uma constante. Suponha que $f(x)$ possui um zero em x^*. Aproxime a função $f(x)$ em torno de x^* por

$$f(x) = f(x^*) + f'(x^*)(x - x^*) + O\left((x - x^*)^2\right). \tag{3.107}$$

Em vista do problema anterior, qual valor de γ você escolheria para que a sequência $x^{(n)}$ convirja rapidamente para x^*.

E 3.3.19. Considere o problema da Questão 3.2.8 e dois seguintes esquemas iterativos.

$$A\begin{cases} I^{(n+1)} = \frac{1}{R}\left[V - v_t \ln\left(1 + \frac{I^{(n)}}{I_R}\right)\right], n > 0 \\ I^{(0)} = 0 \end{cases}$$

$$\text{e} \tag{3.108}$$

$$B\begin{cases} I^{(n+1)} = I_R\left[\exp\left(\frac{V - RI^{(n)}}{v_t}\right) - 1\right], n > 0 \\ I^{(0)} = 0 \end{cases}$$

Verifique numericamente que apenas o processo A é convergente para a, b e c; enquanto apenas o processo B é convergente para os outros itens.

3.4 Método de Newton-Raphson

Nesta seção, apresentamos o **método de Newton-Raphson**[5][6] para calcular o zero de funções reais de uma variável real.

Consideramos que x^* seja um zero de uma dada função $y = f(x)$ continuamente diferenciável, isto é, $f(x^*) = 0$. A fim de usar a iteração do ponto fixo, observamos que, equivalentemente, x^* é um ponto fixo da função:

$$g(x) = x + \alpha(x)f(x), \quad \alpha(x) \neq 0, \tag{3.109}$$

[5] Joseph Raphson, 1648 - 1715, matemático inglês.
[6] Também chamado apenas de método de Newton.

onde $\alpha(x)$ é uma função arbitrária, a qual escolheremos de forma que a iteração do ponto fixo tenha ótima taxa de convergência.

Do **teorema do ponto fixo**, a taxa de convergência é dada em função do valor absoluto da derivada de $g(x)$. Calculando a derivada temos:

$$g'(x) = 1 + \alpha(x)f'(x) + \alpha'(x)f(x). \tag{3.110}$$

No ponto $x = x^*$, temos:

$$g'(x^*) = 1 + \alpha(x^*)f'(x^*) + \alpha'(x^*)f(x^*). \tag{3.111}$$

Como $f(x^*) = 0$, temos:

$$g'(x^*) = 1 + \alpha(x^*)f'(x^*). \tag{3.112}$$

Sabemos que o processo iterativo converge tão mais rápido quanto menor for $|g'(x)|$ nas vizinhanças de x^*. Isto nos leva a escolher:

$$g'(x^*) = 0, \tag{3.113}$$

e, então, temos:

$$\alpha(x^*) = -\frac{1}{f'(x^*)}, \tag{3.114}$$

se $f'(x^*) \neq 0$.

A discussão acima nos motiva a introduzir o método de Newton, cujas iterações são dada por:

$$x^{(n+1)} = x^{(n)} - \frac{f\left(x^{(n)}\right)}{f'\left(x^n\right)}, \quad n \geq 1, \tag{3.115}$$

sendo $x^{(1)}$ uma aproximação inicial dada.

3.4.1 Interpretação geométrica

Seja uma dada função $f(x)$ conforme na Figura 3.6. Para tanto, escolhemos uma aproximação inicial $x^{(1)}$ e computamos:

$$x^{(2)} = x^{(1)} - \frac{f(x^{(1)})}{f'(x^{(1)})}. \tag{3.116}$$

Geometricamente, o ponto $x^{(2)}$ é a interseção da reta tangente ao gráfico da função $f(x)$ no ponto $x = x^{(1)}$ com o eixo das abscissas. Com efeito, a equação desta reta é:

$$y = f'(x^{(1)})(x - x^{(1)}) + f(x^{(1)}). \tag{3.117}$$

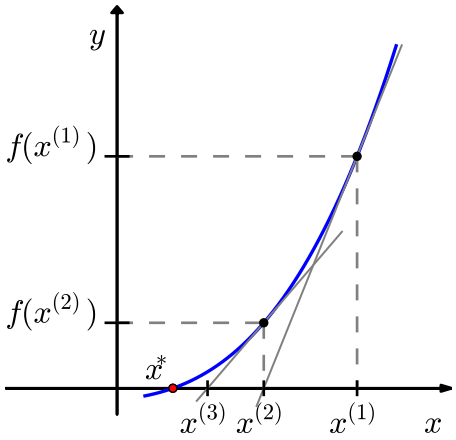


Figura 3.6: Interpretação do método de Newton.

Assim, a interseção desta reta com o eixo das abscissas ($y = 0$) ocorre quando:

$$f'(x^{(1)})(x - x^{(1)}) + f(x^{(1)}) = 0 \Rightarrow x = x^{(1)} - \frac{f(x^{(1)})}{f'(x^{(1)})}. \tag{3.118}$$

Ou seja, dada aproximação $x^{(n)}$, a próxima aproximação $x^{(n+1)}$ é o ponto de interseção entre o eixo das abscissas e a reta tangente ao gráfico da função no ponto $x = x^{(n)}$. Observe a Figura 3.6.

3.4.2 Análise de convergência

Seja $y = f(x)$ uma função com derivadas primeira e segunda contínuas tal que $f(x^*) = 0$ e $f'(x^*) \neq 0$. Seja também a função $g(x)$ definida como:

$$g(x) = x - \frac{f(x)}{f'(x)}. \tag{3.119}$$

Expandindo em série de Taylor em torno de $x = x^*$, obtemos:

$$g(x) = g(x^*) + g'(x^*)(x - x^*) + \frac{g''(x^*)}{2}(x - x^*)^2 + O\left((x - x^*)^3\right). \tag{3.120}$$

Observamos que:

$$g(x^*) = x^* \tag{3.121}$$

$$g'(x^*) = 1 - \frac{f'(x^*)f'(x^*) - f(x^*)f''(x^*)}{(f'(x^*))^2} = 0 \tag{3.122}$$

Portanto:

$$g(x) = x^* + \frac{g''(x^*)}{2}(x - x^*)^2 + O\left((x - x^*)^3\right) \tag{3.123}$$

Com isso, temos:

$$x^{(n+1)} = g(x^{(n)}) = x^* + \frac{g''(x^*)}{2}(x^{(n)} - x^*)^2 + O\left((x - x^*)^3\right), \tag{3.124}$$

ou seja:

$$\left|x^{(n+1)} - x^*\right| \leq C\left|x^{(n)} - x^*\right|^2, \tag{3.125}$$

com constante $C = |g''(x^*)/2|$. Isto mostra que o método de Newton tem **taxa de convergência quadrática**. Mais precisamente, temos o seguinte teorema.

Teorema 3.4.1 (Método de Newton)**.** *Sejam $f \in C^2([a,b])$ com $x^* \in (a,b)$ tal que $f(x^*) = 0$ e:*

$$m := \min_{x\in[a,b]} |f'(x)| > 0 \quad e \quad M := \max_{x\in[a,b]} |f''(x)|. \tag{3.126}$$

Escolhendo $\rho > 0$ tal que:

$$q := \frac{M}{2m}\rho < 1, \tag{3.127}$$

*definimos a **bacia de atração** do método de Newton pelo conjunto:*

$$K_\rho(x^*) := \{x \in \mathbb{R};\ |x - x^*| \leq \rho\} \subset [a,b]. \tag{3.128}$$

Então, para qualquer $x^{(1)} \in K_\rho(x^*)$ a iteração do método de Newton:

$$x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)})}{f'(x^{(n)})}, \tag{3.129}$$

*fornece uma sequência $x^{(n)}$ que converge para x^*, isto é, $x^{(n)} \to x^*$ quando $n \to \infty$. Além disso, temos a seguinte estimativa de erro **a priori**:*

$$|x^{(n)} - x^*| \leq \frac{2m}{M} q^{(2^{n-1})}, \quad n \geq 2, \tag{3.130}$$

*e a seguinte estimativa de erro **a posteriori**:*

$$|x^{(n)} - x^*| \leq \frac{M}{2m}|x^{(n)} - x^{(n-1)}|^2, \quad n \geq 2. \tag{3.131}$$

Demonstração. Para $n \in \mathbb{N}$, $n \geq 2$, temos:

$$x^{n+1} - x^* = x^{(n)} - \frac{f(x^{(n)})}{f'(x^{(n)})} - x^* = -\frac{1}{f(x^{(n)})}\left[f(x^{(n)}) + (x^* - x^{(n)})f'(x^{(n)})\right]. \tag{3.132}$$

Agora, para estimar o lado direito desta equação, usamos o polinômio de Taylor de grau 1 da função $f(x)$ em torno de $x = x^{(n)}$, isto é:

$$f(x^*) = f(x^{(n)}) + (x^* - x^{(n)})f'(x^{(n)}) + \int_{x^{(n)}}^{x^*} f''(t)(x^* - t)\,dt. \tag{3.133}$$

Pela mudança de variável $t = x^{(n)} + s(x^{(n)} - x^*)$, observamos que o resto deste polinômio de Taylor na forma integral é igual a:

$$R(x^*,x^{(n)}) := (x^* - x^{(n)})^2 \int_0^1 f''\left(x^{(n)} + s(x^* - x^{(n)})\right)(1 - s)\,ds. \tag{3.134}$$

Assim, da cota da segunda derivada de $f(x)$, temos:

$$|R(x^*,x^{(n)})| \leq M|x^* - x^{(n)}|^2 \int_0^1 (1 - s)\,ds = \frac{M}{2}|x^* - x^{(n)}|^2. \tag{3.135}$$

Se $x^{(n)} \in K_\rho(x^*)$, então de (3.132) e (3.135) temos:

$$|x^{(n+1)} - x^*| \leq \frac{M}{2m}|x^{(n)} - x^*|^2 \leq \frac{M}{2m}\rho^2 < \rho. \tag{3.136}$$

Isto mostra que se $x^{(n)} \in K_\rho(x^*)$, então $x^{(n+1)} \in K_\rho(x^*)$, isto é, $x^{(n)} \in K_\rho(x^*)$ para todo $n \in \mathbb{R}$.

Agora, obtemos a estimativa **a priori** de (3.4.2), pois:

$$|x^{(n)} - x^*| \leq \frac{2m}{M}\left(\frac{M}{2m}|x^{(n-1)} - x^*|\right)^2 \leq \cdots \leq \frac{2m}{M}\left(\frac{M}{2m}|x^{(1)} - x^*|\right)^{2^{n-1}}. \tag{3.137}$$

Logo:

$$|x^{(n)} - x^*| \leq \frac{2m}{M}q^{2^{n-1}}, \tag{3.138}$$

donde também vemos que $x^{(n)} \to x^*$ quando $n \to \infty$, pois $q < 1$.

Por fim, para provarmos a estimativa **a posteriori** tomamos a seguinte expansão em polinômio de Taylor:

$$f(x^{(n)}) = f(x^{(n-1)}) + (x^{(n)} - x^{(n-1)})f'(x^{(n-1)}) + R(x^{(n)},x^{(n-1)}). \tag{3.139}$$

Aqui, temos:

$$f(x^{(n-1)}) + (x^{(n)} - x^{(n-1)})f'(x^{(n-1)}) = 0 \tag{3.140}$$

e, então, conforme acima:

$$|f(x^{(n)})| = |R(x^{(n)},x^{(n-1)})| \leq \frac{M}{2}|x^{(n)} - x^{(n-1)}|^2. \tag{3.141}$$

Com isso e do teorema do valor médio, concluímos:

$$|x^{(n)} - x^*| \leq \frac{1}{m}|f(x^{(n)}) - f(x^*)| \leq \frac{M}{2m}|x^{(n)} - x^{(n-1)}|^2. \tag{3.142}$$

□

Exemplo 3.4.1. Estime o raio ρ da bacia de atração $K_\rho(x^*)$ para a função $f(x) = \cos(x) - x$ restrita ao intervalo $[0, \pi/2]$.

Solução. O raio da bacia de atração é tal que:

$$\rho < \frac{2m}{M} \tag{3.143}$$

onde $m := \min |f'(x)|$ e $M := \max |f''(x)|$ com o mínimo e o máximo tomados em um intervalo $[a, b]$ que contenha o zero da função $f(x)$. Aqui, por exemplo, podemos tomar $[a, b] = [0, \pi/2]$. Como, neste caso, $f'(x) = -\operatorname{sen}(x) - 1$, temos que $m = 1$. Também, como $f''(x) = -\cos x$, temos $M = 1$. Assim, concluímos que $\rho < 2$ (lembrando que $K_\rho(x^*) \subset [0, \pi/2]$). Ou seja, neste caso as iterações de Newton convergem para o zero de $f(x)$ para qualquer escolha da aproximação inicial $x^{(1)} \in [0, \pi/2]$. ◊

Exercícios

E 3.4.1. Encontre a raiz positiva da função $f(x) = \cos(x) - x^2$ pelo método de Newton inicializando-o com $x^{(0)} = 1$. Realize a iteração até obter estabilidade no *quinto* dígito significativo.

E 3.4.2. Considere o problema de calcular as soluções positivas da equação:

$$\operatorname{tg}(x) = 2x^2. \tag{3.144}$$

a) Use o método gráfico para isolar as duas primeiras raízes positivas em pequenos intervalos. Use a teoria para argumentar quanto à existência e unicidade das raízes dentro intervalos escolhidos.

b) Calcule cada uma das raízes pelo método de Newton com oito dígitos significativos e discuta a convergência.

E 3.4.3. Considere a equação

$$e^{-x^2} = x \tag{3.145}$$

trace o gráfico com auxílio do computador e verifique que ela possui uma raiz positiva. Encontre uma aproximação para esta raiz pelo gráfico e use este valor para inicializar o método de Newton e obtenha uma aproximação para a raiz com 8 dígitos significativos.

E 3.4.4. Isole e encontre as cinco primeiras raízes positivas da equação com 6 dígitos corretos através de traçado de gráfico e do método de Newton.

$$\cos(10x) = e^{-x}. \tag{3.146}$$

Dica: a primeira raiz positiva está no intervalo $(0,0{,}02)$. Fique atento.

E 3.4.5. Encontre as raízes do polinômio $f(x) = x^4 - 4x^2 + 4$ através do método de Newton. O que você observa em relação ao erro obtido? Compare com a situação do Problema 3.2.4.

E 3.4.6. Encontre as raízes reais do polinômio $f(x) = \frac{x^5}{100} + x^4 + 3x + 1$ isolando-as pelo método do gráfico e depois usando o método de Newton. Expresse a solução com 7 dígitos significativos.

E 3.4.7. Considere o método de Newton aplicado para encontrar a raiz de $f(x) = x^3 - 2x + 2$. O que acontece quando $x^{(0)} = 0$? Escolha um valor adequado para inicializar o método e obter a única raiz real desta equação.

E 3.4.8. Justifique a construção do processo iterativo do método de Newton através do conceito de estabilidade de ponto fixo e convergência do método da iteração. Dica: Considere os problemas 3.3.17 e 3.3.18.

E 3.4.9. Entenda a interpretação geométrica ao método de Newton. Encontre uma valor para iniciar o método de Newton aplicado ao problema $f(x) = xe^{-x} = 0$ tal que o esquema iterativo divirja.

E 3.4.10. (Computação) Aplique o método de Newton à função $f(x) = \frac{1}{x} - A$ e construa um esquema computacional para calcular a inversa de A com base em operações de multiplicação e soma/subtração.

E 3.4.11. (Computação) Aplique o método de Newton à função $f(x) = x^n - A$ e construa um esquema computacional para calcular $\sqrt[n]{A}$ para $A > 0$ com base em operações de multiplicação e soma/subtração.

E 3.4.12. (Computação) Aplique o método de Newton à função $f(x) = \frac{1}{x^2} - A$ e construa um esquema computacional para calcular $\frac{1}{\sqrt{A}}$ para $A > 0$ com base em operações de multiplicação e soma/subtração.

E 3.4.13. Considere a função dada por

$$\psi(x) \quad = \quad \ln\left(15 - \ln(x)\right) \tag{3.155}$$

definida para $x \in (0,e^{15})$

a) Use o teorema do ponto fixo para provar que se $x^{(0)}$ pertence ao intervalo [1,3], então a sequência dada iterativamente por

$$x^{(n+1)} = \psi(x^{(n)}), n \geq 0 \tag{3.156}$$

converge para o único ponto fixo, x^*, de ψ. Construa a iteração $x^{(n+1)} = \psi(x^{(n)})$ e obtenha numericamente o valor do ponto fixo x^*. Expresse a resposta com 5 algarismos significativos corretos.

b) Construa a iteração do método de Newton para encontrar x^*, explicitando a relação de recorrência e iniciando com $x_0 = 2$. Use o computador para obter a raiz e expresse a resposta com oito dígitos significativos corretos.

3.5 Método das secantes

O **método das secantes** é uma variação do método de Newton, evitando a necessidade de conhecer-se a derivada analítica de $f(x)$. Dada uma função $f(x)$, a ideia é aproximar sua derivada pela razão fundamental:

$$f'(x) \approx \frac{f(x) - f(x_0)}{x - x_0}, \quad x \approx x_0. \tag{3.157}$$

Mais precisamente, o método de Newton é uma iteração de ponto fixo da forma:

$$x^{(n+1)} = x^{(n)} - \alpha(x^{(n)}) f(x^{(n)}), \quad n \geq 1, \tag{3.158}$$

onde $x^{(1)}$ é uma aproximação inicial dada e $\alpha(x^{(n)}) = 1/f'(x^{(n)})$. Usando a aproximação da derivada acima, com $x = x^{(n)}$ e $x_0 = x^{(n-1)}$, temos:

$$\alpha(x^{(n)}) = \frac{1}{f'(x^{(n)})} \approx \frac{x^{(n)} - x^{(n-1)}}{f(x^{(n)}) - f(x^{(n-1)})}. \tag{3.159}$$

Isto nos motiva a introduzir a **iteração do método das secantes** dada por:

$$x^{(n+1)} = x^{(n)} - f(x^{(n)}) \frac{x^{(n)} - x^{(n-1)}}{f(x^{(n)}) - f(x^{(n-1)})}, \quad n \geq 2. \tag{3.160}$$

Observe que para inicializarmos a iteração acima precisamos de duas aproximações iniciais, a saber, $x^{(1)}$ e $x^{(2)}$. Maneiras apropriadas de escolher estas aproximações podem ser inferidas da interpretação geométrica do método.

Exemplo 3.5.1. Encontre as raízes de $f(x) = \cos(x) - x$.

Solução. Da inspeção do gráfico das funções $y = \cos(x)$ e $y = x$, sabemos que esta equação possui uma raiz em torno de $x = 0{,}8$. Iniciamos o método com $x_0 = 0{,}7$ e $x_1 = 0{,}8$.

$x^{(n-1)}$	$x^{(n)}$	m	$x^{(n+1)}$
0,7	0,8	$\frac{f(0,8)-f(0,7)}{0,8-0,7} =$ $-1{,}6813548$	$0{,}8 - \frac{f(0,8)}{-1,6813548} =$ 0,7385654
0,8	0,7385654	$-1{,}6955107$	0,7390784
0,7385654	0,7390784	$-1{,}6734174$	0,7390851
0,7390784	0,7390851	$-1{,}6736095$	0,7390851

◊

3.5.1 Interpretação geométrica

Enquanto, o método de Newton está relacionado às retas tangentes ao gráfico da função objetivo $f(x)$, o método das secantes, como o próprio nome indica, está relacionado às retas secantes.

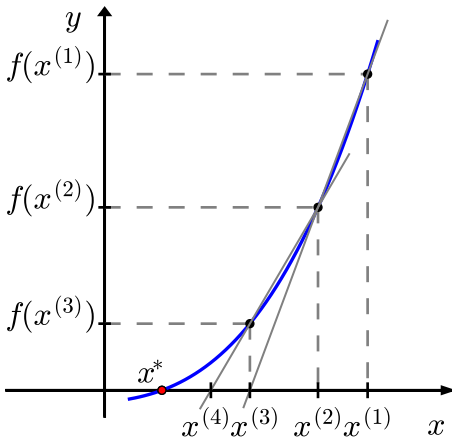


Figura 3.7: Método das secantes.

Sejam $f(x)$ e as aproximações $x^{(1)}$ e $x^{(2)}$ do zero x^* desta função (veja Figura 3.7). A iteração do método das secantes fornece:

$$x^{(3)} = x^{(2)} - f(x^{(2)})\frac{x^{(2)} - x^{(1)}}{f(x^{(2)}) - f(x^{(1)})}. \tag{3.161}$$

De fato, $x^{(3)}$ é o ponto de interseção da reta secante ao gráfico de $f(x)$ pelos pontos $x^{(1)}$ e $x^{(2)}$ com o eixo das abscissas. Com efeito, a equação desta reta secante é:

$$y = \frac{f(x^{(2)}) - f(x^{(1)})}{x^{(2)} - x^{(1)}}(x - x^{(2)}) + f(x^{(2)}). \tag{3.162}$$

Esta reta intercepta o eixo das abscissas no ponto x tal que $y = 0$, isto é:

$$\frac{f(x^{(2)}) - f(x^{(1)})}{x^{(2)} - x^{(1)}}(x - x^{(2)}) + f(x^{(2)}) \Rightarrow x = x^{(2)} - f(x^{(2)})\frac{x^{(2)} - x^{(1)}}{f(x^{(2)}) - f(x^{(1)})}. \tag{3.163}$$

3.5.2 Análise de convergência

Uma análise assintótica semelhante àquela feita para o método de Newton na subseção 3.4.2 nos indica que, para uma função $f(x)$ duas vezes diferenciável, as iterações do método da secante satisfazem:

$$|x^{(n+1)} - x^*| \approx C|x^{(n)} - x^*||x^{(n-1)} - x^*|, \tag{3.164}$$

para aproximações iniciais suficientemente próximas de x^*, onde $f(x^*) = 0$. Além disso, veremos que:

$$|x^{(n+1)} - x^*| \leq C|x^{(n)} - x^*|^p, \;\; p = \frac{\sqrt{5}+1}{2} \approx 1{,}618 \tag{3.165}$$

sob certas condições. Ou seja, o método das secantes tem **taxa de convergência superlinear**.

Teorema 3.5.1 (Método das secantes)**.** *Seja $f \in C^2([a,b])$ uma função com $x^* \in (a,b)$ tal que $f(x^*) = 0$. Sejam, também:*

$$m := \min_{x\in[a,b]} |f'(x)| > 0 \quad e \quad M := \max_{x\in[a,b]} |f''(x)| < \infty. \tag{3.166}$$

Além disso, seja $\rho > 0$ tal que:

$$q := \frac{M}{2m}\rho < 1, \quad K_\rho(x^*) := \{x \in \mathbb{R};\; |x - x^*| \leq \rho\} \subset [a,b]. \tag{3.167}$$

*Então, para aproximações iniciais $x^{(1)}, x^{(2)} \in K_\rho(x^*)$, com $x^{(1)} \neq x^{(2)}$, temos que as iterações do método das secantes $x^{(n)} \in K_\rho(x^*)$, $n \geq 1$, e $x^{(n)} \to x^*$, quando $n \to \infty$. Além disso, vale a seguinte estimativa de convergência **a priori**:*

$$|x^{(n)} - x^*| \leq \frac{2m}{M}q^{\gamma_{n-1}}, \quad n \geq 1, \tag{3.168}$$

onde $\{\gamma_n\}_{n\in\mathbb{N}}$ *é a sequência de Fibonacci*[7][8]*, bem como vale a estimativa* ***a posteriori****:*

$$|x^{(n)} - x^*| \leq \frac{M}{2m}|x^{(n)} - x^{(n-1)}||x^{(n-1)} - x^{(n-2)}|, \quad n \geq 3. \tag{3.169}$$

Demonstração. Sejam $n \in \mathbb{N}$ com $n \geq 2$ e $x^{(n)}, x^{(n-1)} \in K_\rho(x^*)$, tal que $x^{(n)} \neq x^{(n-1)}$, $x^{(n)} \neq x^*$ e $x^{(n-1)} \neq x^*$. Seja, também:

$$g(x^{(n)},x^{(n-1)}) := x^{(n)} - f(x^{(n)})\frac{x^{(n)} - x^{(n-1)}}{f(x^{(n)}) - f(x^{(n-1)})}. \tag{3.170}$$

Com isso, temos:

$$g(x^{(n)},x^{(n-1)}) - x^* = x^{(n)} - f(x^{(n)})\frac{x^{(n)} - x^{(n-1)}}{f(x^{(n)}) - f(x^{(n-1)})} - x^* \tag{3.171}$$

$$= \frac{x^{(n)} - x^{(n-1)}}{f(x^{(n)}) - f(x^{(n-1)})}\left\{(x^{(n)} - x^*)\frac{f(x^{(n)}) - f(x^{(n-1)})}{x^{(n)} - x^{(n-1)}} - f(x^{(n)}) + f(x^*)\right\} \tag{3.172}$$

$$\tag{3.173}$$

Então, da cota assumida para primeira derivada de $f(x)$ e do teorema do valor médio, temos:

$$|g(x^{(n)},x^{(n-1)}) - x^*| \leq \frac{|x^{(n)} - x^*|}{m}\left|\frac{f(x^{(n)}) - f(x^{(n-1)})}{x^{(n)} - x^{(n-1)}} - \frac{f(x^{(n)}) - f(x^*)}{x^{(n)} - x^*}\right|. \tag{3.174}$$

Agora, iremos estimar este último termo a direita. Para tanto, começamos observando que da expansão em polinômio de Taylor de ordem 0 da função $f(x)$ com resto na forma integral, temos:

$$\frac{f(x^{(n)}) - f(x^{(n-1)})}{x^{(n)} - x^{(n-1)}} = -\int_0^1 \frac{d}{dr}f(x^{(n)} + r(x^{(n-1)} - x^{(n)}))\frac{dr}{x^{(n)} - x^{(n-1)}} \tag{3.175}$$

$$= \int_0^1 f'(x^{(n)} + r(x^{(n-1)} - x^{(n)}))\,dr \tag{3.176}$$

De forma análoga, temos:

$$\frac{f(x^{(n)}) - f(x^*)}{x^{(n)} - x^*} = \int_0^1 f'(x^{(n)} + r(x^* - x^{(n)}))\,dr \tag{3.177}$$

Logo, temos:

$$\frac{f(x^{(n)}) - f(x^{(n-1)})}{x^{(n)} - x^{(n-1)}} - \frac{f(x^{(n)}) - f(x^*)}{x^{(n)} - x^*} = \int_0^1 \left[f'(x^{(n)} + r(x^{(n-1)} - x^{(n)})) - f'(x^{(n)} + r(x^* - x^{(n)}))\right]\,dr. \tag{3.178}$$

[7] Leonardo Fibonacci, c. 1170 - c. 1250, matemático italiano.

[8] A sequência de Fibonacci $\{\gamma_n\}_{n\in\mathbb{N}}$ é definida por $\gamma_0 = \gamma_1 = 1$ e $\gamma_{n+1} = \gamma_n - \gamma_{n-1}$, $n \geq 1$.

Agora, novamente temos:

$$\begin{aligned} & f'(x^{(n)} + r(x^{(n-1)} - x^{(n)})) - f'(x^{(n)} + r(x^* - x^{(n)})) \\ &= \int_0^r \frac{d}{ds} f'(x^{(n)} + r(x^{(n-1)} - x^{(n)}) + s(x^* - x^{(n-1)}))\, ds \\ &= \int_0^r f''(x^{(n)} + r(x^{(n-1)} - x^{(n)}) + s(x^* - x^{(n-1)}))\, ds (x^* - x^{(n-1)}). \end{aligned} \tag{3.179}$$

Retornando à Equação (3.178) e usando a cota para a segunda derivada, obtemos:

$$\left| \frac{f(x^{(n)}) - f(x^{(n-1)})}{x^{(n)} - x^{(n-1)}} - \frac{f(x^{(n)}) - f(x^*)}{x^{(n)} - x^*} \right| \leq \frac{M}{2} |x^{(n-1)} - x^*|. \tag{3.180}$$

Utilizando a Equação (3.174), obtemos:

$$|g(x^{(n)}, x^{(n-1)}) - x^*| \leq \frac{M}{2m} |x^{(n)} - x^*||x^{(n-1)} - x^*| \leq \frac{M}{2m} \rho^2 < \rho. \tag{3.181}$$

Portanto, concluímos que as iterações do método da secantes $x^{(n)}$ permanecem no conjunto $K_\rho(x^*)$, se começarem nele. Além disso, temos demonstrado que:

$$|x^{(n+1)} - x^*| \leq \frac{M}{2m} |x^{(n)} - x^*||x^{(n-1)} - x^*|. \tag{3.182}$$

Com isso, temos:

$$\rho_n := \frac{M}{2m} |x^{(n)} - x^*| \Rightarrow \rho_{n+1} \leq \rho_n \rho_{n-1}, \quad n \geq 2. \tag{3.183}$$

Como $\rho_1 \leq q$ e $\rho_2 \leq q$, temos $\rho_n \leq q^{\gamma_{n-1}}$, $n \geq 1$. Isto mostra a estimativa de convergência **a priori**:

$$|x^n - x^*| \leq \frac{2m}{M} q^{\gamma_{n-1}}. \tag{3.184}$$

Além disso, como $\gamma_n \to \infty$ quando $n \to \infty$ e $q < 1$, temos que as iterações do método das secantes $x^{(n)} \to x^*$ quando $n \to \infty$.

Por fim, mostramos a estimativa de convergência **a posteriori**. Para tanto, da cota assumida para a primeira derivada e do teorema do valor médio, temos, para $n \geq 3$:

$$|x^{(n)} - x^*| \leq \frac{1}{m} |f(x^{(n)} - f(x^*)| \tag{3.185}$$

$$= \frac{1}{m} \left| f(x^{(n-1)}) + (x^{(n)} - x^{(n-1)}) \frac{f(x^{(n)}) - f(x^{(n-1)})}{x^{(n)} - x^{(n-1)}} \right| \tag{3.186}$$

$$= \frac{1}{m} \left| x^{(n)} - x^{(n-1)} \right| \left| \frac{f(x^{(n)}) - f(x^{(n-1)})}{x^{(n)} - x^{(n-1)}} + \frac{f(x^{(n-1)})}{x^{(n)} - x^{(n-1)}} \right| \tag{3.187}$$

Agora, a iteração do método das secantes fornece:

$$x^{(n)} = x^{(n-1)} - f(x^{(n-1)})\frac{x^{(n-1)} - x^{(n-2)}}{f(x^{(n-1)}) - f(x^{(n-2)})} \tag{3.188}$$

e temos:

$$\frac{f(x^{(n-1)})}{x^{(n)} - x^{(n-1)}} = -\frac{f(x^{(n-1)}) - f(x^{(n-2)})}{x^{(n-1)} - x^{(n-2)}}. \tag{3.189}$$

Portanto:

$$|x^{(n)} - x^*| \leq \frac{1}{m}|x^{(n)} - x^{(n-1)}| \left| \frac{f(x^{(n-1)}) - f(x^{(n)})}{x^{(n-1)} - x^{(n)}} - \frac{f(x^{(n-1)}) - f(x^{(n-2)})}{x^{(n-1)} - x^{(n-2)}} \right|. \tag{3.190}$$

Observamos que o último termo pode ser estimado como feito acima para o termo análogo na Inequação (3.174). Com isso, obtemos a estimativa desejada:

$$|x^{(n)} - x^*| \leq \frac{M}{2m}|x^{(n)} - x^{(n-1)}||x^{(n)} - x^{(n-2)}|. \tag{3.191}$$

□

Proposição 3.5.1 (Sequência de Fibonacci)**.** *A sequência de Fibonacci* $\{\gamma_n\}_{n\in\mathbb{N}}$ *é assintótica a* $\gamma_n \sim \lambda_1^{n+1}/\sqrt{5}$ *e:*

$$\lim_{n\to\infty} \frac{\gamma_{n+1}}{\gamma_n} = \lambda_1, \tag{3.192}$$

onde $\lambda_1 = (1+\sqrt{5})/2 \approx 1{,}618$ *é a porção áurea.*

Demonstração. A sequência de Fibonacci $\{\gamma_n\}_{n\in\mathbb{N}}$ é definida por $\gamma_0 = \gamma_1 = 1$ e $\gamma_{n+1} = \gamma_n + \gamma_{n-1}$, $n \geq 1$. Logo, satisfaz a seguinte equação de diferenças:

$$\gamma_{n+2} - \gamma_{n+1} - \gamma_n = 0, \quad n \in \mathbb{N}. \tag{3.193}$$

Tomando $\gamma_n = \lambda^n$, $\lambda \neq 0$ temos:

$$\lambda^n\left(\lambda^2 - \lambda - 1\right) = 0 \Rightarrow \lambda^2 - \lambda - 1 = 0 \Rightarrow \lambda_{1,2} = \frac{1 \pm \sqrt{5}}{2}. \tag{3.194}$$

Portanto, $\gamma_n = c_1\lambda_1^n + c_2\lambda_2^n$. Como $\gamma_0 = \gamma_1 = 1$, as constantes satisfazem:

$$\begin{aligned} &c_1 + c_2 = 1 \\ &c_1\lambda_1 + c_2\lambda_2 = 1 \end{aligned} \Rightarrow c_1 = \frac{1+\sqrt{5}}{2\sqrt{5}}, \quad c_2 = -\frac{1-\sqrt{5}}{2\sqrt{5}}. \tag{3.195}$$

Ou seja, obtemos a seguinte forma explícita para os números de Fibonacci:

$$\gamma_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^{n+1} - \left(\frac{1-\sqrt{5}}{2}\right)^{n+1}\right]. \tag{3.196}$$

Daí, segue imediatamente o enunciado. □

Observação 3.5.1. Sob as hipóteses do Teorema 3.5.1 e da Proposição 3.5.1, temos:

$$\lim_{n\to\infty} \frac{|x^{(n+1)} - x^*|}{|x^{(n)} - x^*|^{\lambda_1}} \leq \lim_{n\to\infty} \frac{M}{2m} |x^{(n)} - x^*|^{1-\lambda_1} |x^{(n-1)} - x^*| \tag{3.197}$$

$$\leq \lim_{n\to\infty} \left(\frac{2m}{M}\right)^{1-\lambda_1} q^{(2-\lambda_1)\lambda_1^n/\sqrt{5}} = 0. \tag{3.198}$$

Isto mostra que o método das secantes (nestas hipóteses) tem taxa de convergência superlinear ($\lambda_1 \approx 1{,}6$).

3.6 Critérios de parada

Quando usamos métodos iterativos precisamos determinar um critério de parada. A Tabela 3.4 indica critérios de parada usuais para os métodos que estudamos neste capítulo.

Tabela 3.4: Quadro comparativo.

Método	Convergência	Erro	Critério de parada
Bisseção	Linear ($p = 1$)	$\epsilon_{n+1} = \frac{1}{2}\epsilon$	$\frac{b_n - a_n}{2} <$ erro
Iteração linear	Linear ($p = 1$)	$\epsilon_{n+1} \approx \lvert\phi'(x^*)\rvert\varepsilon_n$	$\frac{\lvert\Delta_n\rvert}{1-\frac{\Delta_n}{\Delta_{n-1}}} <$ erro, $\Delta_n < \Delta_{n-1}$
Newton	Quadrática ($p = 2$)	$\epsilon_{n+1} \approx \frac{1}{2}\left\lvert\frac{f''(x^*)}{f'(x^*)}\right\rvert\varepsilon_n^2$	$\lvert\Delta_n\rvert <$ erro
Secante	$p = \frac{\sqrt{5}+1}{2} \approx 1{,}618$	$\varepsilon_{n+1} \approx \left\lvert\frac{f''(x^*)}{f'(x^*)}\right\rvert\varepsilon_n\varepsilon_{n-1} \approx M\varepsilon_n^{\phi}$	$\lvert\Delta_n\rvert <$ erro

Observação 3.6.1. O erro na tabela sempre se refere ao erro absoluto esperado. Nos três últimos métodos, é comum que se exija como critério de parada que a

condição seja satisfeita por alguns poucos passos consecutivos. Outros critérios podem ser usados. No métodos das secantes, deve-se ter o cuidado de evitar divisões por zero quando $x_{n+1} - x_n$ muito pequeno em relação à resolução do sistema de numeração.

Exercícios

E 3.6.1. Refaça as questões 3.4.3, 3.4.4, 3.4.5 e 3.4.6, usando o método das secantes.

E 3.6.2. Dê uma interpretação geométrica ao método das secantes. Qual a vantagem do método das secantes sobre o método de Newton?

E 3.6.3. Aplique o método das secantes para resolver a equação

$$e^{-x^2} = 2x \tag{3.199}$$

E 3.6.4. Refaça o Problema 3.2.8 usando o método de Newton e das secantes.

E 3.6.5. Seja uma função $f(x)$ dada duas vezes continuamente diferenciável. Faça uma análise assintótica para mostrar que as iterações do método das secantes satisfazem:

$$|x^{(n+1)} - x^*| \approx C|x^{(n)} - x^*||x^{(n-1)} - x^*|, \tag{3.200}$$

para aproximações iniciais $x^{(1)}$ e $x^{(2)}$ suficientemente próximas de x^*, onde $f(x^*) = 0$.

3.7 Exercícios finais

E 3.7.1. Calcule uma equação da reta tangente a curva $y = e^{-(x-1)^2}$ que passa pelo ponto $(3, 1/2)$.

E 3.7.2. Resolva numericamente a inequação:

$$e^{-x^2} < 2x \tag{3.218}$$

E 3.7.3. A equação

$$\cos(\pi x) = e^{-2x} \tag{3.219}$$

tem infinitas raízes. Usando métodos numéricos encontre as primeiras raízes dessa equação. Verifique a j-ésima raiz (z_j) pode ser aproximada por $j - 1/2$ para j

grande. Use o método de Newton para encontrar uma aproximação melhor para z_j.

E 3.7.4. (Eletricidade) A corrente elétrica, I, em Ampères em uma lâmpada em função da tensão elétrica, V, é dada por

$$I = \left(\frac{V}{150}\right)^{0.8} \tag{3.220}$$

Qual a potência da lâmpada quando ligada em série com uma resistência de valor R a uma fonte de 150V quando. (procure erro inferior a 1%)

a) $R = 0\Omega$

b) $R = 10\Omega$

c) $R = 50\Omega$

d) $R = 100\Omega$

E) $R = 500\Omega$

E 3.7.5. (Bioquímica) A concentração sanguínea de um medicamente é modelado pela seguinte expressão

$$c(t) = Ate^{-\lambda t} \tag{3.221}$$

onde $t > 0$ é o tempo em minutos decorrido desde a administração da droga. A é a quantidade administrada em mg/ml e λ é a constante de tempo em min^{-1}. Responda:

a) Sendo $\lambda = 1/3$, em que instantes de tempo a concentração é metade do valor máximo. Calcule com precisão de segundos.

b) Sendo $\lambda = 1/3$ e $A = 100mg/ml$, durante quanto tempo a concentração permanece maior que $10mg/ml$.

E 3.7.6. Considere o seguinte modelo para crescimento populacional em um país:

$$P(t) = A + Be^{\lambda t}. \tag{3.222}$$

onde t é dado em anos. Use t em anos e $t = 0$ para 1960. Encontre os parâmetros A, B e λ com base nos anos de 1960, 1970 e 1991 conforme tabela:

Ano	população
1960	70992343
1970	94508583
1980	121150573
1991	146917459

Use esses parâmetros para calcular a população em 1980 e compare com o valor do censo. Dica: considere $\frac{P(31)-P(0)}{P(10)-P(0)}$ e reduza o sistema a uma equação apenas na variável λ.

E 3.7.7. (Fluidos) Uma boia esférica flutua na água. Sabendo que a boia tem 10ℓ de volume e 2Kg de massa. Calcule a altura da porção molhada da boia.

E 3.7.8. (Fluidos) Uma boia cilíndrica tem secção transversal circular de raio 10cm e comprimento 2m e pesa 10Kg. Sabendo que a boia flutua sobre água com o eixo do cilindro na posição horizontal, calcule a altura da parte molhada da boia.

E 3.7.9. Encontre com 6 casas decimais o ponto da curva $y = \ln x$ mais próximo da origem.

E 3.7.10. (Matemática financeira) Um computador é vendido pelo valor a vista de R$2.000,00 ou em 1+15 prestações de R$200,00. Calcule a taxa de juros associada à venda a prazo.

E 3.7.11. (Matemática financeira) O valor de R$110.000,00 é financiado conforme a seguinte programa de pagamentos:

Mês	pagamento
1	20.000,00
2	20.000,00
3	20.000,00
4	19.000,00
5	18.000,00
6	17.000,00
7	16.000,00

Calcule a taxa de juros envolvida. A data do empréstimo é o mês zero.

E 3.7.12. (Controle de sistemas) Depois de acionado um sistema de aquecedores, a temperatura em um forno evolui conforme a seguinte equação

$$T(t) = 500 - 800e^{-t} + 600e^{-t/3}. \tag{3.223}$$

onde T é a temperatura em Kelvin e t é tempo em horas.

a) Obtenha analiticamente o valor de $\lim_{t\to\infty} T(t)$.

b) Obtenha analiticamente o valor máximo de $T(t)$ e o instante de tempo quando o máximo acontece

c) Obtenha numericamente com precisão de minutos o tempo decorrido até que a temperatura passe pela primeira vez pelo valor de equilíbrio obtido no item a.

c) Obtenha numericamente com precisão de minutos a duração do período durante o qual a temperatura permanece pelo menos 20% superior ao valor de equilíbrio.

E 3.7.13. Encontre os pontos onde a elipse que satisfaz $\frac{x^2}{3} + y^2 = 1$ intersepta a parábola $y = x^2 - 2$.

E 3.7.14. (Otimização) Encontre a área do maior retângulo que é possível inscrever entre a curva $e^{-x^2}(1 + \cos(x))$ e o eixo $y = 0$.

E 3.7.15. (Otimização)Uma indústria consome energia elétrica de duas usinas fornecedoras. O custo de fornecimento em reais por hora como função da potência consumida em kW é dada pelas seguintes funções

$$C_1(x) = 500 + .27x + 4.1 \cdot 10^{-5}x^2 + 2.1 \cdot 10^{-7}x^3 + 4.2 \cdot 10^{-10}x^4 \tag{3.224}$$
$$C_2(x) = 1000 + .22x + 6.3 \cdot 10^{-5}x^2 + 8.5 \cdot 10^{-7}x^3 \tag{3.225}$$

Onde $C_1(x)$ e $C_2(x)$ são os custos de fornecimento das usinas 1 e 2, respectivamente. Calcule o custo mínimo da energia elétrica quando a potência total consumida é $1500kW$. Obs: Para um problema envolvendo mais de duas usinas, veja 5.1.12.

E 3.7.16. (Termodinâmica) A pressão de saturação (em bar) de um dado hidrocarboneto pode ser modelada pela equação de Antoine:

$$\ln\left(P^{sat}\right) = A - \frac{B}{T + C} \tag{3.226}$$

onde T é a temperatura e A, B e C são constantes dadas conforme a seguir:

Hidrocarboneto	A	B	C
N-pentano	9.2131	2477.07	-39.94
N-heptano	9.2535	2911.32	-56.51

a) Calcule a temperatura de bolha de uma mistura de N-pentano e N-heptano à pressão de 1.2bar quando as frações molares dos gases são $z_1 = z_2 = 0.5$. Para tal utilize a seguinte equação:

$$P = \sum_i z_i P_i^{sat} \tag{3.227}$$

b) Calcule a temperatura de orvalho de uma mistura de N-pentano e N-heptano à pressão de 1.2bar quando as frações molares dos gases são $z_1 = z_2 = 0.5$. Para tal utilize a seguinte equação:

$$\frac{1}{P} = \sum_i \frac{z_i}{P_i^{sat}} \tag{3.228}$$

E 3.7.17. Encontre os três primeiros pontos de mínimo da função

$$f(x) = e^{-x/11} + x\cos(2x) \tag{3.229}$$

para $x > 0$ com erro inferior a 10^{-7}.

Capítulo 4

Solução de sistemas lineares

Muitos problemas da engenharia, física e matemática estão associados à solução de sistemas de equações lineares. Nesse capítulo, tratamos de técnicas numéricas empregadas para obter a solução desses sistemas. Iniciamos por uma rápida revisão do método de eliminação gaussiana do ponto de vista computacional. No contexto de análise da propagação dos erros de arredondamento, introduzimos o método de eliminação gaussiana com pivotamento parcial, bem como, apresentamos o conceito de condicionamento de um sistema linear. Além disso, exploramos o conceito de complexidade de algoritmos em álgebra linear. Então, passamos a discutir sobre técnicas iterativas, mais especificamente, sobre os métodos de Jacobi e Gauss-Seidel.

Considere o sistema de equações lineares (escrito na forma algébrica)

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \end{aligned} \tag{4.1}$$

onde m é o número de equações e n é o número de incógnitas. Este sistema pode ser escrito na **forma matricial**

$$Ax = b \tag{4.2}$$

onde:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \text{ e } b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}, \tag{4.3}$$

onde A é chamada de **matriz dos coeficientes**, x de **vetor das incógnitas** e b de **vetor dos termos constantes**.

Definimos também a **matriz completa** (também chamada de **matriz estendida**) de um sistema como $Ax = b$ como $[A|b]$, isto é

$$[A|b] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{array}\right] \tag{4.4}$$

Salvo especificado ao contrário, assumiremos ao longo deste capítulo que a matriz dos coeficientes A é uma matriz real não singular (isto é, invertível).

Ao longo do capítulo, apresentamos algumas computações com `Python`. Nestas, assumiremos que a biblioteca numpy e seu módulo numpy.linalg estão carregados:

```
>>> from __future__ import division
>>> import numpy as np
>>> from numpy import linalg
```

Exemplo 4.0.1. Consideramos o seguinte sistema linear

$$\begin{aligned} x + y + z &= 1 \\ 4x + 4y + 2z &= 2 \\ 2x + y - z &= 0 \end{aligned} \tag{4.5}$$

. Na sua forma matricial, este sistema é escrito como

$$Ax = b \Leftrightarrow \underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 4 & 4 & 2 \\ 2 & 1 & -1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}_{\underline{x}} = \underbrace{\begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}}_{b} \tag{4.6}$$

.

A matriz estendida do sistema acima é

$$E := [A|b] = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 4 & 4 & 2 & 2 \\ 2 & 1 & -1 & 0 \end{bmatrix} \tag{4.7}$$

4.1 Eliminação gaussiana

A **eliminação gaussiana**, também conhecida como **escalonamento**, é um método para resolver sistemas lineares. Este método consiste em manipular o sistema através de determinadas operações elementares, transformando a matriz estendida do sistema em uma matriz triangular (chamada de **matriz escalonada do sistema**). Uma vez, triangularizado o sistema, a solução pode ser obtida via substituição regressiva. Naturalmente estas operações elementares devem preservar a solução do sistema e consistem em:

1. multiplicação de um linha por uma constante não nula.
2. substituição de uma linha por ela mesma somada a um múltiplo de outra linha.
3. permutação de duas linhas.

Exemplo 4.1.1. Resolva o sistema

$$\begin{aligned} x + y + z &= 1 \\ 4x + 4y + 2z &= 2 \\ 2x + y - z &= 0 \end{aligned} \tag{4.8}$$

pelo método de eliminação gaussiana.

Solução. A matriz estendida do sistema é escrita como

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 4 & 4 & 2 & 2 \\ 2 & 1 & -1 & 0 \end{bmatrix} \tag{4.9}$$

No primeiro passo, subtraímos da segunda linha o quádruplo da primeira e subtraímos da terceira linha o dobro da primeira linha:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & -2 & -2 \\ 0 & -1 & -3 & -2 \end{bmatrix} \tag{4.10}$$

No segundo passo, permutamos a segunda linha com a terceira:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -1 & -3 & -2 \\ 0 & 0 & -2 & -2 \end{bmatrix} \tag{4.11}$$

Neste momento, a matriz já se encontra na forma triangular (chamada de **matriz escalonada do sistema**). Da terceira linha, encontramos $-2z = -2$, ou seja, $z = 1$. Substituindo na segunda equação, temos $-y - 3z = -2$, ou seja, $y = -1$ e finalmente, da primeira linha, $x + y + z = 1$, resultando em $x = 1$.

$\Diamond$

Neste Exemplo 4.1.1, o procedimento de eliminação gaussiana foi usado para obtermos um sistema triangular (superior) equivalente ao sistema original. Este, por sua vez, nos permitiu calcular a solução do sistema, isolando cada variável, começando da última linha (última equação), seguindo linha por linha até a primeira.

Alternativamente, podemos continuar o procedimento de eliminação gaussiana, anulando os elementos da matriz estendida acima da diagonal principal. Isto nos leva a uma matriz estendida diagonal (chamada **matriz escalonada reduzida**), na qual a solução do sistema original aparece na última coluna.

Exemplo 4.1.2. No Exemplo 4.1.1, usamos o procedimento de eliminação gaussiana e obtivemos

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 4 & 4 & 2 & 2 \\ 2 & 1 & -1 & 0 \end{bmatrix}}_{\text{matriz estendida}} \sim \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -1 & -3 & -2 \\ 0 & 0 & -2 & -2 \end{bmatrix}}_{\text{matriz escalonada}}. \tag{4.12}$$

Agora, seguindo com o procedimento de eliminação gaussiana, buscaremos anular os elementos acima da diagonal principal. Começamos dividindo cada elemento da última linha pelo valor do elemento da sua diagonal, obtemos

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -1 & -3 & -2 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{4.13}$$

Então, somando da segunda linha o triplo da terceira e subtraindo da primeira a terceira linha, obtemos

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{4.14}$$

Fixamos, agora, na segunda linha. Dividimos esta linha pelo valor do elemento em sua diagonal, isto nos fornece

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{4.15}$$

Por fim, subtraímos da primeira linha a segunda, obtendo a matriz escalonada reduzida

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{4.16}$$

Desta matriz escalonada reduzida temos, imediatamente, $x = 1$, $y = -1$ e $z = 1$, como no Exemplo 4.1.1.

4.1.1 Eliminação gaussiana com pivotamento parcial

A eliminação gaussiana com **pivotamento parcial** consiste em fazer uma permutação de linhas de forma a escolher o maior pivô (em módulo) a cada passo.

Exemplo 4.1.3. Resolva o sistema

$$\begin{aligned} x + y + z &= 1 \\ 2x + y - z &= 0 \\ 2x + 2y + z &= 1 \end{aligned} \tag{4.17}$$

por eliminação gaussiana com pivotamento parcial.

Solução. A matriz estendida do sistema é

$$\begin{aligned}
\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & 0 \\ 2 & 2 & 1 & 1 \end{bmatrix} &\sim \begin{bmatrix} 2 & 1 & -1 & 0 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 \end{bmatrix} \\
&\sim \begin{bmatrix} 2 & 1 & -1 & 0 \\ 0 & 1/2 & 3/2 & 1 \\ 0 & 1 & 2 & 1 \end{bmatrix} \\
&\sim \begin{bmatrix} 2 & 1 & -1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 1/2 & 3/2 & 1 \end{bmatrix} \\
&\sim \begin{bmatrix} 2 & 1 & -1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 1/2 & 1/2 \end{bmatrix}
\end{aligned} \tag{4.18}$$

Encontramos $1/2z = 1/2$, ou seja, $z = 1$. Substituímos na segunda equação e temos $y + 2z = 1$, ou seja, $y = -1$ e, finalmente $2x + y - z = 0$, resultando em $x = 1$.

Em `Python`, podemos fazer estas computações da seguinte forma:

```
E = np.array([[1,1,1,1],
              [2,1,-1,0],
              [2,2,1,1]], dtype='double')
print(E)

#L2 <-> L1
aux = np.copy(E[1,:])
E[1,:] = np.copy(E[0,:])
E[0,:] = np.copy(aux)
print(E)

#zera E[1:2,0]
E[1,:] = E[1,:] - (E[1,0]/E[0,0])*E[0,:]
E[2,:] = E[2,:] - (E[2,0]/E[0,0])*E[0,:]
print(E)
```

```
#zera E[2,1]
E[2,:] = E[2,:] - (E[2,1]/E[1,1])*E[1,:]
print(E)

#sub. regressiva
x = np.zeros(3)
x[2] = E[2,3]/E[2,2];
x[1] = (E[1,3] - E[1,2]*x[2])/E[1,1];
x[0] = (E[0,3] - E[0,2]*x[2] - E[0,1]*x[1])/E[0,0]
print(x)
```

$\Diamond$

A técnica de eliminação gaussiana com pivotamento parcial ajuda a evitar a propagação dos erros de arredondamento. Vejamos o próximo exemplo.

Exemplo 4.1.4 (Problema com elementos com grande diferença de escala)**.** Resolva o seguinte sistema usando eliminação gaussiana sem e com pivotamento parcial. Discuta, em cada caso, o resultado frente a aritmética de ponto flutuante quando $0 < |\epsilon| \ll 1$.

$$\begin{bmatrix} \varepsilon & 2 \\ 1 & \varepsilon \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \end{bmatrix} \tag{4.19}$$

Solução. Vamos, primeiramente, executar a eliminação gaussiana sem pivotamento parcial para $\varepsilon \neq 0$ e $|\varepsilon| \ll 1$:

$$\left[\begin{array}{cc|c} \varepsilon & 2 & 4 \\ 1 & \varepsilon & 3 \end{array}\right] \sim \left[\begin{array}{cc|c} \varepsilon & 2 & 4 \\ 0 & \varepsilon - \frac{2}{\varepsilon} & 3 - \frac{4}{\varepsilon} \end{array}\right] \tag{4.20}$$

Temos

$$y = \frac{3 - 4/\varepsilon}{\varepsilon - 2/\varepsilon} \tag{4.21}$$

e

$$x = \frac{4 - 2y}{\varepsilon} \tag{4.22}$$

Observe que a expressão obtida para y se aproximada de 2 quando ε é pequeno:

$$y = \frac{3 - 4/\varepsilon}{\varepsilon - 2/\varepsilon} = \frac{3\varepsilon - 4}{\varepsilon^2 - 2} \longrightarrow \frac{-4}{-2} = 2, \quad \text{quando } \varepsilon \to 0. \tag{4.23}$$

Já expressão obtida para x depende justamente da diferença $2 - y$:

$$x = \frac{4 - 2y}{\varepsilon} = \frac{2}{\varepsilon}(2 - y) \tag{4.24}$$

Assim, quando ε é pequeno, a primeira expressão, implementada em um sistema de ponto flutuante de acurácia finita, produz $y = 2$ e, consequentemente, a expressão para x produz $x = 0$. Isto é, estamos diante um problema de cancelamento catastrófico.

Agora, quando usamos a eliminação gaussiana com pivotamento parcial, fazemos uma permutação de linhas de forma a escolher o maior pivô a cada passo:

$$\left[\begin{array}{cc|c} \varepsilon & 2 & 4 \\ 1 & \varepsilon & 3 \end{array}\right] \sim \left[\begin{array}{cc|c} 1 & \varepsilon & 3 \\ \varepsilon & 2 & 4 \end{array}\right] \sim \left[\begin{array}{cc|c} 1 & \varepsilon & 3 \\ 0 & 2 - \varepsilon^2 & 4 - 3\varepsilon \end{array}\right] \tag{4.25}$$

Continuando o procedimento, temos:

$$y = \frac{4 - 4\varepsilon}{2 - \varepsilon^2} \tag{4.26}$$

e

$$x = 3 - \varepsilon y \tag{4.27}$$

Observe que tais expressões são analiticamente idênticas às anteriores, no entanto, são mais estáveis numericamente. Quando ε converge a zero, y converge a 2, como no caso anterior. No entanto, mesmo que $y = 2$, a segunda expressão produz $x = 3 - \varepsilon y$, isto é, a aproximação $x \approx 3$ não depende mais de obter $2 - y$ com precisão. ♢

Exercícios resolvidos

ER 4.1.1. Resolva o seguinte sistema por eliminação gaussiana com pivotamento parcial.

$$\begin{aligned} 2y + 2z &= 8 \\ x + 2y + z &= 9 \\ x + y + z &= 6 \end{aligned} \tag{4.28}$$

Solução. A forma matricial do sistema dado é

$$\left[\begin{array}{ccc} 0 & 2 & 2 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{array}\right] \left[\begin{array}{c} x \\ y \\ z \end{array}\right] = \left[\begin{array}{c} 8 \\ 9 \\ 6 \end{array}\right] \tag{4.29}$$

Construímos, então, a matriz completa e seguimos com o procedimento de eliminação gaussiana com pivotamento parcial:

$$\left[\begin{array}{ccc|c} 0 & 2 & 2 & 8 \\ 1 & 2 & 1 & 9 \\ 1 & 1 & 1 & 6 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 2 & 1 & 9 \\ 0 & 2 & 2 & 8 \\ 1 & 1 & 1 & 6 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 2 & 1 & 9 \\ 0 & 2 & 2 & 8 \\ 0 & -1 & 0 & -3 \end{array}\right] \tag{4.30}$$

$$\sim \left[\begin{array}{ccc|c} 1 & 2 & 1 & 9 \\ 0 & 2 & 2 & 8 \\ 0 & 0 & 1 & 1 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 2 & 0 & 8 \\ 0 & 2 & 0 & 6 \\ 0 & 0 & 1 & 1 \end{array}\right] \tag{4.31}$$

$$\sim \left[\begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 2 & 0 & 6 \\ 0 & 0 & 1 & 1 \end{array}\right] \tag{4.32}$$

Portanto $x = 2$, $y = 3$ e $z = 1$.

$\Diamond$

Exercícios

E 4.1.1. Resolva o seguinte sistema de equações lineares

$$\begin{aligned} x + y + z &= 0 \\ x + 10z &= -48 \\ 10y + z &= 25 \end{aligned} \tag{4.33}$$

Usando eliminação gaussiana com pivotamento parcial (não use o computador para resolver essa questão).

E 4.1.2. Resolva o seguinte sistema de equações lineares

$$x + y + z = 0 \tag{4.38}$$
$$x + 10z = -48 \tag{4.39}$$
$$10y + z = 25 \tag{4.40}$$

Usando eliminação gaussiana com pivotamento parcial (não use o computador para resolver essa questão).

E 4.1.3. Calcule a inversa da matriz

$$A = \begin{bmatrix} 1 & 2 & -1 \\ -1 & 2 & 0 \\ 2 & 1 & -1 \end{bmatrix} \tag{4.41}$$

usando eliminação gaussiana com pivotamento parcial.

E 4.1.4. Demonstre que se $ad \neq bc$, então a matriz A dada por:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{4.42}$$

é inversível e sua inversa é dada por:

$$A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}. \tag{4.43}$$

4.2 Complexidade de algoritmos em álgebra linear

Nesta seção, discutiremos um importante conceito em teoria de algoritmos, a complexidade, isto é, uma medida do custo ou eficiência do algoritmo.

Dados dois algoritmos diferentes para resolver o mesmo problema, como podemos escolher qual desses algoritmos é o melhor? Se pensarmos em termos de **eficiência** (ou custo computacional), queremos saber qual desses algoritmos consome menos recursos para realizar a mesma tarefa.

Em geral podemos responder esta pergunta de duas formas: em termos de tempo ou de espaço.

Quando tratamos de **eficiência espacial**, queremos saber quanta memória (em geral RAM) é utilizada pelo algoritmo para armazenar os dados, sejam eles matrizes, vetores ou escalares.

Quando tratamos de **eficiência temporal**, queremos saber quanto tempo um algoritmo demanda para realizar determinada tarefa. Vamos nos concentrar neste segundo conceito, que em geral é o mais difícil de tratar.

Naturalmente o tempo vai depender do tipo de computador utilizado. É razoável pensar que o tempo vai ser proporcional ao número de operações de ponto

flutuante (flops) feitas pelo algoritmo (observe que o tempo total não depende apenas disso, mas também de outros fatores como memória, taxas de transferências de dados da memória para o cpu, redes,...). Entretanto vamos nos concentrar na contagem do número de operações (flops) para realizar determinada tarefa.

No passado (antes dos anos 80), os computadores demoravam mais tempo para realizar operações como multiplicação e divisão, se comparados à adição ou à subtração. Assim, em livros clássicos eram contados apenas o custo das operações $\times$ e $/$. Nos computadores atuais as quatro operações básicas demandam aproximadamente o mesmo tempo. Não obstante, como na maioria dos algoritmos de álgebra linear, o número de multiplicações e divisões é proporcional ao número somas e subtrações (pois a maioria dessas operações podem ser escritas como a combinações de produtos internos), é justificável dizer que o tempo de computação continua podendo ser estimado pelo número de multiplicações e divisões. Desta forma, na maior parte deste material, levaremos em conta somente multiplicações e divisões, a não ser que mencionado o contrário.

Teremos em mente que a ideia é estimar o custo quando lidamos com vetores e matrizes muito grande, isto é, o custo quando estas dimensões crescem infinitamente.

Exemplo 4.2.1 (Produto escalar-vetor)**.** Qual o custo para multiplicar um escalar por um vetor?

Solução. Seja $a \in \mathbb{R}$ e $\boldsymbol{x} \in \mathbb{R}^n$, temos que

$$a\boldsymbol{x} = [a \times x_1, a \times x_2, ..., a \times x_n] \tag{4.44}$$

usando n multiplicações, ou seja, um custo computacional, C, de

$$C = n \text{ flops.} \tag{4.45}$$

$\Diamond$

Exemplo 4.2.2 (Produto vetor-vetor)**.** Qual o custo para calcular o produto interno $\boldsymbol{x} \cdot \boldsymbol{y}$?

Solução. Sejam $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^n$, temos que

$$\boldsymbol{x} \cdot \boldsymbol{y} = x_1 \times y_1 + x_2 \times y_2 + ... + x_n \times y_n \tag{4.46}$$

São realizadas n multiplicações (cada produto x_i por y_i) e $n-1$ somas, ou seja, o custo total de operações é de

$$C := (n) + (n-1) = 2n - 1 \text{ flops} \tag{4.47}$$

$\Diamond$

Exemplo 4.2.3 (Produto matriz-vetor)**.** Qual o custo para calcular o produto de matriz por vetor $A\boldsymbol{x}$?

Solução. Sejam $A \in \mathbb{R}^{n\times n}$ e $\boldsymbol{x} \in \mathbb{R}^n$, temos que

$$\begin{bmatrix} a_{11} & a_{12} & \cdots a_{1n} \\ \vdots & & \vdots \\ a_{n1} & & \cdots a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11} \times x_1 + a_{12}x_2 + ... + a_{1n} \times x_n \\ \vdots \\ a_{n1} \times x_1 + a_{n2}x_2 + ... + a_{nn} \times x_n \end{bmatrix} \tag{4.48}$$

Para obter o primeiro elemento do vetor do lado direito, devemos multiplicar a primeira linha de A pelo vetor coluna $\boldsymbol{x}$. Note que esse é exatamente o custo do produto vetor-vetor do exemplo anterior. Como o custo para cada elemento do vetor do lado direito é o mesmo e temos n elementos, teremos que o custo para multiplicar matriz-vetor é[1]

$$C := n \cdot (2n-1) = 2n^2 - n \text{ flops.} \tag{4.50}$$

À medida que $n \to \infty$, temos

$$\mathcal{O}(2n^2 - n) = \mathcal{O}(2n^2) = \mathcal{O}(n^2) \text{ flops.} \tag{4.51}$$

$\Diamond$

Exemplo 4.2.4 (Produto matriz-matriz)**.** Qual o custo para calcular o produto de duas matrizes A e B?

Solução. Sejam $A, B \in \mathbb{R}^{n\times n}$ temos que

$$\begin{bmatrix} a_{11} & a_{12} & \cdots a_{1n} \\ \vdots & & \vdots \\ a_{n1} & & \cdots a_{nn} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \cdots a_{1n} \\ \vdots & & \vdots \\ b_{n1} & & \cdots b_{nn} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & \cdots c_{1n} \\ \vdots & & \vdots \\ c_{n1} & & \cdots c_{nn} \end{bmatrix} \tag{4.52}$$

onde o elemento d_{ij} é o produto da linha i de A pela coluna j de B,

$$d_{ij} = a_{i1} \times b_{1j} + a_{i2} \times b_{2j} + ... + a_{i2} \times b_{2j} \tag{4.53}$$

[1]Contando apenas multiplicações/divisões obtemos

$$n \cdot \mathcal{O}(n) = \mathcal{O}(n^2) \text{ flops.} \tag{4.49}$$

Note que este produto tem o custo do produto vetor-vetor, ou seja, $2n-1$. Como temos $n \times n$ elementos em D, o custo total para multiplicar duas matrizes é[2]

$$C = n \times n \times (2n-1) = 2n^3 - n^2 \text{ flops.} \tag{4.55}$$

$\Diamond$

4.3 Sistemas triangulares

Considere um sistema linear onde a matriz é triangular superior, ou seja,

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \tag{4.56}$$

tal que todos elementos abaixo da diagonal são iguais a zero.

Podemos resolver esse sistema iniciando pela última equação e isolando x_n obtemos

$$x_n = b_n/a_{nn} \tag{4.57}$$

Substituindo x_n na penúltima equação

$$a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n = b_{n-1} \tag{4.58}$$

e isolando x_{n-1} obtemos

$$x_{n-1} = (b_{n-1} - a_{n-1,n}x_n)/a_{n-1,n-1} \tag{4.59}$$

e continuando desta forma até a primeira equação obteremos

$$x_1 = (b_1 - a_{12}x_2 \cdots - a_{1n}x_n)/a_{11}. \tag{4.60}$$

De forma geral, temos que

$$x_i = (b_i - a_{i,i+1}x_{i+1} \cdots - a_{i,n}x_n)/a_{i,i}, \quad i = 2, \dots, n. \tag{4.61}$$

[2]Contando apenas $\times$ e / obtemos

$$n \times n \times (n) = n^3 \text{ flops.} \tag{4.54}$$

4.4 Fatoração LU

Considere um sistema linear $Ax = b$, onde a matriz A é densa[3]. A fim de resolver o sistema, podemos fatorar a matriz A como o produto de uma matriz L triangular inferior e uma matriz U triangular superior, ou seja, $A = LU$.

Sendo assim, o sistema pode ser reescrito da seguinte forma:

$$Ax = b \tag{4.62}$$
$$(LU)x = b \tag{4.63}$$
$$L(Ux) = b \tag{4.64}$$
$$Ly = b \quad \text{e} \quad Ux = y \tag{4.65}$$

Isto significa que, ao invés de resolvermos o sistema original, podemos resolver o sistema triangular inferior $Ly = b$ e, então, o sistema triangular superior $Ux = y$, o qual nos fornece a solução de $Ax = b$.

A matriz U da fatoração[4] LU é a matriz obtida ao final do escalonamento da matriz A.

A matriz L é construída a partir da matriz identidade I, ao longo do escalonamento de A. Os elementos da matriz L são os múltiplos do primeiro elemento da linha de A a ser zerado dividido pelo pivô acima na mesma coluna.

Por exemplo, para zerar o primeiro elemento da segunda linha de A, calculamos

$$L_{21} = A_{21}/A_{11} \tag{4.66}$$

e fazemos

$$A_{2,:} \Leftarrow A_{2,:} - L_{21}A_{1,:} \tag{4.67}$$

Note que denotamos $A_{i,:}$ para nos referenciarmos a linha i de A. Da mesma forma, se necessário usaremos $A_{:,j}$ para nos referenciarmos a coluna j de A.

Para zerar o primeiro elemento da terceira linha de A, temos

$$L_{31} = A_{31}/A_{11} \tag{4.68}$$

e fazemos

$$A_{3,:} \Leftarrow A_{3,:} - L_{31}A_{1,:} \tag{4.69}$$

até chegarmos ao último elemento da primeira coluna de A.

Repetimos o processo para as próximas colunas, escalonando a matriz A e coletando os elementos L_{ij} abaixo da diagonal[5].

[3] Diferentemente de uma matriz esparsa, uma matriz densa possui a maioria dos elementos diferentes de zero.

[4] Não vamos usar pivotamento nesse primeiro exemplo.

[5] Perceba que a partir da segunda coluna para calcular L_{ij} não usamos os elementos de A, mas os elementos da matriz A em processo de escalonamento

Exemplo 4.4.1. Use a fatoração LU para resolver o seguinte sistema linear:

$$\begin{aligned} x_1 + x_2 + x_3 &= -2 \\ 2x_1 + x_2 - x_3 &= 1 \\ 2x_1 - x_2 + x_3 &= 3 \end{aligned} \tag{4.70}$$

Solução. Começamos fatorando a matriz A dos coeficientes deste sistema:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & -1 \\ 2 & -1 & 1 \end{bmatrix} . = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{I_{3,3}} \underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & -1 \\ 2 & -1 & 1 \end{bmatrix}}_{A} \tag{4.71}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & -1 & -3 \\ 0 & -3 & -1 \end{bmatrix} \tag{4.72}$$

$$= \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 2 & 3 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 0 & -1 & -3 \\ 0 & 0 & 8 \end{bmatrix}}_{U} \tag{4.73}$$

$$\tag{4.74}$$

Completada a fatoração LU, resolvemos, primeiramente, o sistema $Ly = b$:

$$\begin{aligned} y_1 &= -2 \\ 2y_1 + y_2 &= 1 \\ 2y_1 + 3y_2 + y_3 &= 3 \end{aligned} \tag{4.75}$$

o qual no fornece $y_1 = -2$, $y_2 = 5$ e $y_3 = -8$. Por fim, obtemos a solução resolvendo o sistema $Ux = y$:

$$\begin{aligned} x_1 + x_2 + x_3 &= -2 \\ -x_2 - 3x_3 &= 5 \\ 8x_3 &= -8 \end{aligned} \tag{4.76}$$

o qual fornece $x_3 = -1$, $x_2 = -2$ e $x_1 = 1$. ◊

4.4.1 Código Python: Fatoração LU

Em `Python`, podemos implementar o algoritmo para fatoração LU da seguinte forma:

```
def fatoraLU(A):
    U = np.copy(A)
    n = np.shape(U)[0]
    L = np.eye(n)
    for j in np.arange(n-1):
        for i in np.arange(j+1,n):
            L[i,j] = U[i,j]/U[j,j]
            for k in np.arange(j+1,n):
                U[i,k] = U[i,k] - L[i,j]*U[j,k]
            U[i,j] = 0
    return L, U
```

Observação 4.4.1. O custo computacional do algoritmo da fatoração LU é

$$\frac{2n^3}{3} - \frac{n^2}{2} - \frac{n}{6} \text{ flops.} \tag{4.77}$$

4.4.2 Custo computacional para resolver um sistema linear usando fatoração LU

Para calcularmos o custo computacional de um algoritmo completo, uma estratégia é separar o algoritmo em partes menores, mais fáceis de analisar.

Para resolver o sistema, devemos primeiro fatorar a matriz A nas matrizes L e U. Vimos que o custo é

$$\frac{2n^3}{3} - \frac{n^2}{2} - \frac{n}{6} \text{ flops.} \tag{4.78}$$

Depois devemos resolver os sistemas $Ly = b$ e $Ux = y$. O custo de resolver os dois sistemas é (devemos contar duas vezes)

$$2n^2 \text{ flops.} \tag{4.79}$$

Somando esses 3 custos, temos que o custo para resolver um sistema linear usando fatoração LU é

$$\frac{2n^3}{3} + \frac{3n^2}{2} - \frac{n}{6} \text{ flops.} \tag{4.80}$$

Quando n cresce, prevalessem os termos de mais alta ordem, ou seja,

$$\mathcal{O}(\frac{2n^3}{3} + \frac{3n^2}{2} - \frac{n}{6}) = \mathcal{O}(\frac{2n^3}{3} + \frac{3n^2}{2}) = \mathcal{O}(\frac{2n^3}{3}) \tag{4.81}$$

4.4.3 Custo para resolver m sistemas lineares

Devemos apenas multiplicar m pelo custo de resolver um sistema linear usando fatoração LU, ou seja, o custo será

$$m(\frac{2n^3}{3} + \frac{3n^2}{2} - \frac{n}{6}) = \frac{2mn^3}{3} + \frac{3mn^2}{2} - \frac{mn}{6} \tag{4.82}$$

e com $m = n$ temos

$$\frac{2n^4}{3} + \frac{3n^3}{2} - \frac{n^2}{6}. \tag{4.83}$$

Porém, se estivermos resolvendo n sistemas com *a mesma matriz* A (e diferente lado direito $\boldsymbol{b}$ para cada sistema) podemos fazer a fatoração LU uma única vez e contar apenas o custo de resolver os sistemas triangulares obtidos.

Custo para fatoração LU de A: $\frac{2n^3}{3} - \frac{n^2}{2} - \frac{n}{6}$.

Custo para resolver m sistemas triangulares inferiores: mn^2.

Custo para resolver m sistemas triangulares superiores: mn^2.

Somando esses custos obtemos

$$\frac{2n^3}{3} - \frac{n^2}{2} - \frac{n}{6} + 2mn^2 \tag{4.84}$$

que quando $m = n$ obtemos

$$\frac{8n^3}{3} - \frac{n^2}{2} - \frac{n}{6} \text{ flops.} \tag{4.85}$$

4.4.4 Custo para calcular a matriz inversa de A

Como vemos em Álgebra Linear, um método para obter a matriz A^{-1} é realizar o escalonamento da matriz $[A|I]$ onde I é a matriz identidade. Ao terminar o escalonamento, o bloco do lado direito conterá A^{-1}.

Isto é equivalente a resolver n sistemas lineares com a mesma matriz A e os vetores da base canônica $\boldsymbol{e}_i = [0,...,0,1,0,....0]^T$ tal que

$$A\boldsymbol{x}_i = \boldsymbol{e}_i, \qquad i = 1 : n \tag{4.86}$$

onde $\boldsymbol{x}_i$ serão as colunas da matriz A inversa, já que $AX = I$.

O custo para resolver esses n sistemas lineares foi calculado na seção anterior como

$$\frac{8n^3}{3} - \frac{n^2}{2} - \frac{n}{6}. \tag{4.87}$$

Exemplo 4.4.2. Qual o melhor método para resolver um sistema linear: via fatoração LU ou calculando a inversa de A e obtendo $x = A^{-1}b$?

4.5 Método da matriz tridiagonal

O método da matriz tridiagonal ou algoritmo de Thomas[6] ou ainda TDMA (do inglês *tridiagonal matrix algorithm*) é o caso particular da eliminação gaussiana aplicada a matrizes tridiagonais.

Uma matriz tridiagonal é uma matriz quadrada cujos únicos elementos não nulos estão na diagonal principal e nas diagonais imediatamente acima e abaixo da principal. Um sistema tridiagonal é um sistema de equações lineares cuja matriz associada é tridiagonal, conforme a seguir:

$$\begin{bmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}. \tag{4.88}$$

Observamos que não é necessário armazenar todos os n^2 elementos da matriz em memória, sendo suficiente armazenar os vetores a_n, b_n e c_n. Por conveniência, a partir daqui, definiremos os elementos inexistentes na matriz a_1 e c_n como zero:

$$a_1 = c_n = 0. \tag{4.89}$$

O algoritmo para a solução do sistema tridiagonal (4.88) pelo algoritmo de Thomas é dada pelas seguintes expressões:

$$c'_i = \begin{cases} \frac{c_i}{b_i}, & i = 1 \\ \frac{c_i}{b_i - a_i c'_{i-1}}, & i = 2, 3, \ldots, n-1 \end{cases} \tag{4.90}$$

e

$$d'_i = \begin{cases} \frac{d_i}{b_i}, & i = 1 \\ \frac{d_i - a_i d'_{i-1}}{b_i - a_i c'_{i-1}}, & i = 2, 3, \ldots, n. \end{cases} \tag{4.91}$$

Finalmente a solução final é obtida por substituição reversa:

$$x_n = d'_n \tag{4.92}$$

$$x_i = d'_i - c'_i x_{i+1}, \qquad i = n-1, n-2, \ldots, 1. \tag{4.93}$$

Teorema 4.5.1. *A aplicação da eliminação gaussiana sem pivotamento ao sistema (4.88) produz o algoritmo dado em (4.90) e (4.92).*

[6] Llewellyn Hilleth Thomas (21 de outubro de 1903 – 20 de abril de 1992) foi um matemático e físico britânico.

Demonstração. O primeiro passo consiste em dividir todos os elementos da primeira linha de (4.88) por b_1:

$$\begin{bmatrix} 1 & c_1' & & & \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1' \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}, \tag{4.94}$$

onde $c_1' = \frac{c_1}{b_1}$ e $d_1' = \frac{d_1}{b_1}$.

O segundo passo consiste em substituir a segunda linha por ela mesma subtraída da linha 1 multiplicada por a_2 ($l_2 \leftarrow l_2 - a_2 l_1$):

$$\begin{bmatrix} 1 & c_1' & & & \\ 0 & b_2 - a_2 c_1' & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1' \\ d_2 - a_2 d_1' \\ d_3 \\ \vdots \\ d_n \end{bmatrix}. \tag{4.95}$$

Em seguida, dividimos a segunda linha por $b_2 - a_2 c_1'$, a fim de normalizar a diagonal principal:

$$\begin{bmatrix} 1 & c_1' & & & \\ 0 & 1 & c_2' & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1' \\ d_2' \\ d_3 \\ \vdots \\ d_n \end{bmatrix}. \tag{4.96}$$

onde $c_2' = \frac{c_2}{b_2 - a_2 c_1'}$ e $d_2' = \frac{d_2 - a_2 d_1'}{b_2 - a_2 c_1'}$.

O próximo passo consiste em substituir a terceira linha por ela mesma subtraída

da linha 2 multiplicada por a_3 ($l_3 \leftarrow l_3 - a_3 l_2$):

$$\begin{bmatrix} 1 & c'_1 & & & \\ 0 & 1 & c'_2 & & \\ & 0 & b_3 - a_3 c'_2 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d'_1 \\ d'_2 \\ d_3 - a_3 d'_2 \\ \vdots \\ d_n \end{bmatrix}. \tag{4.97}$$

A fim de normalizar o elemento da diagonal da terceira linha, dividimos toda a linha por $d_3 - a_3 d'_2$:

$$\begin{bmatrix} 1 & c'_1 & & & \\ 0 & 1 & c'_2 & & \\ & 0 & 1 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d'_1 \\ d'_2 \\ d'_3 \\ \vdots \\ d_n \end{bmatrix}. \tag{4.98}$$

Este procedimento é realizado até que se atinja a última linha e temos o seguinte sistema:

$$\begin{bmatrix} 1 & c'_1 & & & \\ 0 & 1 & c'_2 & & \\ & 0 & 1 & \ddots & \\ & & \ddots & \ddots & c'_{n-1} \\ & & & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d'_1 \\ d'_2 \\ d'_3 \\ \vdots \\ d'_n \end{bmatrix}. \tag{4.99}$$

Neste estágio, podemos encontrar os x_n através de substituição reversa, isto é: a última linha diz

$$x_n = d'_n. \tag{4.100}$$

A penúltima linha diz

$$x_{n-1} + c'_{n-1} x_n = d'_{n-1} \Longrightarrow x_{n-1} = d'_{n-1} - c'_{n-1} x_n. \tag{4.101}$$

Esse mesmo procedimento aplicado à linha $i = 1, \ldots n-1$, nos dá

$$x_i = d'_i - c'_i x_{i+1}. \tag{4.102}$$

□

Exemplo 4.5.1. Considere a resolução do seguinte sistema tridiagonal pelo algoritmo de Thomas:

$$\begin{bmatrix} 2 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \\ 0 \\ 0 \\ 2 \end{bmatrix}. \tag{4.103}$$

Primeiramente identificamos os vetores a, b, c e d:

$$a = (0, 1, 1, 1, 1) \tag{4.104}$$
$$b = (2, 2, 2, 2, 2) \tag{4.105}$$
$$c = (1, 1, 1, 1, 0) \tag{4.106}$$
$$d = (4, 4, 0, 0, 2) \tag{4.107}$$

Agora, calculamos os vetores c' e d':

$$c'_1 = \frac{c_1}{b_1} = \frac{1}{2} \tag{4.108}$$
$$c'_2 = \frac{c_2}{b_2 - a_2 c'_1} = \frac{1}{2 - 1 \cdot \frac{1}{2}} = \frac{2}{3} \tag{4.109}$$
$$c'_3 = \frac{c_3}{b_3 - a_3 c'_2} = \frac{1}{2 - 1 \cdot \frac{2}{3}} = \frac{3}{4} \tag{4.110}$$
$$c'_4 = \frac{c_4}{b_4 - a_4 c'_3} = \frac{1}{2 - 1 \cdot \frac{3}{4}} = \frac{4}{5} \tag{4.111}$$
$$d'_1 = \frac{d_1}{b_1} = \frac{4}{2} = 2 \tag{4.112}$$
$$d'_2 = \frac{d_2 - a_2 d'_1}{b_2 - a_2 c'_1} = \frac{4 - 1 \cdot 2}{2 - 1 \cdot \frac{1}{2}} = \frac{4}{3} \tag{4.113}$$
$$d'_3 = \frac{d_3 - a_3 d'_2}{b_3 - a_3 c'_2} = \frac{0 - 1 \cdot \frac{4}{3}}{2 - 1 \cdot \frac{2}{3}} = -1 \tag{4.114}$$
$$d'_4 = \frac{d_4 - a_4 d'_3}{b_4 - a_4 c'_3} = \frac{0 - 1 \cdot (-1)}{2 - 1 \cdot \frac{3}{4}} = \frac{4}{5} \tag{4.115}$$
$$d'_5 = \frac{d_5 - a_5 d'_4}{b_5 - a_5 c'_4} = \frac{2 - 1 \cdot \frac{4}{5}}{2 - 1 \cdot \frac{4}{5}} = 1 \tag{4.116}$$

Finalmente, calculamos o vetor x:

$$x_5 = d'_5 = 1 \tag{4.117}$$

$$x_4 = d'_4 - c'_4 \cdot x_5 = \frac{4}{5} - \frac{4}{5} \cdot 1 = 0 \tag{4.118}$$

$$x_3 = d'_3 - c'_3 \cdot x_4 = -1 - \frac{3}{4} \cdot 0 = -1 \tag{4.119}$$

$$x_2 = d'_2 - c'_2 \cdot x_3 = \frac{4}{3} - \frac{2}{3} \cdot (-1) = 2 \tag{4.120}$$

$$x_1 = d'_1 - c'_1 \cdot x_2 = 2 - \frac{1}{2} \cdot 2 = 1 \tag{4.121}$$

E assim, obtemos o vetor $x = [1, 2, -1, 0, 1]$.

Código Python: Método da matriz tridiagonal

```
import numpy as np

def TDMA(a,b,c,d):
    #preliminares
    a = a.astype('double')
    b = b.astype('double')
    c = c.astype('double')
    d = d.astype('double')

    #recupera ordem do sistema
    n=np.shape(a)[0]

    #inicializa vetores auxiliares
    cl=np.zeros(n)
    dl=np.zeros(n)
    x=np.zeros(n)

    #calcula cl e dl
    cl[0]=c[0]/b[0]
    for i in np.arange(1,n-1,1):
       cl[i]=c[i]/(b[i]-a[i]*cl[i-1])

    dl[0]=d[0]/b[0]
    for i in np.arange(1,n,1):
       dl[i]=(d[i]-a[i]*dl[i-1])/(b[i]-a[i]*cl[i-1])
```

```
    #faz substituicao reversa para obter a solucao x
    x[n-1]=dl[n-1]
    for i in np.arange(n-2,-1,-1):
       x[i]=dl[i]-cl[i]*x[i+1]

    return x
```

Nesse código, usou-se `cl` e `dl` para denotar c' e d'. Observe que se for desnecessário preservar os valores originais dos vetores c e d, eles podem, com economia de memória e simplicidade de código, ser sobrescritos pelos vetores c' e d', respectivamente. Eis uma nova implementação:

```
import numpy as np

def TDMA(a,b,c,d):
    #preliminares
    a = a.astype('double')
    b = b.astype('double')
    c = c.astype('double')
    d = d.astype('double')

    #recupera ordem do sistema
    n=np.shape(a)[0]

    #inicializa vetor x
    x=np.zeros(n)

    #calcula cl e dl sobrescrevendo-os em c e d
    c[0]=c[0]/b[0]
    for i in np.arange(1,n-1,1):
       c[i]=c[i]/(b[i]-a[i]*c[i-1])

    d[0]=d[0]/b[0]
    for i in np.arange(1,n,1):
       d[i]=(d[i]-a[i]*d[i-1])/(b[i]-a[i]*c[i-1])

    #faz substituicao reversa para obter a solucao x
    x[n-1]=d[n-1]
```

```
    for i in np.arange(n-2,-1,-1):
       x[i]=d[i]-c[i]*x[i+1]

    return x
```

A solução do sistema do Exemplo 4.5.1 pode ser obtida através dos seguintes comandos:

```
>>>a=np.array([1,1,1,1,1])
>>>b=np.array([2,2,2,2,2])
>>>c=np.array([1,1,1,1,1])
>>>d=np.array([4,4,0,0,2])
>>>TDMA(a,b,c,d)
```

E 4.5.1. Considere o problema linear tridiagonal dado por

$$\begin{bmatrix} 5 & 4 & 0 & 0 & 0 & 0 \\ 1 & 3 & 1 & 0 & 0 & 0 \\ 0 & 2 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 2 & 3 & 2 \\ 0 & 0 & 0 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 13 \\ 10 \\ 20 \\ 16 \\ 35 \\ 17 \end{bmatrix}. \tag{4.122}$$

Identifique os vetores a, b, c e d relativos ao algoritmo da matriz tridiagonal. Depois resolva o sistema usando o computador.

E 4.5.2. Considere o seguinte sistema de equações lineares:

$$\begin{aligned} x_1 - x_2 &= 0 \\ -x_{j-1} + 5x_j - x_{j+1} &= \cos(j/10), \ \ 2 \leq j \leq 10 \\ x_{11} &= x_{10}/2 \end{aligned} \tag{4.128}$$

Identifique os vetores a, b, c e d relativos ao algoritmo da matriz tridiagonal no sistema linear dado. Depois resolva o sistema usando o computador. Veja também Exercício 4.7.4

4.6 Condicionamento de sistemas lineares

Quando lidamos com matrizes no corpo do números reais (ou complexos), existem apenas duas alternativas: i) a matriz é inversível; ii) a matriz não é inversível e, neste caso, é chamada de matriz singular. Ao lidar com a aritmética de precisão finita, encontramos uma situação mais sutil: alguns problema lineares são mais difíceis de serem resolvidos, pois os erros de arredondamento se propagam de forma mais significativa que em outros problemas. Neste caso falamos de problemas bem-condicionados e mal-condicionados. Intuitivamente falando, um problema bem-condicionado é um problema em que os erros de arredondamento se propagam de forma menos importante; enquanto problemas mal-condicionados são problemas em que os erros se propagam de forma mais relevante.

Um caso típico de sistema mal-condicionado é aquele cujos coeficiente estão muito próximos ao de um problema singular. Considere o seguinte exemplo:

Exemplo 4.6.1. Observe que o sistema

$$\begin{bmatrix} 71 & 41 \\ \lambda & 30 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 100 \\ 70 \end{bmatrix} \tag{4.136}$$

é impossível quando $\lambda = \frac{71\times 30}{41} \approx 51{,}95122$.

Considere os próximos três sistemas:

a) $\begin{bmatrix} 71 & 41 \\ 51 & 30 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 100 \\ 70 \end{bmatrix}$, com solução $\begin{bmatrix} 10/3 \\ -10/3 \end{bmatrix}$,

b) $\begin{bmatrix} 71 & 41 \\ 52 & 30 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 100 \\ 70 \end{bmatrix}$, com solução $\begin{bmatrix} -65 \\ 115 \end{bmatrix}$,

c) $\begin{bmatrix} 71 & 41 \\ 52 & 30 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 100{,}4 \\ 69{,}3 \end{bmatrix}$, com solução $\begin{bmatrix} -85{,}35 \\ 150{,}25 \end{bmatrix}$.

Pequenas variações nos coeficientes das matrizes fazem as soluções ficarem bem distintas, isto é, pequenas variações nos dados de entrada acarretaram em grandes variações na solução do sistema. Quando isso acontece, dizemos que o problema é mal-condicionado.

Precisamos uma maneira de medir essas variações. Como os dados de entrada e os dados de saída são vetores (ou matrizes), precisamos introduzir as definições de norma de vetores e matrizes.

4.6.1 Norma de vetores

Definimos a **norma** L^p, $1 \leq p \leq \infty$, de um vetor em $v = (v_1, v_2, \ldots, v_n) \in \mathbb{R}^n$ por:

$$\|v\|_p := \left(\sum_{i=1}^{n} |v_i|^p \right)^{1/p} = (|v_1|^p + |v_2|^p + \cdots + |v_n|^p)^{1/p}, \quad 1 \leq p < \infty. \tag{4.137}$$

Para $p = \infty$, definimos a norma L^∞ (**norma do máximo**) por:

$$\|v\|_\infty = \max_{1 \leq j \leq n} \{|v_j|\}. \tag{4.138}$$

Proposição 4.6.1 (Propriedades de normas)**.** *Sejam dados $\alpha \in \mathbb{R}$ um escalar e os vetores u,$v \in \mathbb{R}^n$. Então, para cada $1 \leq p \leq \infty$, valem as seguintes propriedades:*

a) $\|u\|_p = 0 \Leftrightarrow u = 0$.

b) $\|\alpha u\|_p = |\alpha| \, \|u\|_p$.

c) $\|u + v\|_p \leq \|u\|_p + \|v\|_p$ *(**desigualdade triangular**).*

d) $\|u\|_p \to \|u\|_\infty$ *quando* $p \to \infty$.

Demonstração. Demonstramos cada item em separado.

a) Se $u = 0$, então segue imediatamente da definição da norma L^p, $1 \leq p \leq \infty$, que $\|u\|_p = 0$. Reciprocamente, se $\|u\|_\infty = 0$, então, para cada $i = 1, 2, \ldots, n$, temos:

$$|u_i| \leq \max_{1 \leq j \leq n} \{|u_j|\} = \|u\|_\infty = 0 \Rightarrow u_i = 0. \tag{4.139}$$

Isto é, $u = 0$. Agora, se $\|u\|_p = 0$, $1 \leq p < \infty$, então:

$$0 = \|u\|_p^p := \sum_{i=1}^{n} |u_i|^p \leq n \, \|u\|_\infty \Rightarrow \|u\|_\infty = 0. \tag{4.140}$$

Logo, pelo resultado para a norma do máximo, concluímos que $u = 0$.

b) Segue imediatamente da definição da norma L^p, $1 \leq p \leq \infty$.

c) Em construção ...

d) Em construção ...

□

Exemplo 4.6.2. Calcule a norma L^1, L^2 e L^∞ do vetor coluna $v = (1, 2, -3, 0)$.

Solução.

$$\|v\|_1 = 1+2+3+0 = 6 \tag{4.141}$$
$$\|v\|_2 = \sqrt{1+2^2+3^2+0^2} = \sqrt{14} \tag{4.142}$$
$$\|v\|_\infty = \max\{1,2,3,0\} = 3 \tag{4.143}$$

$\Diamond$

4.6.2 Norma de matrizes

Definimos a norma induzida L^p de uma matriz $A = [a_{i,j}]_{i,j=1}^{n,n}$ da seguinte forma:

$$\|A\|_p = \sup_{\|v\|_p=1} \|Av\|_p, \tag{4.144}$$

ou seja, a norma p de uma matriz é o máximo valor assumido pela norma de Av entre todos os vetores de norma unitária.

Temos as seguintes propriedades, se A e B são matrizes, I é a matriz identidade, v é um vetor e λ é um real (ou complexo):

$$\|A\|_p = 0 \iff A = 0 \tag{4.145}$$
$$\|\lambda A\|_p = |\lambda| \, \|A\|_p \tag{4.146}$$
$$\|A+B\|_p \leq \|A\|_p + \|B\|_p \quad \text{(desigualdade do triângulo)} \tag{4.147}$$
$$\|Av\|_p \leq \|A\|_p \, \|v\|_p \tag{4.148}$$
$$\|AB\|_p \leq \|A\|_p \, \|B\|_p \tag{4.149}$$
$$\|I\|_p = 1 \tag{4.150}$$
$$1 = \|I\|_p = \|AA^{-1}\|_p \leq \|A\|_p \, \|A^{-1}\|_p \quad \text{(se A é inversível)} \tag{4.151}$$

Casos especiais:

$$\|A\|_1 = \max_{j=1}^{n} \sum_{i=1}^{n} |a_{ij}| \tag{4.152}$$
$$\|A\|_2 = \sqrt{\max\{|\lambda| : \lambda \in \sigma(AA^*)\}} \tag{4.153}$$
$$\|A\|_\infty = \max_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}| \tag{4.154}$$

onde $\sigma(M)$ é o conjunto de autovalores da matriz M.

Exemplo 4.6.3. Calcule as normas 1, 2 e ∞ da seguinte matriz:

$$A = \begin{bmatrix} 3 & -5 & 7 \\ 1 & -2 & 4 \\ -8 & 1 & -7 \end{bmatrix} \tag{4.155}$$

Solução.

$$\|A\|_1 = \max\{12, 8, 18\} = 18 \tag{4.156}$$
$$\|A\|_\infty = \max\{15, 7, 16\} = 16 \tag{4.157}$$
$$\|A\|_2 = \sqrt{\max\{0{,}5865124, 21{,}789128, 195{,}62436\}} = 13{,}98657 \tag{4.158}$$

Em `Python` podemos computar normas L^p's de matrizes usando a função numpy.linalg.norm. Neste exemplo, temos:

```
>>> A = np.array([[3,-5,7],
...               [1,-2,4],
...               [-8,1,-7]], dtype='double')
>>> np.linalg.norm(A,1)
18
>>> np.linalg.norm(A,np.inf)
16
>>> np.linalg.norm(A,2)
13.986577820518308
```

◊

4.6.3 Número de condicionamento

O condicionamento de um sistema linear é um conceito relacionado à forma como os erros se propagam dos dados de entrada para os dados de saída. No contexto de um sistema linear $Ax = y$, temos que a solução x depende dos dados de entrada y. Consideremos, então, o problema

$$A(x + \delta_x) = y + \delta_y \tag{4.159}$$

Aqui, δ_x representa uma variação (erro) em x e δ_y representa uma variação em y (erro). Temos:

$$Ax + A\delta_x = y + \delta_y \tag{4.160}$$

e, portanto,

$$A\delta_x = \delta_y. \tag{4.161}$$

Queremos avaliar a razão entre o erro relativo em x e o erro relativo em y, isto

é

$$\frac{\|\delta_x\| / \|x\|}{\|\delta_y\| / \|y\|} = \frac{\|\delta_x\|}{\|x\|} \frac{\|y\|}{\|\delta_y\|} \tag{4.162}$$

$$= \frac{\|A^{-1}\delta_y\|}{\|x\|} \frac{\|Ax\|}{\|\delta_y\|} \tag{4.163}$$

$$\leq \frac{\|A^{-1}\| \, \|\delta_y\|}{\|x\|} \frac{\|A\| \, \|x\|}{\|\delta_y\|} \tag{4.164}$$

$$= \|A\| \, \|A^{-1}\| \tag{4.165}$$

Definição 4.6.1 (Número de condicionamento)**.** *O número de condicionamento de uma matriz não-singular A é*

$$k_p(A) := \|A\|_p \, \|A^{-1}\|_p \tag{4.166}$$

Observação 4.6.1.

- O número de condicionamento depende da norma escolhida.
- O número de condicionamento da matriz identidade é 1.
- O número de condicionamento de qualquer matriz inversível é maior ou igual a 1.

Exemplo 4.6.4. No Exemplo 4.6.1 estudamos a solução de sistemas lineares com as seguintes matrizes de coeficientes:

$$A_1 = \begin{bmatrix} 71 & 41 \\ 51 & 30 \end{bmatrix} \quad \text{e} \quad A_2 = \begin{bmatrix} 71 & 41 \\ 52 & 30 \end{bmatrix}. \tag{4.167}$$

Calcule os números de condicionamento destes sistemas na norma L^p para $p = 1$, 2 e ∞.

Solução. Para a matriz A_1, temos:

$$\begin{aligned} k_1(A_1) &:= \|A_1\|\|A_1^{-1}\| \approx 350{,}36, \\ k_2(A_1) &:= \|A_2\|\|A_2^{-1}\| \approx 262{,}12, \\ k_\infty(A_1) &:= \|A_\infty\|\|A_\infty^{-1}\| \approx 350{,}36. \end{aligned} \tag{4.168}$$

Para a matriz A_2, temos:

$$\begin{aligned} k_1(A_2) &:= \|A_1\|_1\|A_1^{-1}\|_1 \approx 6888{,}0, \\ k_2(A_2) &:= \|A_1\|_2\|A_1^{-1}\|_2 \approx 5163{,}0, \\ k_\infty(A_2) &:= \|A_1\|_\infty\|A_1^{-1}\|_\infty \approx 6888{,}0. \end{aligned} \tag{4.169}$$

Em `Python`, podemos computar estes números de condicionamento para a matriz A_1 com o seguinte código:

```
A = np.array([[71,41],[51,30]],dtype='double')
print(np.linalg.cond(A,p=1))
print(np.linalg.cond(A,p=2))
print(np.linalg.cond(A,p=np.inf))
```

e, análogo para a matriz A_2. ♢

Exercícios

E 4.6.1. Calcule o valor de λ para o qual o problema

$$\begin{cases} 71x + 41y = 10 \\ \lambda x + 30y = 4 \end{cases} \tag{4.170}$$

é impossível, depois calcule os números de condicionamento com norma 1,2 e ∞ quando $\lambda = 51$ e $\lambda = 52$.

E 4.6.2. Calcule o número de condicionamento da matriz

$$A = \begin{bmatrix} 3 & -5 & 7 \\ 1 & -2 & 4 \\ -8 & 1 & -7 \end{bmatrix} \tag{4.171}$$

nas normas 1, 2 e ∞.

E 4.6.3. Calcule o número de condicionamento das matrizes

$$\begin{bmatrix} 71 & 41 \\ 52 & 30 \end{bmatrix} \tag{4.172}$$

e

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 4 & 5 & 5 \end{bmatrix} \tag{4.173}$$

usando as normas 1, 2 e ∞.

E 4.6.4. Usando a norma 1, calcule o número de condicionamento da matriz

$$A = \begin{bmatrix} 1 & 2 \\ 2+\varepsilon & 4 \end{bmatrix} \tag{4.174}$$

em função de ε quando $0 < \varepsilon < 1$. Interprete o limite $\varepsilon \to 0$.

E 4.6.5. Considere os sistemas:

$$\begin{cases} 100000x - 9999.99y = -10 \\ -9999.99x + 1000.1y = 1 \end{cases} \quad \text{e} \quad \begin{cases} 100000x - 9999.99y = -9.999 \\ -9999.99x + 1000.1y = 1.01 \end{cases} \tag{4.175}$$

Encontre a solução de cada um e discuta.

E 4.6.6. Considere os vetores de 10 entradas dados por

$$x_j = \operatorname{sen}(j/10), \quad y_j = j/10 \quad z_j = j/10 - \frac{(j/10)^3}{6}, \ j = 1, \ldots, 10 \tag{4.176}$$

Use o `Python`para construir os seguintes vetores de erro:

$$e_j = \frac{|x_j - y_j|}{|x_j|} \quad f_j = \frac{|x_j - z_j|}{x_j} \tag{4.177}$$

Calcule as normas 1, 2 e ∞ de e e f

4.7 Métodos iterativos para sistemas lineares

Na seção anterior, tratamos de métodos diretos para a resolução de sistemas lineares. Em um **método direto** (por exemplo, solução via fatoração LU) obtemos uma aproximação da solução depois de realizarmos um número finito de operações (só teremos a solução ao final do processo).

Veremos nessa seção dois **métodos iterativos** básicos para obter uma aproximação para a solução de um sistema linear. Geralmente em um método iterativo, iniciamos com uma aproximação para a solução (que pode ser ruim) e vamos melhorando essa aproximação através de sucessivas iterações.

4.7.1 Método de Jacobi

O método de Jacobi pode ser obtido a partir do sistema linear

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = y_1 \tag{4.178}$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = y_2 \tag{4.179}$$
$$\vdots \tag{4.180}$$
$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = y_n \tag{4.181}$$

Isolando o elemento x_1 da primeira equação temos

$$x_1^{(k+1)} = \frac{y_1 - \left(a_{12}x_2^{(k)} + \cdots + a_{1n}x_n^{(k)}\right)}{a_{11}} \tag{4.182}$$

Note que utilizaremos os elementos $x_i^{(k)}$ da iteração k (a direita da equação) para estimar o elemento x_1 da próxima iteração.

Da mesma forma, isolando o elemento x_i de cada equação i, para todo $i = 2,...,n$ podemos construir a iteração

$$\begin{aligned}
x_1^{(k+1)} &= \frac{y_1 - \left(a_{12}x_2^{(k)} + \cdots + a_{1n}x_n^{(k)}\right)}{a_{11}} && (4.183)\\
x_2^{(k+1)} &= \frac{y_2 - \left(a_{21}x_1^{(k)} + a_{23}x_3^{(k)} + \cdots + a_{2n}x_n^{(k)}\right)}{a_{22}} && (4.184)\\
&\vdots && (4.185)\\
x_n^{(k+1)} &= \frac{y_2 - \left(a_{n1}x_1^{(k)} + \cdots + a_{n,n-2}x_{n-2}^{(k)} + a_{n,n-1}x_{n-1}^{(k)}\right)}{a_{nn}} && (4.186)
\end{aligned}$$

Em notação mais compacta, o método de Jacobi consiste na iteração

$$\begin{aligned}
x^{(1)} &= \text{aproximação inicial} && (4.187)\\
x_i^{(k+1)} &= \left(y_i - \sum_{\substack{j=1\\ j\neq i}}^{n} a_{ij}x_j^{(k)}\right)/a_{ii} && (4.188)
\end{aligned}$$

Exemplo 4.7.1. Resolva o sistema

$$\begin{aligned}
10x + y &= 23 && (4.189)\\
x + 8y &= 26 && (4.190)
\end{aligned}$$

usando o método de Jacobi iniciando com $x^{(1)} = y^{(1)} = 0$.

$$x^{(k+1)} = \frac{23 - y^{(k)}}{10} \tag{4.191}$$

$$y^{(k+1)} = \frac{26 - x^{(k)}}{8} \tag{4.192}$$

$$x^{(2)} = \frac{23 - y^{(1)}}{10} = 2{,}3 \tag{4.193}$$

$$y^{(2)} = \frac{26 - x^{(1)}}{8} = 3{,}25 \tag{4.194}$$

$$x^{(3)} = \frac{23 - y^{(2)}}{10} = 1{,}975 \tag{4.195}$$

$$y^{(3)} = \frac{26 - x^{(2)}}{8} = 2{,}9625 \tag{4.196}$$

Exemplo 4.7.2. Considere o seguinte sistema

$$\begin{aligned} -3x_1 + x_2 + x_3 &= 2 \\ 2x_1 + 5x_2 + x_3 &= 5 \\ 2x_1 + 3x_2 + 7x_3 &= -17 \end{aligned} \tag{4.197}$$

Usando o método de Jacobi com aproximação inicial $x^{(1)} = (1, 1, -1)$, obtemos os seguintes resultados:

k	$x^{(k)}$	$\|x^{(k)} - x^{(k-1)}\|_\infty$
1	$(1, 1, 1)$	-x-
2	$(-0{,}67,\ 0{,}80,\ -3{,}14)$	2,1
3	$(-1{,}45,\ 1{,}90,\ -2{,}58)$	1,1
4	$(-0{,}90,\ 2{,}10,\ -2{,}83)$	5,5E−1
5	$(-0{,}91,\ 1{,}92,\ -3{,}07)$	2,4E−1
⋮	⋮	⋮
10	$(-1{,}00,\ 2{,}00,\ -3{,}00)$	6,0E−3

Verifique a resposta.

Aqui, cabe um código Python explicativo. Escreva você mesmo o código.

Veja como participar da escrita do livro em:

https://github.com/livroscolaborativos/CalculoNumerico

Código Python: Método de Jacobi

```
from __future__ import division
import numpy as np
from numpy import linalg

def jacobi(A,b,x0,tol,N):
    #preliminares
    A = A.astype('double')
    b = b.astype('double')
    x0 = x0.astype('double')

    n=np.shape(A)[0]
    x = np.zeros(n)
    it = 0
    #iteracoes
    while (it < N):
        it = it+1
        #iteracao de Jacobi
        for i in np.arange(n):
            x[i] = b[i]
            for j in np.concatenate((np.arange(0,i),np.arange(i+1,n))):
                x[i] -= A[i,j]*x0[j]
            x[i] /= A[i,i]
        #tolerancia
        if (np.linalg.norm(x-x0,np.inf) < tol):
            return x
        #prepara nova iteracao
        x0 = np.copy(x)
    raise NameError('num. max. de iteracoes excedido.')
```

4.7.2 Método de Gauss-Seidel

Assim, como no método de Jacobi, no método de Gauss-Seidel também isolamos o elemento x_i da equação i. Porém perceba que a equação para $x_2^{(k+1)}$ depende de $x_1^{(k)}$ na iteração k. Intuitivamente podemos pensar em usar $x_1^{(k+1)}$ que acabou de ser calculado e temos

$$x_2^{(k+1)} = \frac{y_2 - \left(a_{21}x_1^{(k+1)} + a_{23}x_3^{(k)} + \cdots + a_{2n}x_n^{(k)}\right)}{a_{22}} \tag{4.198}$$

Aplicando esse raciocínio, podemos construir o método de Gauss-Seidel como

$$x_1^{(k+1)} = \frac{y_1 - \left(a_{12}x_2^{(k)} + \cdots + a_{1n}x_n^{(k)}\right)}{a_{11}} \tag{4.199}$$

$$x_2^{(k+1)} = \frac{y_2 - \left(a_{21}x_1^{(k+1)} + a_{23}x_3^{(k)} + \cdots + a_{2n}x_n^{(k)}\right)}{a_{22}} \tag{4.200}$$

$$\vdots \tag{4.201}$$

$$x_n^{(k+1)} = \frac{y_2 - \left(a_{n1}x_1^{(k+1)} + \cdots + a_{n(n-1)}x_{n-1}^{(k+1)}\right)}{a_{nn}} \tag{4.202}$$

Em notação mais compacta, o método de Gauss-Seidel consiste na iteração:

$$x^{(1)} = \text{aproximação inicial} \tag{4.203}$$

$$x_i^{(k+1)} = \frac{y_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}}{a_{ii}} \tag{4.204}$$

Exemplo 4.7.3. Resolva o sistema

$$10x + y = 23 \tag{4.205}$$

$$x + 8y = 26 \tag{4.206}$$

usando o método de Gauss-Seidel com condições iniciais $x^{(1)} = y^{(1)} = 0$.

$$x^{(k+1)} = \frac{23 - y^{(k)}}{10} \tag{4.207}$$

$$y^{(k+1)} = \frac{26 - x^{(k+1)}}{8} \tag{4.208}$$

$$x^{(2)} = \frac{23 - y^{(1)}}{10} = 2{,}3 \tag{4.209}$$

$$y^{(2)} = \frac{26 - x^{(2)}}{8} = 2{,}9625 \tag{4.210}$$

$$x^{(3)} = \frac{23 - y^{(2)}}{10} = 2{,}00375 \tag{4.211}$$

$$y^{(3)} = \frac{26 - x^{(3)}}{8} = 2{,}9995312 \tag{4.212}$$

Código Python: Método de Gauss-Seidel

```
from __future__ import division
import numpy as np
```

```
from numpy import linalg

def gauss_seidel(A,b,x0,tol,N):
    #preliminares
    A = A.astype('double')
    b = b.astype('double')
    x0 = x0.astype('double')

    n=np.shape(A)[0]
    x = np.copy(x0)
    it = 0
    #iteracoes
    while (it < N):
        it = it+1
        #iteracao de Jacobi
        for i in np.arange(n):
            x[i] = b[i]
            for j in np.concatenate((np.arange(0,i),np.arange(i+1,n))):
                x[i] -= A[i,j]*x[j]
            x[i] /= A[i,i]
            print(x[i],A[i,i])
        #tolerancia
        if (np.linalg.norm(x-x0,np.inf) < tol):
            return x
        #prepara nova iteracao
        x0 = np.copy(x)
    raise NameError('num. max. de iteracoes excedido.')
```

4.7.3 Análise de convergência

Nesta seção, analisamos a convergência de métodos iterativos para solução de sistema lineares. Para tanto, consideramos um sistema linear $Ax = b$, onde $A = [a_{i,j}]_{i,j=1}^{n,n}$ é a matriz (real) dos coeficientes, $b = (a_j)_{j=1}^{n}$ é um vetor dos termos constantes e $x = (x_j)_{j=1}^{n}$ é o vetor incógnita. No decorrer, assumimos que A é uma matriz não singular.

Geralmente, métodos iterativos são construídos como uma iteração de ponto fixo. No caso de um sistema linear, reescreve-se a equação matricial em um problema de ponto fixo equivalente, isto é:

$$Ax = b \Leftrightarrow x = Tx + c, \tag{4.213}$$

onde $T = [t_{i,j}]_{i,j=1}^{n,n}$ é chamada de **matriz da iteração** e $c = (c_j)_{j=1}^{n}$ de **vetor**

da iteração. Construídos a matriz T e o vetor c, o método iterativo consiste em computar a iteração:

$$x^{(k+1)} = Tx^{(k)} + c, \quad n \geq 1, \tag{4.214}$$

onde $x^{(1)}$ é uma aproximação inicial dada.

A fim de construirmos as matrizes e os vetores de iteração do método de Jacobi e de Gauss-Seidel, decompomos a matriz A da seguinte forma:

$$A = L + D + U, \tag{4.215}$$

onde D é a matriz diagonal $D = \text{diag}\,(a_{11}, a_{22}, \ldots, a_{nn})$, isto é:

$$D := \begin{bmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ 0 & a_{22} & 0 & \cdots & 0 \\ 0 & 0 & a_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn} \end{bmatrix}, \tag{4.216}$$

e, respectivamente, L e U são as seguintes matrizes triangular inferior e superior:

$$L := \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ a_{21} & 0 & 0 & \cdots & 0 \\ a_{31} & a_{32} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & 0 \end{bmatrix}, \quad U := \begin{bmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & 0 & a_{23} & \cdots & a_{2n} \\ 0 & 0 & 0 & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}. \tag{4.217}$$

Exemplo 4.7.4. Considere o seguinte sistema linear:

$$3x_1 + x_2 - x_3 = 2 \tag{4.218}$$

$$-x_1 - 4x_2 + x_3 = -10 \tag{4.219}$$

$$x_1 - 2x_2 - 5x_3 = 10 \tag{4.220}$$

Escreva o sistema na sua forma matricial $Ax = b$ identificando a matriz dos coeficientes A, o vetor incógnita x e o vetor dos termos constantes b. Em seguida, faça a decomposição $A = L + D + U$.

Solução. A forma matricial deste sistema é $Ax = b$, onde:

$$A = \begin{bmatrix} 3 & 1 & -1 \\ -1 & -4 & 1 \\ 1 & -2 & -5 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} 2 \\ -10 \\ 10 \end{bmatrix}. \tag{4.221}$$

A decomposição da matriz A nas matrizes L triangular inferior, D diagonal e U triangular superior é:

$$\underbrace{\begin{bmatrix} 3 & 1 & -1 \\ -1 & -4 & 1 \\ 1 & -2 & -5 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 1 & -2 & 0 \end{bmatrix}}_{L} + \underbrace{\begin{bmatrix} 3 & 0 & 0 \\ 0 & -4 & 0 \\ 0 & 0 & -5 \end{bmatrix}}_{D} + \underbrace{\begin{bmatrix} 0 & 1 & -1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}}_{U}. \tag{4.222}$$

Em `Python`, podemos construir as matrizes L, D e U, da seguinte forma:

```
>>> A = np.array([[3,1,-1],
...               [-1,-4,1],
...               [1,-2,5]],
...              dtype='double')
>>> D = np.diag(np.diag(A))
>>> L = np.tril(A)-D
>>> U=np.triu(A)-D
```

◊

Iteração de Jacobi

Vamos, agora, usar a decomposição discutida acima para construir a matriz de iteração T_J e o vetor de iteração c_J associado ao método de Jacobi. Neste caso, temos:

$$\begin{aligned} Ax = b &\Leftrightarrow (L + D + U)x = b && (4.223)\\ &\Leftrightarrow Dx = -(L + U)x + b && (4.224)\\ &\Leftrightarrow x = \underbrace{-D^{-1}(L + U)}_{=:T_J} x + \underbrace{D^{-1}b}_{=:c_J}. && (4.225) \end{aligned}$$

Ou seja, a iteração do método de Jacobi escrita na forma matricial é:

$$x^{(k+1)} = T_J x^{(k)} + c_J, \quad k \geq 1, \tag{4.226}$$

com $x^{(1)}$ uma aproximação inicial dada, sendo $T_J := -D^{-1}(L + U)$ a matriz de iteração e $c_J = D^{-1}b$ o vetor da iteração.

Exemplo 4.7.5. Construa a matriz de iteração T_J e o vetor de iteração c_J do método de Jacobi para o sistema dado no Exemplo 4.7.4.

Solução. A matriz de iteração é dada por:

$$T_J := -D^{-1}(L+U) = -\underbrace{\begin{bmatrix} \frac{1}{3} & 0 & 0 \\ 0 & -\frac{1}{4} & 0 \\ 0 & 0 & -\frac{1}{5} \end{bmatrix}}_{D^{-1}} \underbrace{\begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & 2 & 0 \end{bmatrix}}_{(L+U)} = \begin{bmatrix} 0 & -\frac{1}{3} & \frac{1}{3} \\ -\frac{1}{4} & 0 & \frac{1}{4} \\ \frac{1}{5} & \frac{2}{5} & 0 \end{bmatrix}. \quad (4.227)$$

O vetor da iteração de Jacobi é:

$$c_J := D^{-1}b = \underbrace{\begin{bmatrix} \frac{1}{3} & 0 & 0 \\ 0 & -\frac{1}{4} & 0 \\ 0 & 0 & -\frac{1}{5} \end{bmatrix}}_{D^{-1}} \underbrace{\begin{bmatrix} 2 \\ -10 \\ 10 \end{bmatrix}}_{b} = \begin{bmatrix} \frac{2}{3} \\ \frac{5}{2} \\ -2 \end{bmatrix}. \quad (4.228)$$

Em `python`, podemos computar T_J e c_J da seguinte forma:

```
>>> TJ = -np.linalg.inv(D).dot(L+U);
>>> cJ = np.linalg.inv(D).dot(b);
```

◊

Iteração de Gauss-Seidel

A forma matricial da iteração do método de Gauss-Seidel também pode ser construída com base na decomposição $A = L + D + U$. Para tando, fazemos:

$$\begin{aligned} Ax = b &\Leftrightarrow (L+D+U)x = b && (4.229)\\ &\Leftrightarrow (L+D)x = -Ux + b && (4.230)\\ &\Leftrightarrow x = \underbrace{-(L+D)^{-1}U}_{=:T_G} x + \underbrace{(L+D)^{-1}b}_{=:c_G} && (4.231) \end{aligned}$$

Ou seja, a iteração do método de Gauss-Seidel escrita na forma matricial é:

$$x^{(k+1)} = T_G x^{(k)} + c_G, \quad k \geq 1, \quad (4.232)$$

com $x^{(1)}$ uma aproximação inicial dada, sendo $T_G := -(L+D)^{-1}U$ a matriz de iteração e $c_J = (L+D)^{-1}b$ o vetor da iteração.

Exemplo 4.7.6. Construa a matriz de iteração T_G e o vetor de iteração c_G do método de Gauss-Seidel para o sistema dado no Exemplo 4.7.4.

Solução. A matriz de iteração é dada por:

$$T_G = -(L+D)^{-1}U = -\underbrace{\begin{bmatrix} 3 & 0 & 0 \\ -1 & -4 & 0 \\ 1 & -2 & -5 \end{bmatrix}^{-1}}_{(L+D)^{-1}} \underbrace{\begin{bmatrix} 0 & 1 & -1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}}_{U} = \begin{bmatrix} 0 & -\frac{1}{3} & \frac{1}{3} \\ 0 & \frac{1}{12} & \frac{1}{6} \\ 0 & -\frac{1}{10} & 0 \end{bmatrix}. \quad (4.233)$$

O vetor da iteração de Gauss-Seidel é:

$$c_G := (L+D)^{-1}b = \underbrace{\begin{bmatrix} 3 & 0 & 0 \\ -1 & -4 & 0 \\ 1 & -2 & -5 \end{bmatrix}^{-1}}_{(L+D)^{-1}} \underbrace{\begin{bmatrix} 2 \\ -10 \\ 10 \end{bmatrix}}_{b} = \begin{bmatrix} \frac{2}{3} \\ \frac{7}{3} \\ -\frac{28}{10} \end{bmatrix}. \quad (4.234)$$

Em `Python`, podemos computar T_G e c_G da seguinte forma:

```
-->TG = -np.linalg.inv(L+D).dot(U);
-->cG = np.linalg.inv(L+D).dot(b);
```

◊

Condições de convergência

Aqui, vamos discutir condições necessárias e suficientes para a convergência de métodos iterativos. Isto é, dado um sistema $Ax = b$ e uma iteração:

$$x^{(k+1)} = Tx^{(k)} + c, \quad k \geq 1, \quad (4.235)$$

$x^{(1)}$ dado, estabelecemos condições nas quais $x^{(k)} \to x^*$, onde x^* é a solução do sistema dado, isto é, $x^* = Tx^* + c$ ou, equivalentemente, $Ax^* = b$.

Lema 4.7.1. *Seja T uma matriz real $n \times n$. O limite $\lim_{k\to\infty} \left\|T^k\right\|_p = 0$, $1 \leq p \leq \infty$, se, e somente se, $\rho(T) < 1$.*

Demonstração. Aqui, fazemos apenas um esboço da demonstração. Para mais detalhes, veja [8], Teorema 4, pág. 14.

Primeiramente, suponhamos que $\|T\|_p < 1$, $1 \leq p \leq \infty$. Como (veja [8], lema 2, pág. 12):

$$\rho(T) \leq \|T\|_p, \quad (4.236)$$

temos $\rho(T) < 1$, o que mostra a implicação.

Agora, suponhamos que $\rho(T) < 1$ e seja $0 < \epsilon < 1 - \rho(T)$. Então, existe $1 \leq p \leq \infty$ tal que (veja [8], Teorema 3, página 12):

$$\|T\|_p \leq \rho(T) + \epsilon < 1. \tag{4.237}$$

Assim, temos:

$$\lim_{k\to\infty} \|T^k\|_p \leq \lim_{k\to\infty} \|T\|_p^m = 0. \tag{4.238}$$

Da equivalência entre as normas segue a recíproca. □

Observação 4.7.1. Observamos que:

$$\lim_{k\to\infty} \|T^k\|_p = 0, \quad ,1 \leq p \leq \infty, \Leftrightarrow \lim_{k\to\infty} t_{ij}^k = 0, \quad 1 \leq i,j \leq n. \tag{4.239}$$

Lema 4.7.2. *Se $\rho(T) < 1$, então existe $(I-T)^{-1}$ e:*

$$(I-T)^{-1} = \sum_{k=0}^{\infty} T^k. \tag{4.240}$$

Demonstração. Primeiramente, provamos a existência de $(I-T)^{-1}$. Seja λ um autovalor de T e x um autovetor associado, isto é, $Tx = \lambda x$. Então, $(I-T)x = (1-\lambda)x$. Além disso, temos $|\lambda| < \rho(T) < 1$, logo $(1-\lambda) \neq 0$, o que garante que $(I-T)$ é não singular. Agora, mostramos que $(I-T)^{-1}$ admite a expansão acima. Do Lema 4.7.1 e da Observação 4.7.1 temos:

$$(I-T)\sum_{k=0}^{\infty} T^k = \lim_{m\to\infty}(I-T)\sum_{k=0}^{m} T^k = \lim_{m\to\infty}(I-T^{m+1}) = I, \tag{4.241}$$

o que mostra que $(I-T)^{-1} = \sum_{k=0}^{\infty} T^k$. □

Teorema 4.7.1. *A sequência recursiva $\{x^{(k)}\}_{k\in\mathbb{N}}$ dada por:*

$$x^{(k+1)} = Tx^{(k)} + c \tag{4.242}$$

converge para solução de $x = Tx + c$ para qualquer escolha de $x^{(1)}$ se, e somente se, $\rho(T) < 1$.

Demonstração. Primeiramente, assumimos que $\rho(T) < 1$. Observamos que:

$$\begin{aligned} x^{(k+1)} &= Tx^{(k)} + c = T(Tx^{(k-1)} + c) + c && (4.243)\\ &= T^2x^{(k-1)} + (I+T)c && (4.244)\\ &\vdots && (4.245)\\ &= T^{(k)}x^{(1)} + \left(\textstyle\sum_{k=0}^{k-1} T^k\right)c. && (4.246) \end{aligned}$$

Daí, do Lema 4.7.1 e do Lema 4.7.2 temos:

$$\lim_{k\to\infty} x^{(k)} = (I-T)^{(-1)}c. \tag{4.247}$$

Ora, se x^* é a solução de $x = Tx + c$, então $(I-T)x^* = c$, isto é, $x^* = (I-T)^{-1}c$. Logo, temos demonstrado que $x^{(k)}$ converge para a solução de $x = Tx + c$, para qualquer escolha de $x^{(1)}$.

Agora, suponhamos que $x^{(k)}$ converge para x^* solução de $x = Tx + c$, para qualquer escolha de $x^{(1)}$. Seja, então, y um vetor arbitrário e $x^{(1)} = x^* - y$. Observamos que:

$$\begin{aligned} x^* - x^{(k+1)} &= (Tx^* + c) - (Tx^{(k)} + c) && (4.248)\\ &= T(x^* - x^{(k)}) && (4.249)\\ &\vdots && (4.250)\\ &= T^{(k)}(x^* - x^{(1)}) = T^{(k)}y. && (4.251) \end{aligned}$$

Logo, para qualquer $1 \le p \le \infty$, temos, :

$$0 = \lim_{k\to\infty} x^* - x^{(k+1)} = \lim_{k\to\infty} T^{(k)}y. \tag{4.252}$$

Como y é arbitrário, da Observação 4.7.1 temos $\lim_{k\to\infty} \|T^{(k)}\|_p = 0$, $1 \le p \le \infty$. Então, o Lema 4.7.1 garante que $\rho(T) < 1$. □

Observação 4.7.2. Pode-se mostrar que tais métodos iterativos tem taxa de convergência super linear com:

$$\|x^{(k+1)} - x^*\| \approx \rho(T)^k \|x^{(1)} - x^*\|. \tag{4.253}$$

Para mais detalhes, veja [8], pág. 61-64.

Exemplo 4.7.7. Mostre que, para qualquer escolha da aproximação inicial, ambos os métodos de Jacobi e Gauss-Seidel são convergentes quando aplicados ao sistema linear dado no Exemplo 4.7.4.

Solução. Do Teorema 4.7.1, vemos que é necessário e suficiente que $\rho(T_J) < 1$ e $\rho(T_G) < 1$. Computando estes raios espectrais, obtemos $\rho(T_J) \approx 0{,}32$ e $\rho(T_G) \approx 0{,}13$. Isto mostra que ambos os métodos serão convergentes. ◊

Condição suficiente

Uma condição suficiente porém não necessária para que os métodos de Gauss-Seidel e Jacobi convirjam é a que a matriz seja **estritamente diagonal dominante**.

Definição 4.7.1. *Uma matriz A é **estritamente diagonal dominante** quando:*

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|\,, i = 1,...,n \tag{4.254}$$

Definição 4.7.2. *Uma matriz A é **diagonal dominante** quando*

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|\,, i = 1,...,n \tag{4.255}$$

e para ao menos um i, a_{ii} é estritamente maior que a soma dos elementos fora da diagonal.

Teorema 4.7.2. *Se a matriz A for diagonal dominante[7], então os métodos de Jacobi e Gauss-Seidel serão convergentes independente da escolha inicial $x^{(1)}$.*

Se conhecermos a solução exata x do problema, podemos calcular o erro relativo em cada iteração como:

$$\frac{\|x - x^{(k)}\|}{\|x\|}. \tag{4.256}$$

Em geral não temos x, entretanto podemos estimar o vetor **resíduo** $r^{(k)} = b - A\tilde{x^{(k)}}$. Note que quando o erro tende a zero, o resíduo também tende a zero.

Teorema 4.7.3. *O erro relativo e o resíduo estão relacionados como (veja [3])*

$$\frac{\|x - x^{(k)}\|}{\|x\|} \leq \kappa(A)\frac{\|r\|}{\|b\|} \tag{4.257}$$

onde $k(A)$ é o número de condicionamento.

Exemplo 4.7.8. Ambos os métodos de Jacobi e Gauss-Seidel são convergentes para o sistema dado no Exemplo 4.7.4, pois a matriz dos coeficientes deste é uma matriz estritamente diagonal dominante.

Exercícios

E 4.7.1. Considere o problema de 5 incógnitas e cinco equações dado por

[7] Ou se for estritamente diagonal dominante e, consequentemente, diagonal dominante.

$$
\begin{aligned}
x_1 - x_2 &= 1 && (4.258)\\
-x_1 + 2x_2 - x_3 &= 1 && (4.259)\\
-x_2 + (2+\varepsilon)x_3 - x_4 &= 1 && (4.260)\\
-x_3 + 2x_4 - x_5 &= 1 && (4.261)\\
x_4 - x_5 &= 1 && (4.262)
\end{aligned}
$$

a) Escreva na forma $Ax = b$ e resolva usando eliminação gaussiana para $\varepsilon = 10^{-3}$ no `Python`.

b) Obtenha o vetor incógnita x com $\varepsilon = 10^{-3}$ usando Jacobi com tolerância 10^{-2}. Compare o resultado com o resultado obtido no item d.

c) Obtenha o vetor incógnita x com $\varepsilon = 10^{-3}$ usando Gauss-Seidel com tolerância 10^{-2}. Compare o resultado com o resultado obtido no item d.

d) Discuta com base na relação esperada entre tolerância e exatidão conforme estudado na primeira área para problemas de uma variável.

E 4.7.2. Resolva o seguinte sistema pelo método de Jacobi e Gauss-Seidel:

$$
\begin{cases}
5x_1 + x_2 + x_3 &= 50\\
-x_1 + 3x_2 - x_3 &= 10\\
x_1 + 2x_2 + 10x_3 &= -30
\end{cases} \qquad (4.263)
$$

Use como critério de paragem tolerância inferior a 10^{-3} e inicialize com $x^0 = y^0 = z^0 = 0$.

E 4.7.3. Refaça o Exercício **??** construindo um algoritmo que implemente os métodos de Jacobi e Gauss-Seidel.

E 4.7.4. Considere o seguinte sistema de equações lineares:

$$
\begin{aligned}
x_1 - x_2 &= 0\\
-x_{j-1} + 5x_j - x_{j+1} &= \cos(j/10),\ \ 2 \leq j \leq 10\\
x_{11} &= x_{10}/2
\end{aligned} \qquad (4.264)
$$

Construa a iteração para encontrar a solução deste problema pelos métodos de Gauss-Seidel e Jacobi. Usando esses métodos, encontre uma solução aproximada com erro absoluto inferior a 10^{-5}. Veja também Exercício 4.5.2

E 4.7.5. Faça uma permutação de linhas no sistema abaixo e resolva pelos métodos de Jacobi e Gauss-Seidel:

$$x_1 + 10x_2 + 3x_3 = 27 \tag{4.265}$$
$$4x_1 + x_3 = 6 \tag{4.266}$$
$$2x_1 + x_2 + 4x_3 = 12 \tag{4.267}$$

4.8 Cálculo de autovalores e autovetores

Considere o problema de autovalores $Av = \lambda v$, onde A é uma matriz diagonalizável, isto é, existe uma matriz diagonal D e uma matriz ortogonal U tal que $A = UDU^{-1}$.

4.8.1 Método da potência

O método da potência para cálculo do maior autovalor (em módulo) consiste na iteração

$$\begin{cases} x^{(1)} &= \text{aprox. inicial do autovetor}, \\ \lambda^{(1)} &= {x^{(1)}}^T A x^{(1)}, \\ x^{(k+1)} &= \frac{Ax^{(k)}}{\|Ax^{(k)}\|_2}, \\ \lambda^{(k+1)} &= {x^{(k+1)}}^T A x^{(k+1)}, \end{cases} \quad , \quad k \geq 1. \tag{4.268}$$

Observação 4.8.1. Observe que na iteração do método da potência (4.268), temos $Ax^{(k)} = A^{(k)}x^{(1)}$.

Exemplo 4.8.1. A seguinte matriz

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 3 & 0 \\ 3 & 4 & 2 \end{bmatrix} \tag{4.269}$$

tem $\lambda = 3$ como maior autovalor (por quê?). Tomando $x^{(1)} = (1, 1, 1)$, a iteração do método da potência nos fornece os seguintes resultados:

k	$x^{(k)}$	$\lambda^{(k)}$
1	$(1, 1, 1)$	17
2	(0,08, 0,41, 0,91)	4,10
3	(0,02, 0,34, 0,94)	3,50
4	(0,01, 0,30, 0,95)	3,28
5	(0,00, 0,28, 0,96)	3,16
$\vdots$	$\vdots$	$\vdots$
14	(0,00, 0,24, 0,97)	3,00

Aqui, cabe um código Python explicativo. Escreva você mesmo o código.

Veja como participar da escrita do livro em:

https://github.com/livroscolaborativos/CalculoNumerico

Para entendermos melhor o comportamento assintótico da sequência $\{x^{(n)}\}_{n\geq 1}$, primeiro consideramos o caso particular onde A é uma matriz diagonal, isto é,

$$A = \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_3 & \cdots & 0 \\ & \vdots & & \ddots & \\ 0 & 0 & 0 & \cdots & \lambda_n \end{bmatrix}. \tag{4.270}$$

Suponha que um dos autovalores seja estritamente maior que os demais, isto é, $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$. Dado $x^{(1)} = (\xi_1, \xi_2, \xi_3, \ldots, \xi_n)$, então

$$A^k x^{(1)} = A^k \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \vdots \\ \xi_n \end{bmatrix} = \begin{bmatrix} \lambda_1^k \xi_1 \\ \lambda_2^k \xi_2 \\ \lambda_3^k \xi_3 \\ \vdots \\ \lambda_n^k \xi_n \end{bmatrix} = \lambda_1^k \xi_1 \begin{bmatrix} 1 \\ \frac{\xi_2}{\xi_1}\left(\frac{\lambda_2}{\lambda_1}\right)^k \\ \frac{\xi_3}{\xi_1}\left(\frac{\lambda_3}{\lambda_1}\right)^k \\ \vdots \\ \frac{\xi_n}{\xi_1}\left(\frac{\lambda_n}{\lambda_1}\right)^k \end{bmatrix}, \tag{4.271}$$

desde que $\xi_1 \neq 0$. Como $\frac{\lambda_n}{\lambda_1} \leq \frac{\lambda_{n-1}}{\lambda_1} \leq \cdots \frac{\lambda_3}{\lambda_1} \leq \frac{\lambda_2}{\lambda_1} < 1$, então $\left(\frac{\lambda_j}{\lambda_1}\right)^k$ tende a 0 para cada j, $2 \leq j \leq n$. Devido à normalização realizada em cada passo da sequência,

$$x^{(k+1)} = \frac{A^k x^{(1)}}{\|A^k x^{(1)}\|_2} \tag{4.272}$$

converge para $\pm e_1$, $e_1 = (1, 0, 0, \ldots, 0)$. Também, a sequência

$$\lambda^{(k)} = {x^{(k)}}^T A x^{(k)} \tag{4.273}$$

converge para λ_1, pois

$$\lim_{k\to\infty} \lambda^{(k)} = (\pm e_1)^T A(\pm e_1) = \lambda_1 e_1^T e_1 = \lambda_1. \tag{4.274}$$

Considere, agora, o caso onde A é diagonalizável, ou seja, $A = UDU^{-1}$ com U uma matriz ortogonal contendo os autovetores em cada coluna e D uma matriz diagonal contendo os autovalores:

$$D = \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_3 & \cdots & 0 \\ & \vdots & & \ddots & \\ 0 & 0 & 0 & \cdots & \lambda_n \end{bmatrix}. \tag{4.275}$$

Considere a mesma hipótese sobre os autovalores: $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$. Então

$$A^k = (UDU^{-1})(UDU^{-1})(UDU^{-1})\cdots(UDU^{-1}) = UD^kU^{-1} \tag{4.276}$$

visto que $UU^{-1} = I$. Suponha que o chute inicial $x^{(1)}$ pode ser escrito da forma

$$x^{(1)} = UU^{-1}x^{(1)} = U[\xi_1 \;\; \xi_2 \;\; \xi_3 \;\; \cdots \;\; \xi_n]^T \tag{4.277}$$

com $\xi_1 \neq 0$. Então

$$A^k x^{(1)} = (UD^kU^{-1})U \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \vdots \\ \xi_n \end{bmatrix} = U \begin{bmatrix} \lambda_1^k\xi_1 \\ \lambda_2^k\xi_2 \\ \lambda_3^k\xi_3 \\ \vdots \\ \lambda_n^k\xi_n \end{bmatrix} = \lambda_1^k\xi_1 U \begin{bmatrix} 1 \\ \frac{\xi_2}{\xi_1}\left(\frac{\lambda_2}{\lambda_1}\right)^k \\ \frac{\xi_3}{\xi_1}\left(\frac{\lambda_3}{\lambda_1}\right)^k \\ \vdots \\ \frac{\xi_n}{\xi_1}\left(\frac{\lambda_n}{\lambda_1}\right)^k \end{bmatrix}. \tag{4.278}$$

Como na discussão anterior, o último vetor converge para $\pm e_1$ e

$$x^{(k)} = \frac{A^k x^{(1)}}{\|A^k x^{(1)}\|_2} \tag{4.279}$$

converge para $v_1 = \pm U e_1$ que é um múltiplo do autovetor associado a λ_1. Também, a sequência

$$\lambda^{(k)} = {x^{(k)}}^T A x^{(k)} \tag{4.280}$$

converge para o autovalor dominante λ_1:

$$\lim_{k\to\infty} \lambda^{(k)} = v_1^T A v_1 = \lambda_1 v_1^T v_1 = \lambda_1. \tag{4.281}$$

Observação 4.8.2. O método da potência tem duas restrições:

i) A aproximação inicial $x^{(1)}$ não pode ser ortogonal ao autovetor associado ao autovalor dominante.

ii) Um autovalor deve ter o módulo estritamente maior que os demais. Essa restrição impede o funcionamento do método no caso em que o autovalor dominante é complexo, pois eles aparecem em pares conjugados, possuindo o mesmo módulo.

Outra análise para a convergência do método

Aqui, vamos apresentar uma análise alternativa para a convergência do método da potência: se $A \in \mathbb{R}^{n,n}$ é diagonalizável, então existe um conjunto $\{v_j\}_{j=1}^n$ de autovetores de A tais que qualquer elemento $x \in \mathbb{R}^n$ pode ser escrito como uma combinação linear dos v_j. Sejam $\{\lambda_j\}_{j=1}^n$ o conjunto de autovalores associados aos autovetores tal que um deles seja dominante, ou seja, $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots |\lambda_n| > 0$. Como os autovetores são linearmente independentes, todo vetor $x \in \mathbb{R}^n$, $x = (x_1,x_2,...,x_n)$, pode ser escrito com combinação linear dos autovetores da seguinte forma:

$$x = \sum_{j=1}^{n} \beta_j v_j. \tag{4.282}$$

Observamos que se x está na forma (4.282), então $A^k x$ pode ser escrito como

$$A^k x = \sum_{j=1}^{n} \beta_j A^k v_j = \sum_{j=1}^{n} \beta_j \lambda_j^k v_j = \beta_1 \lambda_1^k \left(v_1 + \sum_{j=2}^{n} \frac{\beta_j}{\beta_1} \left(\frac{\lambda_j}{\lambda_1} \right)^k v_j \right). \tag{4.283}$$

Como $\left|\frac{\lambda_j}{\lambda_1}\right| < 1$ para todo $j \geq 2$, temos

$$\sum_{j=2}^{n} \frac{\beta_j}{\beta_1} \left(\frac{\lambda_j}{\lambda_1} \right)^k v_j \to 0. \tag{4.284}$$

Assim,

$$\frac{A^k x}{\|A^k x\|_2} = \frac{\beta_1 \lambda_1^k}{\|A^k x\|} \left(v_1 + O\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \right). \tag{4.285}$$

Como a norma de $\frac{A^k x}{\|A^k x\|}$ é igual a um, temos

$$\left\| \frac{\beta_1 \lambda_1^k}{\|A^k x\|} v_1 \right\| \to 1 \tag{4.286}$$

e, portanto,

$$\left| \frac{\beta_1 \lambda_1^k}{\|A^k x\|} \right| \to \frac{1}{\|v_1\|}. \tag{4.287}$$

Ou seja, se definimos $\alpha^{(k)} = \frac{\beta_1 \lambda_1^k}{\|A^k x\|}$, então

$$|\alpha^{(k)}| \to 1. \tag{4.288}$$

Retornando a (4.285), temos:

$$\frac{A^k x}{\|A^k x\|} - \alpha^{(k)} v_1 \to 0. \tag{4.289}$$

Observe que um múltiplo de autovetor também é um autovetor e, portanto,

$$x^{(k)} = \frac{A^k x^{(1)}}{\|A^k x^{(1)}\|}. \tag{4.290}$$

é um esquema que oscila entre os autovetores ou converge para o autovetor v_1.

4.8.2 Método da iteração inversa

Nesta seção, vamos estudar a sequência

$$x_0, \frac{(A-\sigma I)^{-1} x_0}{\|(A-\sigma I)^{-1} x_0\|_2}, \frac{(A-\sigma I)^{-2} x_0}{\|(A-\sigma I)^{-2} x_0\|_2}, \frac{(A-\sigma I)^{-3} x_0}{\|(A-\sigma I)^{-3} x_0\|_2}, \cdots \tag{4.291}$$

para valores iniciais x_0 e σ. Para o caso onde A é diagonalizável podemos escrever $A = UDU^{-1}$ com D diagonal,

$$D = \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_3 & \cdots & 0 \\ & \vdots & & \ddots & \\ 0 & 0 & 0 & \cdots & \lambda_n \end{bmatrix}. \tag{4.292}$$

Assim, $A-\sigma I = U(D-\sigma I)U^{-1}$ e, portanto, $(A-\sigma I)^{-1} = U(D-\sigma I)^{-1}U^{-1}$. Observamos que as matrizes A e $(A-\sigma I)^{-1}$ possuem os mesmos autovetores associados aos autovalores λ_j e $(\lambda_j-\sigma)^{-1}$, respectivamente. Agora, supomos que σ satisfaça $|\lambda_k-\sigma| < |\lambda_j-\sigma|$ para algum k e para todo $j \neq k$. Também, Supomos que o chute inicial x_0 possa ser escrito da forma

$$x_0 = UU^{-1}x_0 = U[\xi_1 \quad \xi_2 \quad \xi_3 \quad \cdots \quad \xi_n]^T \tag{4.293}$$

com $\xi_k \neq 0$. Então

$$(A-\sigma I)^{-k}x_0 = (U(D-\sigma I)^{-k}U^{-1})U \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \vdots \\ \xi_n \end{bmatrix} \tag{4.294}$$

$$= U \begin{bmatrix} (\lambda_1-\sigma)^{-k}\xi_1 \\ (\lambda_2-\sigma)^{-k}\xi_2 \\ (\lambda_3-\sigma)^{-k}\xi_3 \\ \vdots \\ (\lambda_n-\sigma)^{-k}\xi_n \end{bmatrix} = (\lambda_i-\sigma)^{-k}\xi_1 U \begin{bmatrix} \frac{\xi_1}{\xi_i}\left(\frac{\lambda_i-\sigma}{\lambda_1-\sigma}\right)^k \\ \cdots \\ 1 \\ \vdots \\ \cdots \\ \frac{\xi_n}{\xi_i}\left(\frac{\lambda_i-\sigma}{\lambda_n-\sigma}\right)^k \end{bmatrix} \tag{4.295}$$

Como $\left|\frac{\lambda_i-\sigma}{\lambda_j-\sigma}\right| < 1$, o último vetor converge para $\pm e_i$ e

$$x_k = \frac{(A-\sigma I)^{-k}x_0}{\|(A-\sigma I)^{-k}x_0\|_2} \tag{4.296}$$

converge para $Ue_i = v_i$ que é um múltiplo do autovetor associado a λ_i. Também, a sequência

$$\lambda_k = x_k^T A x_k \tag{4.297}$$

converge para λ_i:

$$\lim_{k\to\infty} \lambda_k = v_i^T A v_i = \lambda_i v_i^T v_i = \lambda_i. \tag{4.298}$$

A método da iteração inversa tem restrições semelhantes àquelas do método da potência:

i) O chute x_0 não pode ser ortogonal ao autovetor associado ao autovalor λ_i.

ii) O chute σ deve estar mais próximo de λ_i do que dos λ_j, $j \neq i$.

A vantagem é que conseguimos calcular qualquer autovalor, não apenas o autovalor dominante.

Exercícios resolvidos

Esta seção carece de exercícios resolvidos. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

Exercícios

E 4.8.1. Calcule o autovalor dominante e o autovetor associado da matriz

$$\begin{bmatrix} 4 & 41 & 78 \\ 48 & 28 & 21 \\ 26 & 13 & 11 \end{bmatrix}. \tag{4.299}$$

Expresse sua resposta com seis dígitos significativos.

E 4.8.2. Calcule o autovalor dominante e o autovetor associado da matriz

$$\begin{bmatrix} 3 & 4 \\ 2 & -1 \end{bmatrix} \tag{4.300}$$

usando o método da potência inciando com o vetor $x = [1 \quad 1]^T$.

E 4.8.3. A norma L_2 de um matriz A é dada pela raiz quadrada do autovalor dominante da matriz A^*A (autovalor de maior módulo), isto é:

$$\|A\|_2 = \sqrt{\max\{|\lambda| : \lambda \in \sigma(A^*A)\}}. \tag{4.301}$$

Use o método da potência para obter a norma L_2 da seguinte matriz:

$$A = \begin{bmatrix} 69 & 84 & 88 \\ 15 & -40 & 11 \\ 70 & 41 & 20 \end{bmatrix}. \tag{4.302}$$

Expresse sua resposta com seis dígitos significativos.

E 4.8.4. Os autovalores de uma matriz triangular são os elementos da diagonal principal. Verifique o método da potência aplicada à seguinte matriz:

$$\begin{bmatrix} 2 & 3 & 1 \\ 0 & 3 & -1 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.303}$$

4.9 Exercícios finais

E 4.9.1. O circuito linear da Figura 4.9.1 pode ser modelado pelo sistema dado a seguir. Escreva esse sistema na forma matricial sendo as tensões V_1, V_2, V_3, V_4 e V_5 as cinco incógnitas. Resolva esse problema quando $V = 127$ e

a) $R_1 = R_2 = R_3 = R_4 = 2$ e $R_5 = R_6 = R_7 = 100$ e $R_8 = 50$

b) $R_1 = R_2 = R_3 = R_4 = 2$ e $R_5 = 50$ e $R_6 = R_7 = R_8 = 100$

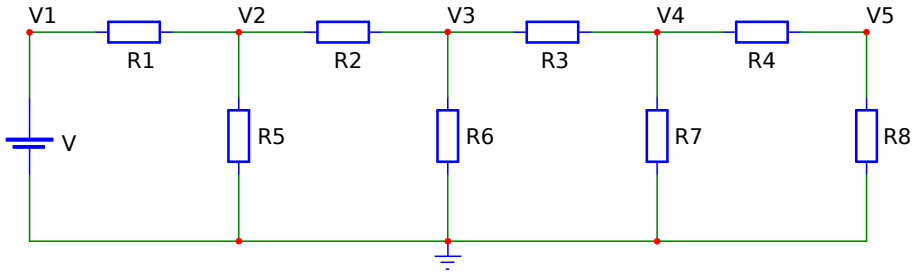
$$V_1 = V \tag{4.304}$$

$$\frac{V_1 - V_2}{R_1} + \frac{V_3 - V_2}{R_2} - \frac{V_2}{R_5} = 0 \tag{4.305}$$

$$\frac{V_2 - V_3}{R_2} + \frac{V_4 - V_3}{R_3} - \frac{V_3}{R_6} = 0 \tag{4.306}$$

$$\frac{V_3 - V_4}{R_3} + \frac{V_5 - V_4}{R_4} - \frac{V_4}{R_7} = 0 \tag{4.307}$$

$$\frac{V_4 - V_5}{R_4} - \frac{V_5}{R_8} = 0 \tag{4.308}$$



Complete a tabela abaixo representado a solução com 4 algarismos significativos:

Caso	V_1	V_2	V_3	V_4	V_5
a					
b					

Então, refaça este problema reduzindo o sistema para apenas 4 incógnitas (V_2, V_3, V_4 e V_5).

E 4.9.2. Resolva o Problema 4.9.1 pelos métodos de Jacobi e Gauss-Seidel.

E 4.9.3. (Interpolação) Resolva os seguintes problemas:

a) Encontre o polinômio $P(x) = ax^2 + bx + c$ que passa pelos pontos $(-1, -3)$, $(1, -1)$ e (2,9).

b) Encontre os coeficientes A e B da função $f(x) = A\,\mathrm{sen}\,(x) + B\cos(x)$ tais que $f(1) = 1.4$ e $f(2) = 2.8$.

c) Encontre a função $g(x) = A_1\,\mathrm{sen}\,(x) + B_1\cos(x) + A_2\,\mathrm{sen}\,(2x) + B_2\cos(2x)$ tais que $f(1) = 1$, $f(2) = 2$, $f(3) = 3$ e $f(4) = 4$.

Capítulo 5

Solução de sistemas de equações não lineares

Neste capítulo, estudaremos o método de Newton aplicado à resolução de sistemas não lineares de equações.

O método de Newton aplicado a encontrar a raiz x^* da função $y = f(x)$ estudado na Seção 3.4 consiste em um processo iterativo. Em cada passo deste processo, dispomos de uma aproximação $x^{(k)}$ para x^* e construímos uma nova aproximação $x^{(k+1)}$. Cada passo do método de Newton envolve os seguintes procedimentos:

- Linearização da função $f(x)$ no ponto $x^{(k)}$:

$$f(x) = f(x^{(k)}) + (x - x^{(k)})f'(x^{(k)}) + O\left(|x - x^{(k)}|^2\right) \tag{5.1}$$

- A aproximação $x^{(k+1)}$ é definida como o valor de x em que a linearização $f(x^{(k)}) + (x - x^{(k)})f'(x^{(k)})$ passa por zero.

Queremos, agora, generalizar o método de Newton a fim de resolver problemas de várias equações e várias incógnitas, ou seja, encontrar x_1,x_2,$\ldots$ x_n que satisfazem as seguintes equações:

$$\begin{aligned}
f_1(x_1,x_2,\ldots,x_n) &= 0 && (5.2)\\
f_2(x_1,x_2,\ldots,x_n) &= 0 && (5.3)\\
&\vdots && (5.4)\\
f_n(x_1,x_2,\ldots,x_n) &= 0 && (5.5)
\end{aligned}$$

Podemos escrever este problema na forma vetorial definindo o vetor $x = [x_1,x_2,\ldots,x_n]^T$ e a função vetorial

$$F(x) = \begin{bmatrix} f_1(x_1,x_2,\ldots,x_n) \\ f_2(x_1,x_2,\ldots,x_n) \\ \vdots \\ f_n(x_1,x_2,\ldots,x_n) \end{bmatrix}. \tag{5.6}$$

Exemplo 5.0.1. Suponha que queiramos resolver numericamente o seguinte sistema de duas equações e duas incógnitas:

$$\frac{x_1^2}{3} + x_2^2 = 1 \tag{5.7}$$

$$x_1^2 + \frac{x_2^2}{4} = 1 \tag{5.8}$$

Então definimos

$$F(x) = \begin{bmatrix} \frac{x_1^2}{3} + x_2^2 - 1 \\ \\ x_1^2 + \frac{x_2^2}{4} - 1 \end{bmatrix} \tag{5.9}$$

Neste momento, dispomos de um problema na forma $F(x) = 0$ e precisamos desenvolver uma técnica para linearizar a função $F(x)$. Para tal, precisamos de alguns conceitos do cálculo de várias variáveis.

Observe que $F(x) - F(x^{(0)})$ pode ser escrito como

$$F(x) - F(x^{(0)}) = \begin{bmatrix} f_1(x_1,x_2,\ldots,x_n) - f_1(x_1^{(0)},x_2^{(0)},\ldots,x_n^{(0)}) \\ f_2(x_1,x_2,\ldots,x_n) - f_2(x_1^{(0)},x_2^{(0)},\ldots,x_n^{(0)}) \\ \vdots \\ f_n(x_1,x_2,\ldots,x_n) - f_n(x_1^{(0)},x_2^{(0)},\ldots,x_n^{(0)}) \end{bmatrix} \tag{5.10}$$

Usamos a regra da cadeia

$$df_i = \frac{\partial f_i}{\partial x_1}dx_1 + \frac{\partial f_i}{\partial x_2}dx_2 + \cdots + \frac{\partial f_i}{\partial x_n}dx_n = \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}dx_j \tag{5.11}$$

e aproximamos as diferenças por derivadas parciais:

$$f_i(x_1,x_2,\ldots,x_n) - f_i(x_1^{(0)},x_2^{(0)},\ldots,x_n^{(0)}) \approx \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}\left(x_j - x_j^{(0)}\right) \tag{5.12}$$

Portanto,

$$F(x) - F(x^{(0)}) \approx \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \dots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \begin{bmatrix} x_1 - x_1^{(0)} \\ x_2 - x_2^{(0)} \\ \vdots \\ x_n - x_n^{(0)} \end{bmatrix}, \tag{5.13}$$

Definimos, então, a matriz jacobiana por

$$J_F = \frac{\partial(f_1, f_2, \dots, f_n)}{\partial(x_1, x_2, \dots, x_n)} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \dots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}. \tag{5.14}$$

Isto é, a matriz jacobiana de uma função ou simplesmente, o jacobiano de uma função $F(x)$ é a matriz formada pelas suas derivadas parciais:

$$(J_F)_{ij} = \frac{\partial f_i}{\partial x_j}. \tag{5.15}$$

Nestes termos, podemos reescrever (5.13) como

$$F(x) \approx F(x^{(0)}) + J_F(x^{(0)})(x - x^{(0)}) \tag{5.16}$$

Esta expressão é chamada de linearização de $F(x)$ no ponto $x^{(0)}$ e generaliza a linearização em uma dimensão dada por $f(x) \approx f(x^{(0)}) + f'(x^{(0)})(x - x^{(0)})$.

Ao longo deste capítulo, assumiremos que as seguintes bibliotecas e módulos `Python` estão importadas:

```
import numpy as np
from numpy import linalg
```

5.1 Método de Newton para sistemas

Nesta seção, construiremos o método de Newton ou Newton-Raphson generalizado para sistemas. Assumimos, portanto, que a função $F(x)$ é diferenciável e que existe um ponto x^* tal que $F(x^*) = 0$. Seja $x^{(k)}$ uma aproximação para x^*, queremos construir uma nova aproximação $x^{(k+1)}$ através da linearização de $F(x)$ no ponto $x^{(k)}$.

- Linearização da função $F(x)$ no ponto $x^{(k)}$:

$$F(x) = F(x^{(k)}) + J_F\left(x^{(k)}\right)\left(x - x^{(k)}\right) + O\left(\|x - x^{(k)}\|^2\right) \tag{5.17}$$

- A aproximação $x^{(k)}$ é definida como o ponto x em que a linearização $F(x^{(k)}) + J_F\left(x^{(k)}\right)\left(x - x^{(k)}\right)$ é nula, ou seja:

$$F(x^{(k)}) + J_F\left(x^{(k)}\right)\left(x^{(k+1)} - x^{(k)}\right) = 0 \tag{5.18}$$

Supondo que a matriz jacobina seja inversível no ponto $x^{(k)}$, temos:

$$J_F\left(x^{(k)}\right)\left(x^{(k+1)} - x^{(k)}\right) = -F(x^{(k)}) \tag{5.19}$$

$$x^{(k+1)} - x^{(k)} = -J_F^{-1}\left(x^{(k)}\right)F(x^{(k)}) \tag{5.20}$$

$$x^{(k+1)} = x^{(k)} - J_F^{-1}\left(x^{(k)}\right)F(x^{(k)}) \tag{5.21}$$

Desta forma, o método iterativo de Newton-Raphson para encontrar as raízes de $F(x) = 0$ é dado por:

$$\begin{cases} x^{(k+1)} = x^{(k)} - J_F^{-1}\left(x^{(k)}\right)F(x^{(k)}), \ \ n \geq 0 \\ x^{(0)} = \text{dado inicial} \end{cases} \tag{5.22}$$

Observação 5.1.1. Usamos subíndices para indicar o elemento de um vetor e superíndices para indicar o passo da iteração. Assim, $x^{(k)}$ se refere à iteração k e $x_i^{(k)}$ se refere à componente i no vetor $x^{(k)}$.

Observação 5.1.2. A notação $J_F^{-1}\left(x^{(k)}\right)$ enfatiza que a jacobiana deve ser calculada a cada passo.

Observação 5.1.3. Podemos definir o passo $\Delta^{(k)}$ como

$$\Delta^{(k)} = x^{(k+1)} - x^{(k)} \tag{5.23}$$

Assim, $\Delta^{(k)} = -J_F^{-1}\left(x^{(k)}\right) F(x^{(k)})$, ou seja, $\Delta^{(k)}$ resolve o problema linear:

$$J_F\left(x^{(k)}\right)\Delta^{(k)} = -F(x^{(k)}) \tag{5.24}$$

Em geral, é menos custoso resolver o sistema acima do que calcular o inverso da jacobiana e multiplicar pelo vetor $F(x^{(k)})$.

Exemplo 5.1.1. Retornamos ao nosso exemplo inicial, isto é, resolver numericamente o seguinte sistema não linear:

$$\frac{x_1^2}{3} + x_2^2 = 1 \tag{5.25}$$

$$x_1^2 + \frac{x_2^2}{4} = 1 \tag{5.26}$$

Para tal, definimos a função $F(x)$:

$$F(x) = \begin{bmatrix} \frac{x_1^2}{3} + x_2^2 - 1 \\ x_1^2 + \frac{x_2^2}{4} - 1 \end{bmatrix} \tag{5.27}$$

cuja jacobiana é:

$$J_F = \begin{bmatrix} \frac{2x_1}{3} & 2x_2 \\ 2x_1 & \frac{x_2}{2} \end{bmatrix} \tag{5.28}$$

Faremos a implementação numérica em `Python`. Para tal definimos as funções que implementarão $F(x)$ e a $J_F(x)$

```
>>> def F(x):
...     y = np.zeros(2)
...     y[0] = x[0]**2/3 + x[1]**2 - 1
...     y[1] = x[0]**2 + x[1]**2/4 - 1
...     return y
...
>>> def JF(x):
...     y = np.zeros((2,2))
...     y[0,0] = 2*x[0]/3
...     y[0,1] = 2*x[1]
...     y[1,0] = 2*x[0]
...     y[1,1] = x[1]/2
...     return y
...
```

Desta forma, se x é uma aproximação para a raiz, pode-se calcular a próxima aproximação através dos comandos:

```
>>> delta = -np.linalg.inv(JF(x)).dot(F(x))
>>> x = x + delta
```

Ou simplesmente

```
>>> x = x - np.linalg.inv(JF(x)).dot(F(x))
```

Observe que as soluções exatas desse sistema são $\left(\pm\sqrt{\frac{9}{11}}, \pm\sqrt{\frac{8}{11}}\right)$.

Exemplo 5.1.2. Encontre uma aproximação para a solução do sistema

$$x_1^2 = \cos(x_1 x_2) + 1 \tag{5.29}$$
$$\operatorname{sen}(x_2) = 2\cos(x_1) \tag{5.30}$$

que fica próxima ao ponto $x_1 = 1{,}5$ e $x_2 = 0{,}5$.

Solução. Vamos, aqui, dar as principais ideias para obter a solução usando o método de Newton. Começamos definindo nossa aproximação inicial por $x^{(1)} = (1{,}5, 0{,}5)$. Então iteramos:

$$x^{(n+1)} = x^{(n)} - J_F^{-1}(x)F(x), \quad n \geq 1. \tag{5.31}$$

onde

$$F(x) = \begin{bmatrix} x_1^2 - \cos(x_1 x_2) - 1 \\ \operatorname{sen}(x_2) - 2\cos(x_1) \end{bmatrix} \tag{5.32}$$

e sua jacobiana é

$$J_F(x) = \begin{bmatrix} 2x_1 + x_2 \operatorname{sen}(x_1 x_2) & x_1 \operatorname{sen}(x_1 x_2) \\ 2\operatorname{sen}(x_1) & \cos(x_2) \end{bmatrix} \tag{5.33}$$

As iterações convergem para $x = (1{,}3468109,\ 0{,}4603195)$.

Em `Python`, podemos implementá-las com o seguinte código:

```
def F(x):
    y = np.zeros(2)

    y[0] = x[0]**2 - np.cos(x[0]*x[1]) - 1
    y[1] = np.sin(x[1]) - 2*np.cos(x[0])
```

```
    return y

def JF(x):
    y = np.zeros((2,2))

    y[0,0] = 2*x[0] + x[1]*np.sin(x[0]*x[1])
    y[0,1] = x[0]*np.sin(x[0]*x[1])

    y[1,0] =  2*np.sin(x[0])
    y[1,1] = np.cos(x[1])

    return y
```

E agora, basta iterar:

```
>>> x = np.array([1.5,0.5])
>>> x=x-np.linalg.inv(JF(x)).dot(F(x))
```

◊

5.1.1 Código Python: Newton para Sistemas

```
from __future__ import division
import numpy as np
from numpy import linalg

def newton(F,JF,x0,TOL,N):
    #preliminares
    x = np.copy(x0).astype('double')
    k=0
    #iteracoes
    while (k < N):
       k += 1
       #iteracao Newton
       delta = -np.linalg.inv(JF(x)).dot(F(x))
       x = x + delta
       #criterio de parada
       if (np.linalg.norm(delta,np.inf) < TOL):
           return x

    raise NameError('num. max. iter. excedido.')
```

Exercícios resolvidos

Esta seção carece de exercícios resolvidos. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

Exercícios

E 5.1.1. Faça o que se pede:

a) Encontre o gradiente da função

$$f(x,y) = x^2y + \cos(xy) - 4 \tag{5.34}$$

b) Encontre a matriz jacobiana associada à função

$$F(x,y) = \begin{bmatrix} x\cos(x) + y \\ e^{-2x+y} \end{bmatrix}. \tag{5.35}$$

c) Encontre a matriz jacobiana associada à função

$$L(x) = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 - y_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 - y_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 - y_3 \end{bmatrix}. \tag{5.36}$$

E 5.1.2. Encontre uma aproximação numérica para o seguinte problema não linear de três equações e três incógnitas:

$$2x_1 - x_2 = \cos(x_1) \tag{5.39}$$
$$-x_1 + 2x_2 - x_3 = \cos(x_2) \tag{5.40}$$
$$-x_2 + x_3 = \cos(x_3) \tag{5.41}$$

Partindo das seguintes aproximações iniciais:

a) $x^{(0)} = [1,\ 1,\ 1]^T$

b) $x^{(0)} = [-0{,}5,\ -2,\ -3]^T$

c) $x^{(0)} = [-2,\ -3,\ -4]^T$

d) $x^{(0)} = [0,\ 0,\ 0]^T$

E 5.1.3. Encontre os pontos de intersecção entre a parábola $y = x^2 + 1$ e a elipse $x^2 + y^2/4 = 1$ seguindo os seguintes passos:

a) Faça um esboço das duas curvas e entenda o problema. Verifique que existem dois pontos de intersecção, um no primeiro quadrante e outro no segundo quadrante do plano xy.

b) A partir de seu esboço, encontre aproximações para x e y em cada ponto.

c) Escreva o problema na forma $F\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

d) Encontre a jacobiana J_F.

e) Construa a iteração do método de Newton.

f) Implemente no computador.

g) Resolva o sistema analiticamente e compare as respostas.

E 5.1.4. Encontre os pontos de intersecção entre a parábola $y = x^2$ e a curva $y = \cos(x)$ seguindo os seguintes passos:

a) Faça um esboço das duas curvas, entenda o problema. Verifique que existem dois pontos de intersecção, um no primeiro quadrante e outro no segundo quadrante do plano xy.

b) A partir de seu esboço, encontre aproximações para x e y em cada ponto.

c) Escreva o problema na forma $F\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

d) Encontre a jacobiana J_F.

e) Construa a iteração do método de Newton.

f) Implemente no `Python`.

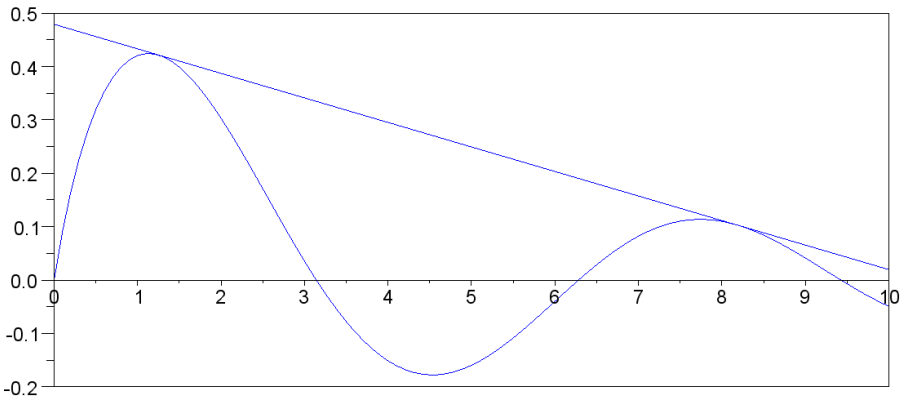


Figura 5.1: Reta bitangente a uma curva.

g) Transforme o sistema em um problema de uma única variável e compare com a resposta do Problema 3.4.1.

E 5.1.5. Encontre uma aproximação com erro inferior a 10^{-5} em cada incógnita para a solução próxima da origem do sistema

$$\begin{aligned} 6x - 2y + e^z &= 2 && (5.42)\\ \operatorname{sen}(x) - y + z &= 0 && (5.43)\\ \operatorname{sen}(x) + 2y + 3z &= 1 && (5.44) \end{aligned}$$

E 5.1.6. (Entenda casos particulares)

- Considere a função $L(x) = Ax - b$, onde A é uma matriz $n \times n$ inversível e b um vetor coluna em $\mathbb{R}^n$. O que acontece quando aplicamos o método de Newton para encontrar as raízes de $L(x)$?
- Mostre que o método de Newton-Raphson aplicado a uma função diferenciável do tipo $f : \mathbb{R} \to \mathbb{R}$ se reduz ao método de Newton estudado na primeira área.

E 5.1.7. Considere a função $f(x) = \frac{\operatorname{sen}(x)}{x+1}$, encontre a equação da reta que tangencia dois pontos da curva $y = f(x)$ próximos ao primeiro e segundo ponto de máximo no primeiro quadrante, respectivamente. Veja a Figura 5.1.

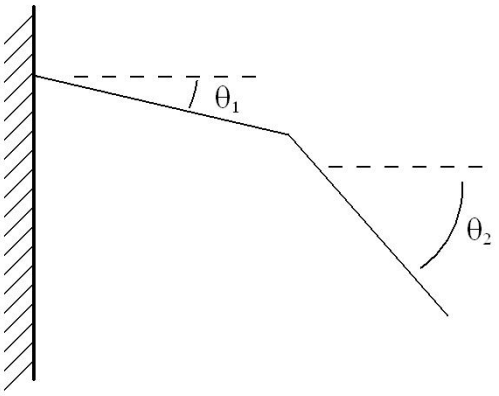


Figura 5.2: Sistema mecânico com dois segmentos.

E 5.1.8. (Estática) Considere o sistema mecânico constituído de dois segmentos de mesmo comprimento L presos entre si e a uma parede por articulações conforme a Figura 5.2.

O momento em cada articulação é proporcional à deflexão com constante de proporcionalidade k. Os segmentos são feitos de material homogêneo de peso P. A condição de equilíbrio pode ser expressa em termos dos ângulos θ_1 e θ_2 conforme:

$$k\theta_1 = \frac{3PL}{2}\cos\theta_1 + k(\theta_2 - \theta_1) \tag{5.51}$$

$$k(\theta_2 - \theta_1) = \frac{PL}{2}\cos\theta_2 \tag{5.52}$$

Considere $P = 100N$, $L = 1m$ e calcule os ângulos θ_1 e θ_2 quando:

a) $k = 1000$ Nm/rad

b) $k = 500$ Nm/rad

c) $k = 100$ Nm/rad

d) $k = 10$ Nm/rad

Obs: Você deve escolher valores para iniciar o método. Como você interpretaria fisicamente a solução para produzir palpites iniciais satisfatórios? O que se altera entre o caso a e o caso d?

E 5.1.9. (estática - problemas de três variáveis) Considere, agora, o sistema mecânico semelhante ao do Problema 5.1.8, porém constituído de três segmentos de mesmo comprimento L presos entre si e a uma parede por articulações.

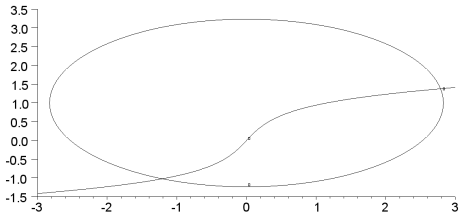


Figura 5.3: intersecção entre duas curvas.

O momento em cada articulação é proporcional à deflexão com constante de proporcionalidade k. Os segmentos são feitos de material homogêneo de peso P. A condição de equilíbrio pode ser expressa em termos dos ângulos θ_1, θ_2 e θ_3 conforme:

$$k\theta_1 = \frac{5PL}{2}\cos\theta_1 + k(\theta_2 - \theta_1) \tag{5.53}$$

$$k(\theta_2 - \theta_1) = \frac{3PL}{2}\cos\theta_2 + k(\theta_3 - \theta_2) \tag{5.54}$$

$$k(\theta_3 - \theta_2) = \frac{PL}{2}\cos\theta_3 \tag{5.55}$$

Considere $P = 10\text{N}$, $L = 1\text{m}$ e calcule os ângulos θ_1, θ_2 e θ_3 quando:

a) $k = 1000\text{Nm/rad}$

b) $k = 100\text{Nm/rad}$

c) $k = 10\text{Nm/rad}$

E 5.1.10. Considere o problema de encontrar os pontos de intersecção das curvas descritas por (ver Figura 5.3):

$$\frac{x^2}{8} + \frac{(y-1)^2}{5} = 1 \tag{5.56}$$

$$\tag{5.57}$$

$$\tan^{-1}(x) + x = y + y^3 \tag{5.58}$$

Com base no gráfico, encontre soluções aproximadas para o problema e use-as para iniciar o método de Newton-Raphson. Encontre as raízes com erro inferior a 10^{-5}.

E 5.1.11. Considere o sistema de equações dado por

$$\frac{(x-3)^2}{16} + \frac{(y-1)^2}{36} = 1 \tag{5.59}$$

$$\tanh(x) + x = 2\,\text{sen}\; y - 0.01y^3 \tag{5.60}$$

Usando procedimentos analíticos, determine uma região limitada do plano onde se encontram necessariamente todas as raízes do problema. Encontre as raízes desse sistema com pelo menos quatro dígitos significativos corretos usando o método de Newton. Você deve construir o método de Newton indicando as funções envolvidas e calculando a matriz jacobiana analiticamente. Use que $\frac{d}{du}\tanh u = 1 - \tanh^2 u$, se precisar.

E 5.1.12. (Otimização) Uma indústria consome energia elétrica de três usinas fornecedoras. O custo de fornecimento em reais por hora como função da potência consumida em kW é dada pelas seguintes funções

$$C_1(x) = 10 + .3x + 10^{-4}x^2 + 3.4 \cdot 10^{-9}x^4 \quad (5.61)$$
$$C_2(x) = 50 + .25x + 2 \cdot 10^{-4}x^2 + 4.3 \cdot 10^{-7}x^3 \quad (5.62)$$
$$C_3(x) = 500 + .19x + 5 \cdot 10^{-4}x^2 + 1.1 \cdot 10^{-7}x^4 \quad (5.63)$$

Calcule a distribuição de consumo que produz custo mínimo quando a potência total consumida é $1500kW$. Dica: Denote por x_1, x_2 e x_3 as potências consumidas das usinas 1, 2 e 3, respectivamente. O custo total será dado por $C(x_1,x_2,x_3) = C_1(x_1) + C_2(x_2) + C_3(x_3)$ enquanto o consumo total é $x_1 + x_2 + x_3 = 1500$. Isto é, queremos minimizar a função custo total dada por:

$$C(x_1,x_2,x_3) = C_1(x_1) + C_2(x_2) + C_3(x_3) \quad (5.64)$$

restrita à condição

$$G(x_1,x_2,x_3) = x_1 + x_2 + x_3 - 1500 = 0. \quad (5.65)$$

Pelos multiplicadores de Lagrange, temos que resolver o sistema dado por:

$$\nabla C(x_1,x_2,x_3) = \lambda \nabla G(x_1,x_2,x_3) \quad (5.66)$$
$$G(x_1,x_2,x_3) = 0 \quad (5.67)$$

E 5.1.13. Encontre a função do tipo $f(x) = Ab^x$ que melhor aproxima os pontos (0, 3,1), (1, 4,4) e (2, 6,7) pelo critério dos mínimos quadrados. Dica: Você deve encontrar os valores de A e b que minimizam o resíduo dado por

$$R = [3{,}1 - f(0)]^2 + [4{,}4 - f(1)]^2 + [6{,}7 - f(2)]^2 . \quad (5.68)$$

Dica: Para construir aproximações para resposta e iniciar o método, considere a função $f(x) = Ab^x$ que passa pelo primeiro e terceiro ponto.

E 5.1.14. Encontre o valor máximo da função

$$f(x,y) = -x^4 - y^6 + 3xy^3 - x \tag{5.69}$$

na região $(x,y) \in [-2,0] \times [-2,0]$ seguindo os seguintes passos:

a) Defina a função $z = f(x,y) = -x^4 - y^6 + 3xy^3 - x$ e trace o gráfico de contorno na região.

b) Com base no gráfico, encontre valores aproximados para as coordenadas xy do ponto de máximo.

c) Sabendo que o ponto de máximo acontece quando o gradiente é nulo, escreva o problema como um sistema de duas equações não lineares e duas incógnitas.

d) Implemente o método de Newton.

E 5.1.15. A função $f(x,y,z) = \operatorname{sen}(x) + \operatorname{sen}(2y) + \operatorname{sen}(3z)$ possui um máximo quando $x = \pi/2$, $y = \pi/4$ e $z = \pi/6$. Calcule numericamente este ponto.

E 5.1.16. Encontre as raizes do problema

$$\begin{aligned} 3x - \cos(yz + z) - 1/2 &= 0 && (5.70)\\ 4x^2 - 25y^2 + 0.4y + 2 &= 0 && (5.71)\\ e^{-xy} + 2x - 5z &= 10 && (5.72) \end{aligned}$$

no cubo $|x| < 2, |y| < 2, |z| < 2$. Dica: Reduza a um problema de duas incógnitas e use recursos gráficos para aproximar as raízes na região.

E 5.1.17. Considere o seguinte sistema de equações não lineares:

$$\begin{aligned} x_1 - x_2 &= 0\\ -x_{j-1} + 5(x_j + x_j^3) - x_{j+1} &= 10\exp(-j/3), \quad 2 \le j \le 10\\ x_{11} &= 1 \end{aligned} \tag{5.73}$$

a) Escreva este sistema na forma $F(x) = 0$ onde $x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{11} \end{bmatrix}$ e calcule analiticamente a matriz jacobiana $\frac{\partial(F_1,\ldots,F_{11})}{\partial(x_1,\ldots x_{11})}$. Dica: Use a regularidade nas expressões para abreviar a notação.

b) Construa a iteração para encontrar a única solução deste problema pelo método de Newton e, usando esse método, encontre uma solução aproximada com erro absoluto inferior a 10^{-4}.

E 5.1.18. Considere a função

$$f(x,y) = \frac{e^{-(x-1)^2-(y-2)^2}}{1+x^2+y^2} \tag{5.76}$$

a) Encontre o valor máximo desta função.

b) Usando multiplicadores de Lagrange, encontre o valor máximo desta função restrito à condição

$$(x-1)^2 + (y-2)^2 = 1. \tag{5.77}$$

c) Parametrize a circunferência para transformar o problema de máximo com restrição em um problema de uma única variável. Resolva usando as técnicas de equações lineares de uma variável.

5.2 Linearização de uma função de várias variáveis

Nesta seção, discutimos de forma distinta e mais rigorosa os conceitos de matriz jacobiana e linearização de uma função de várias variáveis.

5.2.1 Gradiente

Considere primeiramente uma função $f : \mathbb{R}^n \to \mathbb{R}$, ou seja, uma função que mapeia n variáveis reais em um único real, por exemplo:

$$f(x) = x_1^2 + x_2^2/4 \tag{5.78}$$

Para construirmos a linearização, fixemos uma direção no espaço $\mathbb{R}^n$, ou seja, um vetor v:

$$v = [v_1, v_2, \cdots, v_n]^T \tag{5.79}$$

Queremos estudar como a função $f(x)$ varia quando "andamos" na direção v a partir do ponto $x^{(0)}$. Para tal, inserimos um parâmetro real pequeno h, dizemos que

$$x = x^{(0)} + hv \tag{5.80}$$

e definimos a função auxiliar

$$g(h) = f(x^0 + hv). \tag{5.81}$$

Observamos que a função $g(h)$ é uma função de $\mathbb{R}$ em $\mathbb{R}$.

A linearização de $g(h)$ em torno de $h = 0$ é dada por

$$g(h) = g(0) + hg'(0) + O(h^2) \tag{5.82}$$

Observamos que $g(h) = f(x^{(0)} + hv)$ e $g(0) = f(x^{(0)})$. Precisamos calcular $g'(0)$:

$$g'(h) = \frac{d}{dh}g(h) = \frac{d}{dh}f(x^{(0)} + hv). \tag{5.83}$$

Pela regra da cadeia temos:

$$\frac{d}{dh}f(x^{(0)} + hv) = \sum_{j=1}^{n} \frac{\partial f}{\partial x_j}\frac{dx_j}{dh}. \tag{5.84}$$

Observamos que $x_j = x_j^{(0)} + hv_j$, portanto

$$\frac{dx_j}{dh} = v_j \tag{5.85}$$

Assim:

$$\frac{d}{dh}f(x^{(0)} + hv) = \sum_{j=1}^{n} \frac{\partial f}{\partial x_j}v_j. \tag{5.86}$$

Observamos que esta expressão pode ser vista como o produto interno entre o gradiente de f e o vetor v:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \qquad v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \tag{5.87}$$

Na notação cálculo vetorial escrevemos este produto interno como $\nabla f \cdot v = v \cdot \nabla f$ na notação de produto matricial, escrevemos $(\nabla f)^T v = v^T \nabla f$. Esta quantidade é conhecida como **derivada direcional** de f no ponto $x^{(0)}$ na direção v, sobretudo quando $\|v\| = 1$.

Podemos escrever a linearização $g(h) = g(0) + hg'(0) + O(h^2)$ como

$$f(x^{(0)} + hv) = f(x^{(0)}) + h\nabla^T f(x^{(0)})\, v + O(h^2) \tag{5.88}$$

Finalmente, escrevemos $x = x^{(0)} + hv$, ou seja, $hv = x - x^{(0)}$

$$f(x) = f(x^{(0)}) + \nabla^T f(x^{(0)})\,(x - x^{(0)}) + O(\|x - x^{(0)}\|^2) \tag{5.89}$$

Observação 5.2.1. Observe a semelhança com a linearização no caso em uma dimensão. A notação $\nabla^T f(x^{(0)})$ é o transposto do vetor gradiente associado à função $f(x)$ no ponto $x^{(0)}$:

$$\nabla^T f(x^{(0)}) = \left[\frac{\partial f\left(x^{(0)}\right)}{\partial x_1},\ \frac{\partial f\left(x^{(0)}\right)}{\partial x_2},\ \cdots,\ \frac{\partial f\left(x^{(0)}\right)}{\partial x_n}\right] \tag{5.90}$$

5.2.2 Matriz jacobiana

Interessamo-nos, agora, pela linearização da função $F : \mathbb{R}^n \to \mathbb{R}^n$. Lembramos que $F(x)$ pode ser escrita como um vetor de funções $f_j : \mathbb{R}^{\kappa} \to \mathbb{R}$:

$$F(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{bmatrix} \tag{5.91}$$

Linearizando cada uma das funções f_j, temos:

$$F(x) \;=\; \underbrace{\begin{bmatrix} f_1\left(x^{(0)}\right) + \nabla^T f_1(x^{(0)})\,\left(x - x^{(0)}\right) + O(\|x - x^{(0)}\|^2) \\ \\ f_2\left(x^{(0)}\right) + \nabla^T f_2(x^{(0)})\,\left(x - x^{(0)}\right) + O(\|x - x^{(0)}\|^2) \\ \\ \vdots \\ \\ f_n\left(x^{(0)}\right) + \nabla^T f_n(x^{(0)})\,\left(x - x^{(0)}\right) + O(\|x - x^{(0)}\|^2) \end{bmatrix}}_{\text{Vetor coluna}} \tag{5.92}$$

ou, equivalentemente:

$$F(x) = \underbrace{\begin{bmatrix} f_1\left(x^{(0)}\right) \\ f_2\left(x^{(0)}\right) \\ \vdots \\ f_n\left(x^{(0)}\right) \end{bmatrix}}_{\text{Vetor coluna}} + \underbrace{\begin{bmatrix} \nabla^T f_1(x^{(0)}) \\ \nabla^T f_2(x^{(0)}) \\ \vdots \\ \nabla^T f_n(x^{(0)}) \end{bmatrix}}_{\text{Matriz jacobiana}} \underbrace{\left(x - x^{(0)}\right)}_{\text{Vetor coluna}} + O(\|x - x^{(0)}\|^2) \quad (5.93)$$

Podemos escrever a linearização de $F(x)$ na seguinte forma mais enxuta:

$$F(x) = F\left(x^{(0)}\right) + J_F(x^{(0)})\left(x - x^{(0)}\right) + O\left(\left\|x - x^{(0)}\right\|^2\right) \quad (5.94)$$

A matriz jacobiana J_F é matriz cujas linhas são os gradientes transpostos de f_j, ou seja:

$$J_F = \frac{\partial(f_1, f_2, \ldots, f_n)}{\partial(x_1, x_2, \ldots, x_n)} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \quad (5.95)$$

A matriz jacobiana de uma função ou simplesmente, o jacobiano de uma função $F(x)$ é a matriz formada pelas suas derivadas parciais:

$$(J_F)_{ij} = \frac{\partial f_i}{\partial x_j} \quad (5.96)$$

Exemplo 5.2.1. Calcule a matriz jacobiana da função

$$F(x) = \begin{bmatrix} \frac{x_1^2}{3} + x_2^2 - 1 \\ \\ x_1^2 + \frac{x_2^2}{4} - 1 \end{bmatrix} \tag{5.97}$$

$$J_F = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \frac{2x_1}{3} & 2x_2 \\ \\ 2x_1 & \frac{x_2}{2} \end{bmatrix} \tag{5.98}$$

Capítulo 6

Interpolação

Neste capítulo, discutimos os problemas de **interpolação**. Mais precisamente, dada uma sequência de n reais $x_1 < x_2 < \ldots < x_n$, um conjunto de pontos $\{(x_i, y_i) \in I \times \mathbb{R}\}_{i=1}^{n}$, onde $I = [x_1,x_n]$ e uma família de funções $\mathcal{F}_I = \{\varphi : I \to \mathbb{R}\}$, o problema de interpolação consiste em encontrar alguma função $f \in \mathcal{F}_I$ tal que

$$f(x_i) = y_i, \quad i = 1, 2, \ldots, n. \tag{6.1}$$

Chamamos uma tal f de **função interpoladora** dos pontos dados. Ou ainda, dizemos que f interpola os pontos dados.

Exemplo 6.0.1. Um dos problemas de interpolação mais simples é o de encontrar a equação da reta que passa por dois pontos dados. Por exemplo, sejam dados o conjunto de pontos $\{(1, 1), (2, 2)\}$ e a família de funções $\mathcal{F}_{[1,2]}$:

$$\mathcal{F}_{[1,2]} = \{f : [1,2] \to \mathbb{R} \ ; \ [1,2] \ni x \mapsto f(x) = a + bx; \ a,b \in \mathbb{R}\}\,. \tag{6.2}$$

Para que uma f na família seja a função interpoladora do conjunto de pontos dados, precisamos que

$$\begin{array}{l} a + bx_1 = y_1 \\ a + bx_2 = y_2 \end{array} \quad \text{isto é} \quad \begin{array}{l} a + b = 1 \\ a + 2b = 2 \end{array} \tag{6.3}$$

o que nos fornece $a = 0$ e $b = 1$. Então, a função interpoladora f é tal que $f(x) = x$ para um $x \in [1,2]$. Os pontos e a reta interpolada estão esboçados na Figura 6.1.

Um problema de interpolação cuja a família de funções constitui-se de polinômios é chamado de problema de interpolação polinomial.

Ao longo do capítulo, faremos alguns comentários usando códigos em `Python 2.7`. Nestes, assumiremos que os seguintes módulos estão carregados:

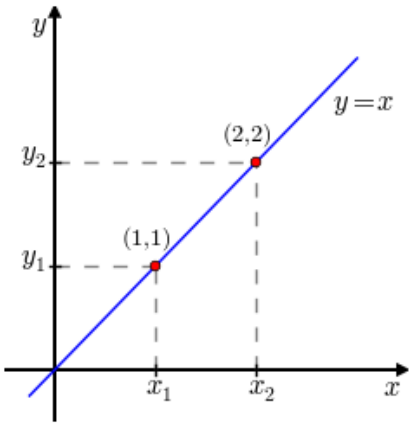


Figura 6.1: Exemplo de interpolação de dois pontos por uma reta, veja o Exemplo 6.0.1.

```
from __future__ import division
import numpy as np
from numpy import linalg
from numpy.polynomial import polynomial as poly
import matplotlib.pyplot as plt
```

6.1 Interpolação polinomial

Interpolação polinomial é um caso particular do problema geral de interpolação, no qual a família de funções é constituída de polinômios. A escolha de polinômios como funções interpolantes é natural por diversos motivos, entre eles: se p é um polinômio de grau n, o valor $p(x)$ para um x real é calculado através de $n+1$ operações de multiplicação e $n+1$ operações de adição. Para tanto, pode-se usar o algoritmo de Horner[1]. Dado um polinômio p de grau n da forma

$$p(x) = \sum_{k=0}^{n} a_k x^k, \tag{6.4}$$

é possível reescrevê-lo como a sequência de operações dada por

$$a_0 + x\left(a_1 + x\left(a_2 + x\left(\ldots + x\left(a_{n-1} + xa_n\right)\ldots\right)\right)\right). \tag{6.5}$$

Também, derivadas e primitivas de polinômios são também polinômios cuja relação algébrica com o original é simples. Além disso, o teorema da aproximação

[1] William George Horner, 1786 - 1837, matemático britânico.

de Weierstrass estabelece que qualquer função contínua definida em um intervalo fechado pode ser aproximada uniformemente por um polinômio tão bem quanto se queira.

Teorema 6.1.1 (Weierstrass)**.** *Seja f uma função contínua definida no intervalo fechado $[a,b]$ e seja δ um número positivo. Então existe um polinômio p, tal que para todo $x \in [a,b]$,*

$$|f(x) - p(x)| < \delta. \tag{6.6}$$

Observe que para o problema ser bem determinado, é necessário restringirmos o grau dos polinômios. Dado um conjunto de n pontos a serem interpolados $\{(x_i,y_i)\}_{i=1}^n$, $x_i \neq x_j$ para $i \neq j$, a família de polinômios $\mathcal{F} = \mathbb{P}_{n-1}$ deve ser escolhida, onde:

$$\mathbb{P}_{n-1} := \left\{ p : x \mapsto p(x) = \sum_{k=0}^{n-1} a_k x^k;\ \{a_0,a_1,\ldots,a_{n-1}\} \in \mathbb{R} \right\}, \tag{6.7}$$

isto é, a família dos polinômios reais de grau menor ou igual a $n-1$.

O Exemplo 6.0.1 discute um dos casos mais simples de interpolação polinomial, o qual consiste em interpolar uma reta por dois pontos. Neste caso, a família de funções consiste de polinômios de grau 1. Se buscarmos interpolar uma parábola pelos dois pontos dados, o problema fica subdeterminado, pois existem infinitas parábolas que passam por dois pontos dados. Além disso, se buscarmos interpolar uma reta por três pontos dados, o problema estaria sobredeterminado e poderia não ter solução se os pontos não fossem colineares. Veja o Exercício **??**.

Assim, dado um conjunto com n pontos $\{(x_i,y_i)\}_{i=1}^n$, chamamos de **polinômio interpolador** o polinômio de grau menor ou igual a $n-1$ que os interpola.

Exemplo 6.1.1. Encontre o polinômio interpolador do conjunto de pontos $\{(0,1), (1,6), (2,5), (3,-8)\}$.

Solução. Como o conjunto consiste de 4 pontos, o polinômio interpolador deve ser da forma:

$$p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3. \tag{6.8}$$

As condições de interpolação são $p(x_i) = y_i$, $i = 0,1,2,3$, o que nos leva ao sistema linear:

$$\begin{array}{rcrcrcrcl} a_0 & & & & & & & = & 1 \\ a_0 & + & a_1 & + & a_2 & + & a_3 & = & 6 \\ a_0 & + & 2a_1 & + & 4a_2 & + & 8a_3 & = & 5 \\ a_0 & + & 3a_1 & + & 9a_2 & + & 27a_3 & = & -8 \end{array} \tag{6.9}$$

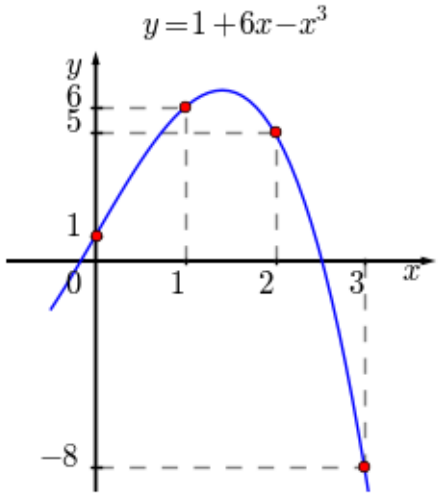


Figura 6.2: Polinômio interpolador do conjunto de pontos $\{(0,1),\ (1,6),\ (2,5),\ (3,-8)\}$. Veja o Exemplo 6.1.1.

cuja solução é $a_0 = 1$, $a_1 = 6$, $a_2 = 0$ e $a_3 = -1$. Portanto, o polinômio interpolador é $p(x) = 1 + 6x - x^3$. Veja Figura 6.2.

Em `Python`, podemos encontrar o polinômio interpolador e esboçar seu gráfico com os seguintes comandos:

```
>>> xi = np.array([0,1,2,3], dtype='double')
>>> yi = np.array([1,6,5,-8], dtype='double')
>>> A = np.array([xi**3,xi**2,xi**1,xi**0]).transpose()
>>> a = np.linalg.inv(A).dot(yi);a
array([ -1.,  0.,  6.,  1. ])
>>> xx = np.linspace(-0.5,3.25);
>>> plt.plot(xi,yi,'ro',xx,np.polyval(a,xx),'b-')
>>> plt.grid();plt.show()
```

◊

Teorema 6.1.2. *Seja $\{(x_i,y_i)\}_{i=1}^n$ um conjunto de n pares ordenados de números reais tais que $x_i \neq x_j$ se $i \neq j$, então existe um único polinômio $p(x)$ de grau $n-1$ ou inferior que passa por todos os pontos dados, isto é, $p(x_i) = y_i, i = 1, \ldots, n$.*

Demonstração. Observe que o problema de encontrar os coeficientes a_0, a_1,..., a_{n-1} do polinômio

$$p(x) = a_0 + a_1 x + a_2 x^2 + \cdots a_{n-1} x^{n-1} = \sum_{k=0}^{n-1} a_k x^k \tag{6.10}$$

tal que $p(x_i) = y_i$ é equivalente a resolver o sistema linear com n equações e n incógnitas dado por

$$\begin{aligned} a_0 + a_1x_1 + a_1x_1^2 + \cdots + a_{n-1}x_1^{n-1} &= y_1, \\ a_0 + a_1x_2 + a_2x_2^2 + \cdots + a_{n-1}x_2^{n-1} &= y_2, \\ &\vdots \\ a_0 + a_1x_n + a_2x_n^2 + \cdots + a_{n-1}x_n^{n-1} &= y_n. \end{aligned} \tag{6.11}$$

O qual pode ser escrito na forma matricial como

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & \cdots & x_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} \tag{6.12}$$

A matriz envolvida é uma **matriz de Vandermonde**[2] de ordem n cujo determinante é dado pelo produtório duplo

$$\prod_{1 \le i < j \le n} (x_j - x_i) \tag{6.13}$$

É fácil ver que se as abscissas são diferentes dois a dois, então o determinante é não nulo. Disto decorre que a matriz envolvida é inversível e, portanto, o sistema possui uma solução que é única. □

Esta abordagem direta que usamos no Exemplo 6.1.1 e na demonstração do Teorema 6.1.2 se mostra ineficiente quando o número de pontos é grande e quando existe grande variação nas abscissas. Neste caso, a matriz de Vandermonde é mal condicionada (ver [6]), o que acarreta um aumento dos erros de arredondamento na solução do sistema.

Uma maneira de resolver este problema é escrever o polinômio em uma base que produza um sistema bem condicionado.

Exercícios resolvidos

Esta seção carece de exercícios resolvidos. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

[2] Alexandre-Théophile Vandermonde, 1735 - 1796, matemático francês.

ER 6.1.1. Mostre que:

a) Existem infinitas parábolas que interpolam dois pontos dados $\{(x_1, y_1), (x_2, y_2)\}$, com $x_1 \neq x_2$.

b) Não existe reta que interpola os pontos $\{(1, 1), (2, 2{,}1), (3, 3)\}$.

c) Não existe parábola de equação $y = a_0 + a_1x + a_2x^2$ que interpola dois pontos dados $\{(x_1, y_1), (x_1, y_2)\}$, com $y_1 \neq y_2$. Mas, existem infinitas parábolas de equação $x = a_0 + a_1y + a_2y^2$ que interpolam estes pontos.

Solução. a) Uma parábola de equação $y = a_1 + a_2x + a_3x^2$ que interpola os pontos deve satisfazer o sistema:

$$\begin{aligned} a_1 + a_2x_1 + a_3x_1^2 &= y_1 \\ a_1 + a_2x_2 + a_3x_2^2 &= y_2 \end{aligned} \tag{6.14}$$

Sem perda de generalidade, para cada $a_3 \in \mathbb{R}$ dado, temos:

$$\begin{aligned} a_1 + a_2x_1 &= y_1 - a_3x_1^2 \\ a_1 + a_2x_2 &= y_2 - a_3x_2^2, \end{aligned} \tag{6.15}$$

o qual tem solução única, pois $x_1 \neq x_2$. Ou seja, para cada $a_3 \in \mathbb{R}$ dado, existem $a_1, a_2 \in \mathbb{R}$ tais que a parábola de equação $y = a_1 + a_2x + a_3x^2$ interpola os pontos dados.

b) Certamente não existem retas de equação $x = a$ que interpolam os pontos dados. Consideremos então retas de equação $y = a_1 + a_2x$. Para uma tal reta interpolar os pontos dados é necessário que:

$$\begin{aligned} a_1 + a_2 &= 1 \\ a_1 + 2a_2 &= 2{,}1, \\ a_1 + 3a_2 &= 3 \end{aligned} \tag{6.16}$$

o qual é um sistema impossível.

c) Não existe uma parábola de equação $y = a_1 + a_2x + a_3x^2$ que interpole os pontos dados, pois tal equação determina uma função de x em y. Agora, para mostrar que existem infinitas parábolas de equação $x = a_1 + a_2y + a_3y^2$ que interpolam os pontos dados, basta seguir um raciocínio análogo ao do item a), trocando x por y e y por x.

$\Diamond$

Exercícios

Esta seção carece de exercícios. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

E 6.1.1. Encontre o polinômio interpolador para o conjunto de pontos $\{(-2,-47), (0,-3), (1,4), (2,41)\}$. Então, faça um gráfico com os pontos e o polinômio interpolador encontrado.

E 6.1.2. Encontre o polinômio interpolador para o conjunto de pontos $\{(-1,1{,}25), (0{,}5,0{,}5), (1,1{,}25), (1{,}25,1{,}8125)\}$.

6.2 Diferenças divididas de Newton

Dado um conjunto com n pontos $\{(x_i,y_i)\}_{i=1}^n$, o **método das diferenças divididas de Newton** consiste em construir o polinômio interpolador da forma

$$\begin{aligned} p(x) = a_1 &+ a_2(x-x_1) + a_3(x-x_1)(x-x_2) + \cdots \\ &+ a_n(x-x_1)(x-x_2)\cdots(x-x_{n-1}). \end{aligned} \tag{6.17}$$

Como $p(x_i) = y_i$, $i = 1,2,\ldots,n$, os coeficientes a_i satisfazem o seguinte sistema triangular inferior:

$$\begin{aligned} a_1 &= y_1 \\ a_1 + a_2(x_2-x_1) &= y_2 \\ a_1 + a_2(x_3-x_1) + a_3(x_3-x_1)(x_3-x_2) &= y_3 \\ &\vdots \\ a_1 + a_2(x_n-x_1) + \cdots + a_n(x_n-x_1)\cdots(x_n-x_{n-1}) &= y_n \end{aligned} \tag{6.18}$$

Resolvendo de cima para baixo, obtemos

$$\begin{aligned} a_1 &= y_1 \\ a_2 &= \frac{y_2-a_1}{x_2-x_1} = \frac{y_2-y_1}{x_2-x_1} \\ a_3 &= \frac{y_3 - a_2(x_3-x_1) - a_1}{(x_3-x_1)(x_3-x_2)} = \frac{\frac{y_3-y_2}{(x_3-x_2)} - \frac{y_2-y_1}{(x_2-x_1)}}{(x_3-x_1)} \\ &\cdots \end{aligned} \tag{6.19}$$

Tabela 6.1: Esquema de diferenças divididas para um conjunto com três pontos $\{(x_i, y_i)\}_{i=1}^{3}$.

j	x_j	$f[x_j]$	$f[x_{j-1},x_j]$	$f[x_{j-2},x_{j-1},x_j]$
1	x_1	$\boldsymbol{f[x_1]} = y_1$		
			$\boldsymbol{f[x_1,x_2]} = \dfrac{f[x_2]-f[x_1]}{x_2-x_1}$	
2	x_2	$f[x_2] = y_2$		$\boldsymbol{f[x_1,x_2,x_3]} = \dfrac{f[x_2,x_3]-f[x_1,x_2]}{x_3-x_1}$
			$f[x_2,x_3] = \dfrac{f[x_3]-f[x_2]}{x_3-x_2}$	
3	x_3	$f[x_3] = y_3$		

Note que os coeficientes são obtidos por diferenças das ordenadas divididas por diferenças das abscissas dos pontos dados. Para vermos isso mais claramente, introduzimos a seguinte notação:

$$f[x_j] := y_j \tag{6.20}$$

$$f[x_j, x_{j+1}] := \frac{f[x_{j+1}] - f[x_j]}{x_{j+1} - x_j} \tag{6.21}$$

$$f[x_j, x_{j+1}, x_{j+2}] := \frac{f[x_{j+1}, x_{j+2}] - f[x_j, x_{j+1}]}{x_{j+2} - x_j} \tag{6.22}$$

$$\vdots \tag{6.23}$$

$$f[x_j, x_{j+1}, \ldots, x_{j+k}] := \frac{f[x_{j+1}, x_{j+2}, \ldots, x_{j+k}] - f[x_j, x_{j+1}, \ldots, x_{j+k-1}]}{x_{j+k} - x_j} \tag{6.24}$$

Chamamos $f[x_j]$ de diferença dividida de ordem zero (ou primeira diferença dividida), $f[x_i,x_j+1]$ de diferença dividida de ordem 1 (ou segunda diferença dividida) e assim por diante.

Uma inspeção cuidadosa dos coeficientes obtidos em (6.19) nos mostra que

$$a_k = f[x_1,x_2,\ldots,x_k] \tag{6.25}$$

Isto nos permite esquematizar o método conforme apresentado na Tabela 6.1.

Exemplo 6.2.1. Use o método de diferenças divididas para encontrar o polinômio que passe pelos pontos $(-1,3)$,$(0,1)$,$(1,3)$,$(3,43)$.

Solução. Usando o esquema apresentado na Tabela 6.1, obtemos

j	x_j	$f[x_j]$	$f[x_{j-1},x_j]$	$f[x_{j-2},x_{j-1},x_j]$	$f[x_{j-3},x_{j-2},x_{j-1},x_j]$
1	-1	**3**			
			$\frac{1-3}{0-(-1)} = \mathbf{-2}$		
2	0	1		$\frac{2-(-2)}{1-(-1)} = \mathbf{2}$	
			$\frac{3-1}{1-0} = 2$		$\frac{6-2}{3-(-1)} = \mathbf{1}$
3	1	3		$\frac{20-2}{3-0} = 6$	
			$\frac{43-3}{3-1} = 20$		
4	3	43			

Portanto, o polinômio interpolador do conjunto de pontos dados é

$$p(x) = 3 - 2(x+1) + 2(x+1)x + (x+1)x(x-1) \tag{6.26}$$

ou, equivalentemente, $p(x) = x^3 + 2x^2 - x + 1$.

Em `Python`, podemos fazer as contas acima da segunte forma:

```
x = np.array([-1,0,1,3], dtype="double")
y = np.array([3,1,3,43], dtype="double")
#inicializando a tabela
T = np.zeros((4,4));
#primeira coluna
T[:,0]=y;
#segunda coluna
T[1,1]=(T[1,0]-T[0,0])/(x[1]-x[0]);
T[2,1]=(T[2,0]-T[1,0])/(x[2]-x[1]);
T[3,1]=(T[3,0]-T[2,0])/(x[3]-x[2]);
#terceira coluna
T[2,2]=(T[2,1]-T[1,1])/(x[2]-x[0]);
T[3,2]=(T[3,1]-T[2,1])/(x[3]-x[1]);
#quarta coluna
T[3,3]=(T[3,2]-T[2,2])/(x[3]-x[0]);
print(T)
#polinomio interpolador
p = np.array([T[0,0]], dtype="double")
paux = np.array([-x[0],1], dtype="double")
p.resize(2)
p += T[1,1]*paux
```

```
paux = poly.polymul(paux,[-x[1],1])
p.resize(3)
p += T[2,2]*paux
paux = poly.polymul(paux,[-x[2],1])
p.resize(4)
p += T[3,3]*paux
```

$\Diamond$

6.3 Polinômios de Lagrange

Outra maneira clássica de resolver o problema da interpolação polinomial é através dos polinômios de Lagrange. Dado um conjunto de pontos $\{x_j\}_{j=1}^n$ distintos dois a dois, definimos os polinômios de Lagrange como os polinômios de grau $n-1$ que satisfazem

$$L_k(x_j) = \begin{cases} 1, & \text{se } k = j \\ 0, & \text{se } k \neq j \end{cases} \tag{6.27}$$

Assim, o polinômio $p(x)$ de grau $n-1$ que interpola os pontos dados, isto é, $p(x_j) = y_j, j = 1, \ldots, n$ é dado por

$$p(x) = y_1 L_1(x) + y_2 L_2(x) + \cdots + y_n L_n(x) = \sum_{k=1}^{n} y_k L_k(x). \tag{6.28}$$

Para construir os polinômios de Lagrange, podemos analisar a sua forma fatorada, ou seja:

$$L_k(x) = c_k \prod_{\substack{j=1 \\ j\neq i}}^{n} (x - x_j) \tag{6.29}$$

onde o coeficiente c_k é obtido da condição $L_k(x_k) = 1$:

$$L_k(x_k) = c_k \prod_{\substack{j=1 \\ j\neq i}}^{n} (x_k - x_j) \Longrightarrow c_k = \frac{1}{\prod\limits_{\substack{j=1 \\ j\neq i}}^{n} (x_k - x_j)} \tag{6.30}$$

Portanto,

$$L_k(x) = \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{(x - x_j)}{(x_k - x_j)} \tag{6.31}$$

Observação 6.3.1. O problema de interpolação quando escrito usando como base os polinômios de Lagrange produz um sistema linear diagonal.

Exemplo 6.3.1. Encontre o polinômio da forma $p(x) = a_1 + a_2x + a_3x^2 + a_4x^3$ que passa pelos pontos $(0,0)$, $(1,1)$, $(2,4)$, $(3,9)$.

Solução. Escrevemos:

$$L_1(x) = \frac{(x-1)(x-2)(x-3)}{(0-1)(0-2)(0-3)} = -\frac{1}{6}x^3 + x^2 - \frac{11}{6}x + 1 \tag{6.32}$$

$$L_2(x) = \frac{x(x-2)(x-3)}{1(1-2)(1-3)} = \frac{1}{2}x^3 - \frac{5}{2}x^2 + 3x \tag{6.33}$$

$$L_3(x) = \frac{x(x-1)(x-3)}{2(2-1)(2-3)} = -\frac{1}{2}x^3 + 2x^2 - \frac{3}{2}x \tag{6.34}$$

$$L_4(x) = \frac{x(x-1)(x-2)}{3(3-1)(3-2)} = \frac{1}{6}x^3 - \frac{1}{2}x^2 + \frac{1}{3}x \tag{6.35}$$

Assim, temos:

$$P(x) = 0 \cdot L_1(x) + 1 \cdot L_2(x) + 4 \cdot L_3(x) + 9 \cdot L_4(x) = x^2 \tag{6.36}$$

$\Diamond$

6.4 Aproximação de funções reais por polinômios interpoladores

Teorema 6.4.1. *Dados $n+1$ pontos distintos, x_0, x_1, $\cdots$, x_n, dentro de um intervalo $[a,b]$ e uma função f com $n+1$ derivadas contínuas nesse intervalo ($f \in C^{n+1}[a,b]$), então para cada x em $[a,b]$, existe um número $\xi(x)$ em (a,b) tal que*

$$f(x) = P(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x-x_0)(x-x_1)\cdots(x-x_n), \tag{6.37}$$

onde $P(x)$ é o polinômio interpolador. Em especial, pode-se dizer que

$$|f(x) - P(x)| \leq \frac{M}{(n+1)!}\left|(x-x_0)(x-x_1)\cdots(x-x_n)\right|, \tag{6.38}$$

onde

$$M = \max_{x\in[a,b]} |f^{(n+1)}(\xi(x))| \tag{6.39}$$

Exemplo 6.4.1. Considere a função $f(x) = \cos(x)$ e o polinômio $P(x)$ de grau 2 tal que $P(0) = \cos(0) = 1$, $P(\frac{1}{2}) = \cos(\frac{1}{2})$ e $P(1) = \cos(1)$. Use a fórmula de Lagrange para encontrar $P(x)$. Encontre o erro máximo que se assume ao aproximar o valor de $\cos(x)$ pelo de $P(x)$ no intervalo [0,1]. Trace os gráficos de $f(x)$ e $P(x)$ no intervalo [0,1] no mesmo plano cartesiano e, depois, trace o gráfico da diferença $\cos(x) - P(x)$. Encontre o erro efetivo máximo $|\cos(x) - P(x)|$.

Solução. Usando polinômios de Lagrange, obtemos

$$\begin{aligned} P(x) &= 1\frac{(x-\frac{1}{2})(x-1)}{(0-\frac{1}{2})(0-1)} \\ &+ \cos\left(\frac{1}{2}\right)\frac{(x-0)(x-1)}{(\frac{1}{2}-0)(\frac{1}{2}-1)} \\ &+ \cos(1)\frac{(x-0)(x-\frac{1}{2})}{(1-0)(1-\frac{1}{2})} \qquad (6.40) \\ &\approx 1 - 0{,}0299720583066x - 0{,}4297256358252x^2 \qquad (6.41) \end{aligned}$$

Aqui, cabe um código Python explicativo. Escreva você mesmo o código.

Veja como participar da escrita do livro em:

https://github.com/livroscolaborativos/CalculoNumerico

Para estimar o erro máximo, precisamos estimar a derivada terceira de $f(x)$:

$$|f'''(x)| = |\operatorname{sen}(x)| \leq \operatorname{sen}(1) < 0{,}85 \qquad (6.42)$$

e, assim,

$$\max_{x\in[0,1]} \left| x\left(x-\frac{1}{2}\right)(x-1)\right|. \qquad (6.43)$$

O polinômio de grau três $Q(x) = x\left(x-\frac{1}{2}\right)(x-1)$ tem um mínimo (negativo) em $x_1 = \frac{3+\sqrt{3}}{6}$ e um máximo (positivo) em $x_2 = \frac{3-\sqrt{3}}{6}$. Logo:

$$\max_{x\in[0,1]} \left| x\left(x-\frac{1}{2}\right)(x-1)\right| \leq \max\{|Q(x_1)|,\, |Q(x_2)|\} \approx 0{,}0481125. \qquad (6.44)$$

Portanto:

$$|f(x) - P(x)| < \frac{0{,}85}{3!}0{,}0481125 \approx 0{,}0068159 < 7\cdot 10^{-3} \qquad (6.45)$$

Para estimar o erro efetivo máximo, basta encontrar o máximo de $|P(x) - \cos(x)|$. O mínimo (negativo) de $P(x) - \cos(x)$ acontece em $x_1 = 4{,}29\cdot 10^{-3}$ e o máximo (positivo) acontece em $x_2 = 3{,}29\cdot 10^{-3}$. Portanto, o erro máximo efetivo é $4{,}29\cdot 10^{-3}$. ♢

Exemplo 6.4.2. Considere o problema de aproximar o valor da integral $\int_0^1 f(x)dx$ pelo valor da integral do polinômio $P(x)$ que coincide com $f(x)$ nos pontos $x_0 = 0$, $x_1 = \frac{1}{2}$ e $x_2 = 1$. Use a fórmula de Lagrange para encontrar $P(x)$. Obtenha o valor de $\int_0^1 P(x)dx$ e encontre uma expressão para o erro de truncamento.

O polinômio interpolador de $f(x)$ é

$$P(x) = f(0)\frac{(x-\frac{1}{2})(x-1)}{(0-\frac{1}{2})(0-1)} + f\left(\frac{1}{2}\right)\frac{(x-0)(x-1)}{(\frac{1}{2}-0)(\frac{1}{2}-1)} + f(1)\frac{(x-0)(x-\frac{1}{2})}{(1-0)(1-\frac{1}{2})} \quad (6.46)$$

$$= f(0)(2x^2-3x+1) + f\left(\frac{1}{2}\right)(-4x^2+4x) + f(1)(2x^2-x) \quad (6.47)$$

e a integral de $P(x)$ é:

$$\int_0^1 P(x)dx = \left[f(0)\left(\frac{2}{3}x^3 - \frac{3}{2}x^2 + x\right)\right]_0^1 + \left[f\left(\frac{1}{2}\right)\left(-\frac{4}{3}x^3 + 2x^2\right)\right]_0^1 \quad (6.48)$$

$$+ \left[f(1)\left(\frac{2}{3}x^3 - \frac{1}{2}x^2\right)\right]_0^1 \quad (6.49)$$

$$= f(0)\left(\frac{2}{3} - \frac{3}{2} + 1\right) + f\left(\frac{1}{2}\right)\left(-\frac{4}{3} + 2\right) + f(1)\left(\frac{2}{3} - \frac{1}{2}\right) \quad (6.50)$$

$$= \frac{1}{6}f(0) + \frac{2}{3}f\left(\frac{1}{2}\right) + \frac{1}{6}f(1) \quad (6.51)$$

Para fazer a estimativa de erro usando o Teorema 6.4.1 e temos

$$\left|\int_0^1 f(x)dx - \int_0^1 P(x)dx\right| = \left|\int_0^1 f(x) - P(x)dx\right| \quad (6.52)$$

$$\leq \int_0^1 |f(x) - P(x)|dx \quad (6.53)$$

$$\leq \frac{M}{6}\int_0^1 \left|x\left(x-\frac{1}{2}\right)(x-1)\right|dx \quad (6.54)$$

$$= \frac{M}{6}\left[\int_0^{1/2} x\left(x-\frac{1}{2}\right)(x-1)dx \right. \quad (6.55)$$

$$\left. - \int_{1/2}^1 x\left(x-\frac{1}{2}\right)(x-1)dx\right] \quad (6.56)$$

$$= \frac{M}{6}\left[\frac{1}{64} - \left(-\frac{1}{64}\right)\right] = \frac{M}{192}. \quad (6.57)$$

Lembramos que $M = \max_{x\in[0,1]} |f'''(x)|$.

Observação 6.4.1. Existem estimativas melhores para o erro de truncamento para este esquema de integração numérica. Veremos com mais detalhes tais esquemas na teoria de integração numérica.

Exemplo 6.4.3. Use o resultado do exemplo anterior para aproximar o valor das seguintes integrais:

a) $\int_0^1 \ln(x+1)dx$

b) $\int_0^1 e^{-x^2}dx$

Solução. Usando a fórmula obtida, temos que

$$\int_0^1 \ln(x+1)dx \approx 0{,}39 \pm \frac{1}{96} \tag{6.58}$$

$$\int_0^1 e^{-x^2}dx \approx 0{,}75 \pm \frac{3{,}87}{192} \tag{6.59}$$

$\Diamond$

Exercícios

E 6.4.1. Use as mesmas técnicas usadas o resultado do Exemplo 6.4.2 para obter uma aproximação do valor de:

$$\int_0^1 f(x)dx \tag{6.60}$$

através do polinômio interpolador que coincide com $f(x)$ nos pontos $x = 0$ e $x = 1$.

6.5 Interpolação linear segmentada

Considere o conjunto $(x_i,y_i)_{j=1}^n$ de n pontos. Assumiremos que $x_{i+1} > x_i$, ou seja, as abscissas são distintas e estão em ordem crescente. A função linear que interpola os pontos x_i e x_{i+1} no intervalo i é dada por

$$P_i(x) = y_i\frac{(x_{i+1}-x)}{(x_{i+1}-x_i)} + y_{i+1}\frac{(x-x_i)}{(x_{i+1}-x_i)} \tag{6.61}$$

O resultado da interpolação linear segmentada é a seguinte função contínua definida por partes no intervalo $[x_1,x_n]$:

$$f(x) = P_i(x), \qquad x \in [x_i,x_{i+1}] \tag{6.62}$$

Exemplo 6.5.1. Construa uma função linear por partes que interpola os pontos (0,0), (1,4), (2,3), (3,0), (4,2), (5,0).

A função procurada pode ser construída da seguinte forma:

$$f(x) = \begin{cases} 0\frac{x-1}{0-1} + 1\frac{x-0}{1-0}, & 0 \leq x < 1 \\ 4\frac{x-2}{1-2} + 3\frac{x-1}{2-1}, & 1 \leq x < 2 \\ 3\frac{x-3}{2-3} + 0\frac{x-2}{3-2}, & 2 \leq x < 3 \\ 0\frac{x-4}{3-4} + 2\frac{x-3}{4-3}, & 3 \leq x < 4 \\ 2\frac{x-5}{4-5} + 0\frac{x-4}{5-4}, & 4 \leq x \leq 5 \end{cases} \tag{6.63}$$

Simplificando, obtemos:

$$f(x) = \begin{cases} x, & 0 \leq x < 1 \\ -x + 5, & 1 \leq x < 2 \\ -3x + 9, & 2 \leq x < 3 \\ 2x - 6, & 3 \leq x < 4 \\ -2x + 10, & 4 \leq x \leq 5 \end{cases} \tag{6.64}$$

A Figura 6.3 é um esboço da função $f(x)$ obtida.

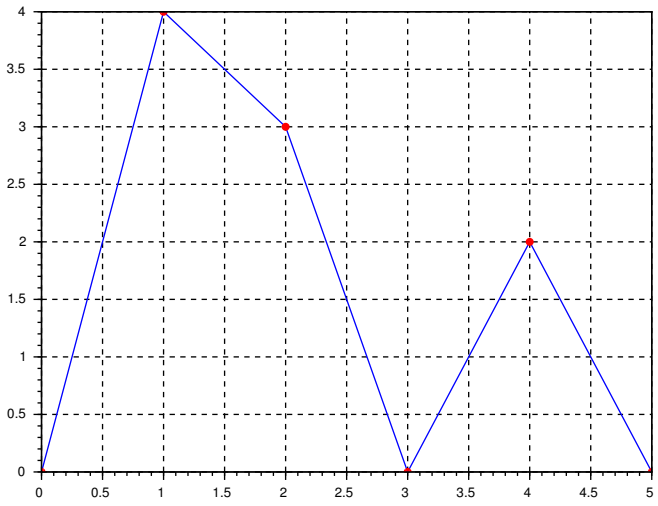


Figura 6.3: Interpolação linear segmentada.

6.6 Interpolação cúbica segmentada - spline

A ideia empregada na interpolação linear segmentada pode ser estendida através da utilização de polinômios de grau superior. A escolha de polinômios de grau superior implica uma maior liberdade (há um número maior de coeficientes) na construção da interpolação. Parte dessa liberdade pode ser utilizada na exigência de suavidade para a interpolação.

Definição 6.6.1 (spline de ordem m)**.** *Dado um conjunto de n pontos $\mathcal{I} = \{(x_j,y_j)\}_{j=1}^{n}$ tais que $x_{j+1} > x_j$, ou seja, as abscissas são distintas e estão em ordem crescente; um spline de ordem m que interpola estes pontos é uma função s com as seguintes propriedades:*

i) Em cada intervalo $[x_j,x_{j+1})$, $j = 1,2,\ldots n-2$ e no segmento $[x_{n-1},x_n]$ s é um polinômio de grau menor ou igual a m;

ii) Em algum dos intervalos s é um polinômio de grau m;

iii) Em cada $x_j \in \mathcal{I}$, $s(x_j) = y_j$, isto é, o spline interpola os pontos dados;

iv) s é uma função de classe $\mathcal{C}^{m-1}$, isto é, é função $m-1$ vezes continuamente diferenciável.

São $n-1$ intervalos e em cada um deles há $m+1$ coeficientes a se determinar. As condições *iii* e *iv* impostas pela definição correspondem respectivamente a n e $m(n-2)$ equações. Estas últimas, se devem à exigência de continuidade nos pontos internos, ou seja, os pontos de $\mathcal{I}$ com índices $j = 2,3,\ldots,n-1$. Portanto, há $m-1$ coeficientes a mais do que o número de equações e, à exceção do caso $m = 1$ (interpolação linear segmentada), o problema é subdeterminado. Ou seja, uma vez fixada a ordem $m > 1$, existem infinitos splines de ordem m que interpolam os pontos do conjunto $\mathcal{I}$.

O caso $m = 3$, denominado spline cúbico, é de grande interesse pois reproduz o comportamento físico de réguas delgadas com estrutura elástica homogênea e perfil uniforme sujeitas aos vínculos representados pelos pontos do conjunto $\mathcal{I}$. A equação diferencial que rege o comportamento do perfil dessas réguas é um caso particular do equação da viga de Euler-Bernoulli. Neste caso, a equação tem a forma

$$\frac{d^4y}{dx^4} = 0, \tag{6.65}$$

cuja solução geral é um polinômio de grau 3.

Vamos supor que um spline cúbico que interpola o conjunto de pontos $\mathcal{I}$ é conhecido. Como esse spline é uma função de classe $\mathcal{C}^2$, as suas derivadas nos pontos do conjunto $\mathcal{I}$ são conhecidas também. Seja y'_j, o valor dessa derivada em

$x = x_j$. Agora, vamos considerar dois pares de pontos sucessivos de $\mathcal{I}$, (x_j,y_j) e (x_{j+1},y_{j+1}). A forma do spline cúbico no intervalo $[x_j,x_{j+1})$ pode ser identificada com a solução da equação diferencial (6.65) no intervalo (x_j,x_{j+1}) sujeita às condições de contorno

$$y(x_j) = y_j, \quad y'(x_j) = y'_j, \quad y(x_{j+1}) = y_{j+1} \quad \text{e} \quad y'(x_{j+1}) = y'_{j+1}. \tag{6.66}$$

A solução desse problema de contorno é escrita de modo conveniente como

$$s_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3, \tag{6.67}$$

onde as constantes a_j, b_j, c_j e d_j se relacionam às do problema de contorno. As duas primeiras seguem imediatamente das condições de contorno em x_j:

$$a_j = y_j \quad \text{e} \quad b_j = y'_j. \tag{6.68}$$

As duas últimas são obtidas pela solução do sistema de equações formado pelas condições de contorno em x_{j+1}:

$$c_j = 3\frac{y_{j+1} - y_j}{(x_{j+1} - x_j)^2} - \frac{y'_{j+1} + 2y'_j}{x_{j+1} - x_j} \quad \text{e} \quad d_j = -2\frac{y_{j+1} - y_j}{(x_{j+1} - x_j)^3} + \frac{y'_{j+1} + y'_j}{(x_{j+1} - x_j)^2} \tag{6.69}$$

Esta relação entre o conjunto de valores para a derivada de um spline cúbico $\{y'_j\}_{j=1}^n$ nos pontos de interpolação $\mathcal{I}$ e os coeficientes dos polinômios em cada intervalo de interpolação pode ser resumida na seguinte proposição:

Proposição 6.6.1. *Seja s um spline cúbico que interpola o conjunto de pontos $\mathcal{I} = \{(x_j,y_j)\}_{j=1}^n \subset \mathbb{R}^2$ tais que $x_{j+1} > x_j$. Se $\{y'_j\}_{j=1}^n$ é o conjunto dos valores da derivada de s em x_j, então em cada intervalo $[x_j,x_{j+1})$ (fechado também à direita quando $j = n-1$) o spline é igual a s_j:*

$$s_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3, \tag{6.70}$$

onde

$$\begin{aligned} a_j &= y_j, \quad c_j = 3\frac{y_{j+1} - y_j}{h_j^2} - \frac{y'_{j+1} + 2y'_j}{h_j}, \\ b_j &= y'_j, \quad d_j = -2\frac{y_{j+1} - y_j}{h_j^3} + \frac{y'_{j+1} + y'_j}{h_j^2} \end{aligned} \tag{6.71}$$

e

$$h_j = x_{j+1} - x_j, \quad j = 1,2,\ldots,n-1 \tag{6.72}$$

é a distância entre as abscissas de dois pontos de interpolação consecutivos.

De acordo com a proposição anterior, toda informação sobre um spline cúbico é armazenada no conjunto $\{(x_j,y_j,y'_j)\}_{j=1}^n$. Por construção, uma função s definida a partir de (6.70), (6.71) e (6.72) com um conjunto $\{(x_j,y_j,y'_j)\}_{j=1}^n \subset \mathbb{R}^3$, onde $x_{j+1} > x_j$ é de classe $\mathcal{C}^1$ mas não necessariamente um spline cúbico. Para ser um spline cúbico, os valores do conjunto $\{y'_j\}_{j=1}^n$ devem garantir a continuidade da derivada segunda de s em todo intervalo (x_1,x_n). Ou seja, devemos ter

$$\lim_{x \nearrow x_{j+1}} s''_j(x) = s''_{j+1}(x_{j+1}) \tag{6.73}$$

em todos os pontos internos $j = 1,2,\ldots,n-2$. Em termos dos coeficientes dos polinômios cúbicos (6.70), a equação anterior assume a forma

$$2c_j + 6d_j h_j = 2c_{j+1}, \quad j = 1,2,\ldots,n-2. \tag{6.74}$$

Esta última equação e (6.71) permitem construir um sistema de equações lineares para as variáveis y'_j:

Proposição 6.6.2. *Dado o conjunto de pontos $\mathcal{I} = \{(x_j,y_j)\}_{j=1}^n \subset \mathbb{R}^2$ tais que $x_{j+1} > x_j$, as derivadas de um spline cúbico que interpola os pontos $\mathcal{I}$, y'_j, $j = 1,2,\ldots,n$ satisfazem o sistema de equações algébricas lineares*

$$h_j y'_{j-1} + 2(h_{j-1}+h_j)y'_j + h_{j-1}y'_{j+1} = 3\left(h_j \frac{y_j - y_{j-1}}{h_{j-1}} + h_{j-1}\frac{y_{j+1}-y_j}{h_j}\right), \tag{6.75}$$

onde $j = 2,3,\ldots,n-1$ e $h_j = x_{j+1} - x_j$.

O sistema de equações (6.75) é subdeterminado. São n variáveis e $n-2$ equações. A inclusão de duas equações adicionais linearmente independentes das $n-2$ equações (6.75) possibilita a existência de uma única solução. Tipicamente essas equações adicionais envolvem o comportamento do spline na fronteira ou na sua vizinhança. A seguir, veremos quatro escolhas mais conhecidas.

6.6.1 Spline natural

Uma forma de definir as duas equações adicionais para completar o sistema (6.75) é impor condições de fronteira livres (ou naturais), ou seja,

$$s''(x_1) = s''(x_n) = 0. \tag{6.76}$$

De acordo com (6.70) essas equações implicam respectivamente

$$c_1 = 0 \quad \text{e} \quad 2c_{n-1} + 6d_{n-1}h_{n-1} = 0, \tag{6.77}$$

ou seja,

$$\begin{cases} 2y_1' + y_2' = 3\dfrac{y_2 - y_1}{h_1} \\ \\ y_{n-1}' + 2y_n' = 3\dfrac{y_n - y_{n-1}}{h_{n-1}} \end{cases}. \tag{6.78}$$

Essas duas equações em conjunto com as equações (6.75) formam um sistema de n equações algébricas lineares $Ay' = z$, onde

$$A = \begin{bmatrix} 2 & 1 & 0 & 0 & \cdots & 0 & 0 \\ h_2 & 2(h_1+h_2) & h_1 & 0 & \cdots & 0 & 0 \\ 0 & h_3 & 2(h_2+h_3) & h_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & h_{n-1} & 2(h_{n-1}+h_{n-2}) & h_{n-2} \\ 0 & 0 & 0 & \cdots & 0 & 1 & 2 \end{bmatrix}, \tag{6.79}$$

$$y' = \begin{bmatrix} y_1' \\ y_2' \\ \vdots \\ y_n' \end{bmatrix} \quad \text{e} \quad z = 3\begin{bmatrix} \frac{y_2-y_1}{h_1} \\ h_2\frac{y_2-y_1}{h_1} + h_1\frac{y_3-y_2}{h_2} \\ h_3\frac{y_3-y_2}{h_2} + h_2\frac{y_4-y_3}{h_3} \\ \vdots \\ h_{n-1}\frac{y_{n-1}-y_{n-2}}{h_{n-2}} + h_{n-2}\frac{y_n-y_{n-1}}{h_{n-1}} \\ \frac{y_n-y_{n-1}}{h_{n-1}} \end{bmatrix}. \tag{6.80}$$

Observe que a matriz A é diagonal dominante estrita e, portanto, o sistema $Ay' = z$ possui solução única. Calculado y', os valores dos a_j, b_j, c_j e d_j são obtidos diretamente pelas expressões (6.71).

Exemplo 6.6.1. Construa um spline cúbico natural que passe pelos pontos $(2, 4{,}5)$, $(5, -1{,}9)$, $(9, 0{,}5)$ e $(12, -0{,}5)$.

Solução. O spline desejado é uma função definida por partes da forma:

$$s(x) = \begin{cases} a_1 + b_1(x-2) + c_1(x-2)^2 + d_1(x-2)^3, & 2 \le x < 5 \\ a_2 + b_2(x-5) + c_2(x-5)^2 + d_2(x-5)^3, & 5 \le x < 9 \\ a_3 + b_3(x-9) + c_3(x-9)^2 + d_3(x-9)^3, & 9 \le x \le 12 \end{cases}. \tag{6.81}$$

As variáveis y'_1, y'_2, y'_3 e y'_4 resolvem o sistema $Ay' = z$, onde

$$A = \begin{bmatrix} 2 & 1 & 0 & 0 \\ 4 & 2(4+3) & 3 & 0 \\ 0 & 3 & 2(3+4) & 4 \\ 0 & 0 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 0 & 0 \\ 4 & 14 & 3 & 0 \\ 0 & 3 & 14 & 4 \\ 0 & 0 & 1 & 2 \end{bmatrix}, \tag{6.82}$$

$$y = \begin{bmatrix} y'_1 \\ y'_2 \\ y'_3 \\ y'_4 \end{bmatrix} \quad \text{e} \quad z = 3 \begin{bmatrix} \frac{1}{3}(-1{,}9 - 4{,}5) \\ \frac{4}{3}(-1{,}9 - 4{,}5) + \frac{3}{4}(0{,}5 - (-1{,}9)) \\ \frac{3}{4}(0{,}5 - (-1{,}9)) + \frac{4}{3}(-0.5 - (0{,}5)) \\ \frac{1}{3}(-0{,}5 - (0.5)) \end{bmatrix} = \begin{bmatrix} -6{,}4 \\ -20{,}2 \\ 1{,}4 \\ -1 \end{bmatrix}. \tag{6.83}$$

A solução é $y'_1 = -2{,}8\bar{3}$, $y'_2 = -0{,}7\bar{3}$, $y'_3 = 0{,}4\bar{6}$ e $y'_4 = -0{,}7\bar{3}$. Calculamos os coeficientes usando as expressões (6.71):

$$\begin{aligned} a_1 &= y_1 = 4{,}5, & b_1 &= y'_1 = -2{,}8\bar{3}, \\ a_2 &= y_2 = -1{,}9, & b_2 &= y'_2 = -0{,}7\bar{3}, \\ a_3 &= y_3 = 0{,}5. & b_3 &= y'_3 = 0{,}4\bar{6}, \end{aligned} \tag{6.84}$$

$$\begin{aligned} c_1 &= 0, & d_1 &= 0{,}0\bar{7}, \\ c_2 &= 0{,}7, & d_2 &= -0{,}091\bar{6}, \\ c_3 &= -0{,}4, & d_3 &= 0{,}0\bar{4}. \end{aligned}$$

Portanto:

$$S(x) = \begin{cases} 4{,}5 - 2{,}8\bar{3}(x-2) + 0{,}0\bar{7}(x-2)^3 & , 2 \le x < 5 \\ -1{,}9 - 0{,}7\bar{3}(x-5) + 0{,}7(x-5)^2 - 0{,}091\bar{6}(x-5)^3 & , 5 \le x < 9 \\ 0{,}5 + 0{,}4\bar{6}(x-9) - 0{,}4(x-9)^2 + 0{,}0\bar{4}(x-9)^3 & , 9 \le x \le 12 \end{cases}. \tag{6.85}$$

$\Diamond$

6.6.2 Spline fixado

O spline fixado s é obtido pela escolha dos valores das derivadas nas extremidades do intervalo de interpolação. Isto diminui o número de variáveis para $n-2$ pois y'_1 e y'_n deixam de ser incógnitas.

As equações (6.75) formam um sistema de $n-2$ equações $Ay' = z$, onde

$$A = \begin{bmatrix} 2(h_1+h_2) & h_1 & 0 & 0 & \cdots & 0 & 0 \\ h_3 & 2(h_2+h_3) & h_2 & 0 & \cdots & 0 & 0 \\ 0 & h_4 & 2(h_3+h_4) & h_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & h_{n-2} & 2(h_{n-3}+h_{n-2}) & h_{n-3} \\ 0 & 0 & 0 & \cdots & 0 & h_{n-1} & 2(h_{n-2}+h_{n-1}) \end{bmatrix}, \tag{6.86}$$

$$y' = \begin{bmatrix} y'_2 \\ y'_3 \\ \vdots \\ y'_{n-1} \end{bmatrix} \quad \text{e} \quad z = 3\begin{bmatrix} h_2\frac{y_2-y_1}{h_1} + h_1\frac{y_3-y_2}{h_2} - h_2y'_1 \\ h_3\frac{y_3-y_2}{h_2} + h_2\frac{y_4-y_3}{h_3} \\ \vdots \\ h_{n-2}\frac{y_{n-2}-y_{n-3}}{h_{n-3}} + h_{n-3}\frac{y_{n-1}-y_{n-2}}{h_{n-2}} \\ h_{n-1}\frac{y_{n-1}-y_{n-2}}{h_{n-2}} + h_{n-2}\frac{y_n-y_{n-1}}{h_{n-1}} - h_{n-2}y'_n \end{bmatrix}. \tag{6.87}$$

Observe que a matriz A é diagonal dominante estrita e, portanto, o sistema $Ay' = z$ possui solução única.

6.6.3 Spline *not-a-knot*

O spline *not-a-knot* é definido com um spline cúbico que satisfaz as equações adicionais

$$\lim_{x \nearrow x_2} s'''_1(x) = s'''_2(x_2) \quad \text{e} \quad \lim_{x \nearrow x_{n-1}} s'''_{n-2}(x) = s'''_{n-1}(x_{n-1}). \tag{6.88}$$

Em termos dos coeficientes (6.70), as equações anteriores correspondem a

$$d_1 = d_2 \quad \text{e} \quad d_{n-2} = d_{n-1}, \tag{6.89}$$

ou seja,

$$\begin{cases} h_2^2y'_1 + (h_2^2 - h_1^2)y'_2 - h_1^2y'_3 = 2\left(h_2^2\dfrac{y_2-y_1}{h_1} - h_1^2\dfrac{y_3-y_2}{h_2}\right) \\ \\ h_{n-1}^2y'_{n-2} + (h_{n-1}^2 - h_{n-2}^2)y'_{n-1} - h_{n-2}^2y'_n = 2\left(h_{n-1}^2\dfrac{y_{n-1}-y_{n-2}}{h_{n-2}} - h_{n-2}^2\dfrac{y_n-y_{n-1}}{h_{n-1}}\right) \end{cases}. \tag{6.90}$$

Essas duas equações agregadas às equações (6.75) formam um sistema de n equações $Ay' = z$, onde

$$A = \begin{bmatrix} h_2^2 & h_2^2 - h_1^2 & -h_1^2 & 0 & \cdots & 0 & 0 \\ h_2 & 2(h_1+h_2) & h_1 & 0 & \cdots & 0 & 0 \\ 0 & h_3 & 2(h_2+h_3) & h_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & h_{n-1} & 2(h_{n-2}+h_{n-1}) & h_{n-2} \\ 0 & 0 & 0 & \cdots & h_{n-1}^2 & h_{n-1}^2 - h_{n-2}^2 & -h_{n-2}^2 \end{bmatrix}, \tag{6.91}$$

$$y' = \begin{bmatrix} y_1' \\ y_2' \\ \vdots \\ y_n' \end{bmatrix} \quad \text{e} \quad z = \begin{bmatrix} 2\left(h_2^2 \dfrac{y_2 - y_1}{h_1} - h_1^2 \dfrac{y_3 - y_2}{h_2}\right) \\ 3\left(h_2 \frac{y_2 - y_1}{h_1} + h_1 \frac{y_3 - y_2}{h_2}\right) \\ \vdots \\ 3\left(h_{n-1} \frac{y_{n-1} - y_{n-2}}{h_{n-2}} + h_{n-2} \frac{y_n - y_{n-1}}{h_{n-1}}\right) \\ 2\left(h_{n-1}^2 \dfrac{y_{n-1} - y_{n-2}}{h_{n-2}} - h_{n-2}^2 \dfrac{y_n - y_{n-1}}{h_{n-1}}\right) \end{bmatrix}. \tag{6.92}$$

Se reduzirmos esse sistema pela eliminação das incógnitas y_1' e y_n', o sistema resultante possui uma matriz de coeficientes diagonal dominante estrita, portanto, a solução é única.

O termo *not-a-knot* (não nó) relaciona-se à nomenclatura dos splines. O termo *nó* é utilizado para os pontos interpolados. Neles, a derivada terceira da função spline é descontínua, portanto, quando impomos a continuidade dessa derivada em x_2 e x_{n-1} é como se esses pontos deixassem de ser nós.

6.6.4 Spline periódico

Se o conjunto de n pontos da interpolação $\mathcal{I}$ for tal que $y_1 = y_n$, então é possível construir o spline periódico, definido com um spline cúbico que satisfaz as seguintes condições de periodicidade

$$s_1'(x_1) = s_{n-1}'(x_n) \quad \text{e} \quad s_1''(x_1) = s_{n-1}''(x_n). \tag{6.93}$$

Em termos dos coeficientes (6.70)

$$b_1 = b_{n-1} \quad \text{e} \quad 2c_1 = 2c_{n-1} + 6d_{n-1}h_{n-1}, \tag{6.94}$$

ou seja,

$$\begin{cases} y_1' - y_n' = 0 \\ \\ 2h_{n-1}y_1' + h_{n-1}y_2' + h_1 y_{n-1}' + 2h_1 y_n' = 3\left(h_{n-1}\dfrac{y_2 - y_1}{h_1} + h_1\dfrac{y_n - y_{n-1}}{h_{n-1}}\right) \end{cases}. \tag{6.95}$$

Essas duas equações agregadas às equações (6.75) formam um sistema de n equações $Ay' = z$, onde

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & -1 \\ h_2 & 2(h_1+h_2) & h_1 & 0 & \cdots & 0 & 0 \\ 0 & h_3 & 2(h_2+h_3) & h_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & h_{n-1} & 2(h_{n-2}+h_{n-1}) & h_{n-2} \\ 2h_{n-1} & h_{n-1} & 0 & \cdots & 0 & h_1 & 2h_1 \end{bmatrix}, \tag{6.96}$$

$$y' = \begin{bmatrix} y_1' \\ y_2' \\ \vdots \\ y_n' \end{bmatrix} \quad \text{e} \quad z = 3\begin{bmatrix} 0 \\ h_2\frac{y_2-y_1}{h_1} + h_1\frac{y_3-y_2}{h_2} \\ \vdots \\ h_{n-1}\frac{y_{n-1}-y_{n-2}}{h_{n-2}} + h_{n-2}\frac{y_n-y_{n-1}}{h_{n-1}} \\ h_{n-1}\frac{y_2-y_1}{h_1} + h_1\frac{y_n-y_{n-1}}{h_{n-1}} \end{bmatrix}. \tag{6.97}$$

Neste caso também, se reduzirmos esse sistema pela eliminação das incógnitas y_1' e y_n', o sistema resultante possui uma matriz de coeficientes diagonal dominante estrita, portanto, a solução é única.

Capítulo 7

Ajuste de curvas

Neste capítulo, abordamos os problemas de **ajuste de curvas** pelo **método dos mínimos quadrados**. Mais precisamente, dado um conjunto de N pontos $\{(x_j, y_j) \in \mathbb{R}^2\}_{j=1}^{N}$ e uma família de funções $\mathcal{F} = \{f : \mathbb{R} \to \mathbb{R}; y = f(x)\}$, o problema de ajuste de curvas consiste em encontrar uma função da família $\mathcal{F}$ que melhor se ajusta aos pontos dados, não necessariamente que os interpola.

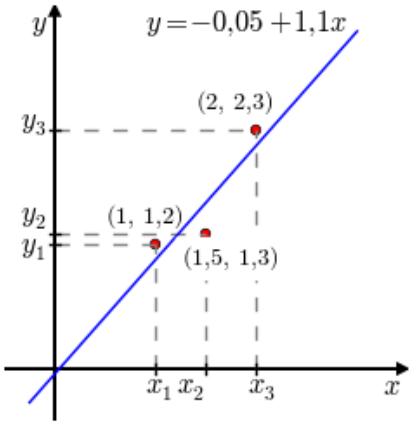


Figura 7.1: Exemplo de um problema de ajuste de uma reta entre três pontos, veja o Exemplo 7.0.1.

Aqui, o termo "melhor se ajusta" é entendido no sentido de mínimos quadrados, isto é, buscamos encontrar uma função $f \in \mathcal{F}$ tal que $f(x)$ resolve o seguinte problema de minimização

$$\min_{f \in \mathcal{F}} \sum_{j=1}^{N} \left(f(x_j) - y_j\right)^2, \tag{7.1}$$

ou seja, $f(x)$ é a função da família $\mathcal{F}$ cujo erro quadrático entre y_j e $f(x_j)$, $j = 1, 2, \ldots, N$, é mínimo. A expressão

$$\begin{aligned} R &:= \sum_{j=1}^{N} (f(x_j) - y_j)^2 \\ &= (f(x_1) - y_1)^2 + (f(x_2) - y_2)^2 + \cdots + (f(x_N) - y_N)^2 \end{aligned} \tag{7.2}$$

é chamada de **resíduo** e consiste na soma dos quadrados das diferenças entre a ordenadas y_j e o valor da função procurada $f(x_j)$.

Exemplo 7.0.1. Dado o conjunto de pontos $\{(1, 1{,}2), (1{,}5, 1{,}3), (2, 2{,}3)\}$ e a família de retas $f(x) = a + bx$, podemos mostrar que $f(x) = -0{,}05 + 1{,}1x$ é a reta que melhor aproxima os pontos dados no sentido de mínimos quadrados. Os pontos e a reta ajustada e são esboçados na Figura 7.1.

Na sequência, discutimos o procedimento de ajuste de uma reta, então, mostramos a generalização da técnica para problemas lineares de ajuste e, por fim, discutimos alguns problemas de ajuste não lineares.

Ao longo deste capítulo, assumiremos que as seguintes bibliotecas e módulos `Python` estão carregadas:

```
>>> from __future__ import division
>>> import numpy as np
>>> from numpy import linalg
>>> import matplotlib.pyplot as plt
```

7.1 Ajuste de uma reta

Nesta seção, discutiremos o procedimento de ajuste de uma reta a um conjunto de pontos dados. Em outras palavras, discutiremos o método de solução para o problema de encontrar o polinômio do primeiro grau que melhor se aproxima a um dado conjunto de pontos pelo método dos mínimos quadrados.

Seja, então, $\{(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)\}$ um conjunto de N pontos dados. Buscamos encontrar a função $f(x) = a_1 + a_2 x$ tal que o resíduo

$$R = \sum_{j=1}^{N} (f(x_j) - y_j)^2 \tag{7.3}$$

seja mínimo.

Para tal, primeiro observamos que $f(x_j) = a_1 + a_2 x_j$ e, portanto, o resíduo pode ser escrito explicitamente como uma função de a_1 e a_2 conforme a seguinte

expressão:

$$R(a_1,a_2) = \sum_{j=1}^{N}(a_1 + a_2x_j - y_j)^2. \tag{7.4}$$

Observamos que $R(a_1,a_2)$ é uma forma quadrática e que seu mínimo ocorre quando suas derivadas parciais primeiras são iguais a zero, isto é,

$$\frac{\partial R}{\partial a_1} = \frac{\partial}{\partial a_1}\sum_{j=1}^{N}(a_1 + a_2x_j - y_j)^2 = 0, \tag{7.5}$$

$$\frac{\partial R}{\partial a_2} = \frac{\partial}{\partial a_2}\sum_{j=1}^{N}(a_1 + a_2x_j - y_j)^2 = 0. \tag{7.6}$$

Ou seja,

$$2\sum_{j=1}^{N}(a_1 + a_2x_j - y_j)\cdot 1 = 0, \tag{7.7}$$

$$2\sum_{j=1}^{N}(a_1 + a_2x_j - y_j)\cdot x_j = 0, \tag{7.8}$$

e isolando as incógnitas temos

$$a_1\sum_{j=1}^{N}1 + a_2\sum_{j=1}^{N}x_j = \sum_{j=1}^{N}y_j, \tag{7.9}$$

$$a_1\sum_{j=1}^{N}x_j + a_2\sum_{j=1}^{N}x_j^2 = \sum_{j=1}^{N}y_jx_j. \tag{7.10}$$

Observando que $\sum_{j=1}^{N}1 = N$, o sistema linear acima pode ser escrito na forma matricial $Ma = w$, isto é,

$$\underbrace{\begin{bmatrix} N & \sum_{j=1}^{N}x_j \\ \sum_{j=1}^{N}x_j & \sum_{j=1}^{N}x_j^2 \end{bmatrix}}_{M}\underbrace{\begin{bmatrix} a_1 \\ a_2 \end{bmatrix}}_{a} = \underbrace{\begin{bmatrix} \sum_{j=1}^{N}y_j \\ \sum_{j=1}^{N}x_jy_j \end{bmatrix}}_{w}. \tag{7.11}$$

Este sistema linear de duas equações e duas incógnitas admite uma única solução quando o determinante da matriz dos coeficientes for não nulo, isto é,

$$N\sum_{j=1}^{N}x_j^2 - \left(\sum_{j=1}^{N}x_j\right)^2 \neq 0 \tag{7.12}$$

Pode-se mostrar usando a **desigualdade de Cauchy–Schwarz** que isto acontece quando existem pelo menos duas abscissas diferentes envolvidas no ajuste. Usando a fórmula da inversa de uma matriz dois-por-dois, chegamos às seguintes fórmulas para os coeficientes a_1 e a_2:

$$\begin{aligned} a_1 &= \frac{\sum_{j=1}^N x_j^2 \cdot \sum_{j=1}^N y_j - \sum_{j=1}^N x_j \cdot \sum_{j=1}^N x_j y_j}{N \sum_{j=1}^N x_j^2 - \left(\sum_{j=1}^N x_j\right)^2} \\ a_2 &= \frac{N \sum_{j=1}^N x_j y_j - \sum_{j=1}^N x_j \cdot \sum_{j=1}^N y_j}{N \sum_{j=1}^N x_j^2 - \left(\sum_{j=1}^N x_j\right)^2} \end{aligned} \tag{7.13}$$

Por fim, observamos que o sistema $Ma = w$ descrito na Equação (7.11) pode ser reescrito na forma $V^T V a = V^T y$, onde $V := [1\ x]$ é a matriz dos coeficientes do seguinte sistema linear sobre determinado:

$$\begin{aligned} a_1 + a_2 x_1 &= y_1 \\ a_1 + a_2 x_2 &= y_2 \\ &\vdots \\ a_1 + a_2 x_N &= y_N \end{aligned} \tag{7.14}$$

Se os pontos dados não são colineares, este sistema não tem solução. Mas, sempre que pelo menos duas abscissas foram diferentes, $M = V^T V$ é uma matriz invertível e (veja o Exercício 7.1.1), então

$$a = \left(V^T V\right)^{-1} V^T y, \tag{7.15}$$

nos fornece a chamada solução por mínimos quadrados do sistema (7.14). Note que esta é uma forma de obter os coeficientes $a = (a_1, a_2)$ equivalente àquela dada em (7.13).

Exemplo 7.1.1. Retornemos ao Exemplo 7.0.1. Isto é, dado o conjunto de pontos $\{(1,1{,}2), (1{,}5, 1{,}3), (2, 2{,}3)\}$, encontrar a função do tipo $f(x) = a_1 + a_2 x$ que melhor se ajusta os pontos dados no sentido de mínimos quadrados.

Solução. Usando as fórmulas em (7.13), obtemos

$$a_1 = \frac{7{,}25 \cdot 4{,}8 - 4{,}5 \cdot 7{,}75}{3 \cdot 7{,}25 - 20{,}25} = -0{,}05, \tag{7.16}$$

$$a_2 = \frac{3 \cdot 7{,}75 - 4{,}5 \cdot 4{,}8}{3 \cdot 7{,}25 - 20{,}25} = 1{,}1. \tag{7.17}$$

Ou seja, verificamos que, de fato, a função $f(x) = -0{,}05 + 1{,}1x$ corresponde à reta que melhor ajusta os pontos dados no sentido de mínimos quadrados. Os pontos e a reta ajustada estão esboçados na Figura 7.1.

Deixamos ao leitor a verificação de que os coeficientes a_1 e a_2 também podem ser obtidos pela expressão (7.15).

Em `Python`, podemos computar os coeficientes a_1 e a_2 da seguinte forma:

```
>>> xi = np.array([1, 1.5, 2])
>>> yi = np.array([1.2,1.3,2.3])
>>> V = np.array([xi**1,xi**0]).transpose();V
array([[ 1. ,  1. ],
       [ 1.5,  1. ],
       [ 2. ,  1. ]])
>>> a = ((np.linalg.inv((V.transpose()).dot(V))).dot(V.transpose())).dot(yi);a
array([ 1.1 , -0.05])
```

Então, o gráfico da função ajustada e dos pontos pode ser obtido com os comandos:

```
>>> xx = np.linspace(0.5,2.5)
>>> plt.plot(xi,yi,'ro',xx,np.polyval(a,xx),'b-')
>>> plt.grid();plt.show()
```

$\Diamond$

O procedimento apresentado de ajuste de uma reta por mínimos quadrados pode ser generalizado para qualquer família de funções que seja um espaço vetorial de dimensão finita. Problemas de ajuste com tais famílias de funções é o que chamamos de problemas de ajuste linear, os quais exploramos em detalhe na próxima seção.

Exercício resolvido

ER 7.1.1. a) Mostre que o sistema linear $Ma = w$ descrito na Equação 7.11 pode ser reescrito na forma $V^TVa = V^Ty$, onde $V = [1\ x]$.

b) Mostre que V, como definido no item a), tem posto igual a 2 quando pelo menos duas abscissas do conjunto de pontos $\{(x_j, y_j)\}_{j=1}^N$ são diferentes. E, portanto, $M = V^TV$ é uma matriz invertível.

Solução. a) Basta observar que

$$V^TV = \begin{bmatrix} 1 & 1 \cdots & 1 & \\ x_1 & x_2 & \cdots & x_N \end{bmatrix} \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_N \end{bmatrix} = \begin{bmatrix} N & \sum_{j=1}^N x_j \\ \sum_{j=1}^N x_j & \sum_{j=1}^N x_j^2 \end{bmatrix} = M \qquad (7.18)$$

e

$$V^T y = \begin{bmatrix} 1 & 1 \cdots & 1 & \\ x_1 & x_2 & \cdots & x_N \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{bmatrix} = \begin{bmatrix} \sum_{j=1}^{N} y_j \\ \sum_{j=1}^{N} x_j y_j \end{bmatrix} = w. \tag{7.19}$$

b) Sejam $x_i \neq x_j$ duas abscissas diferentes. Então, a i-ésima e j-ésima linhas na matriz V são linearmente independentes e, portanto, o posto de V é igual a 2. Por fim, $V^T V$ é não singular, pois, se u é tal que $V^T V u = 0$, então

$$0 = u^T V^T V u = (Vu)^T (Vu) = (Vu) \cdot (Vu) \Rightarrow Vu = 0. \tag{7.20}$$

Agora, $Vu = 0$ é uma combinação linear das linhas de V igual a zero, logo $u = 0$, pois as linhas de V são linearmente independentes como mostrado antes. Concluímos que se $V^T V u = 0$, então $u = 0$, isto é, $V^T V$ é não singular. ♢

Exercícios

E 7.1.1. Sejam dados o conjunto de pontos $\{(0{,}23, -0{,}54), (-0{,}30, -0{,}54), (0{,}04, -0{,}57)\}$. Encontre a função $f(x) = a_1 + a_2 x$ que melhor se ajusta no sentido de mínimos quadrados aos pontos dados. Faça, então, um gráfico com os pontos e o esboço da função ajustada.

E 7.1.2. Seja dado o conjunto de pontos $\{(-0{,}35, 0{,}2), (0{,}15, -0{,}5), (0{,}23, 0{,}54), (0{,}35, 0{,}7)\}$. Encontre a função $f(x) = a_1 + a_2 x$ que melhor se ajusta no sentido de mínimos quadrados aos pontos dados. Faça, então, um gráfico com os pontos e o esboço da função ajustada.

E 7.1.3. Seja dado o conjunto de pontos $\{(-1{,}94, 1{,}02), (-1{,}44, 0{,}59), (0{,}93, -0{,}28), (1{,}39, -1{,}04)\}$. Encontre a função $f(x) = a_1 + a_2 x$ que melhor se ajusta no sentido de mínimos quadrados aos pontos dados. Então, responda cada item:

a) Encontre o valor de $f(1)$.

b) Encontre o valor de $f(0{,}93)$.

c) Encontre o valor de $|f(0{,}93) - (-0{,}28)|$.

d) Encontre o valor do resíduo $R = \sum_{j=1}^{N} (f(x_j) - y_j)^2$.

Forneça os valores calculados com 7 dígitos significativo por arredondamento.

7.2 Ajuste linear geral

O problema geral de ajuste linear consiste em dada uma família $\mathcal{F}$ gerada pelo conjunto de m funções $\{f_1(x), f_2(x), \ldots, f_m(x)\}$ e um conjunto de n pares ordenados $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, calcular os coeficientes $a_1, a_2, \ldots, a_m$ tais que a função dada por

$$f(x) = \sum_{j=1}^{m} a_j f_j(x) = a_1 f_1(x) + a_2 f_2(x) + \ldots + a_m f_m(x) \tag{7.21}$$

minimiza o resíduo

$$R = \sum_{i=1}^{n} \left[f(x_i) - y_i \right]^2 . \tag{7.22}$$

Aqui, a minimização é feita por todas as possíveis escolhas dos coeficientes $a_1, a_2, \ldots, a_m$.

Com o objetivo de tornar a desenvolvimento mais claro, vamos escrever R como a soma dos resíduos parciais:

$$R = \sum_{i=1}^{n} R_i, \quad \text{onde} \quad R_i := \left[f(x_i) - y_i \right]^2 . \tag{7.23}$$

Do fato que $f(x_i) = \sum_{j=1}^{m} a_j f_j(x_i)$, temos que cada resíduo pode ser escrito como

$$R_i = \left[\sum_{j=1}^{m} a_j f_j(x_i) - y_i \right]^2 . \tag{7.24}$$

A fim de encontrar o ponto de mínimo, resolvemos o sistema oriundo de igualar a zero cada uma das derivadas parciais de R em relação aos m coeficientes a_j, isto é, devemos resolver:

$$\frac{\partial R}{\partial a_1} = 2 \sum_{i=1}^{n} \frac{\partial R_i}{\partial a_1} = 2 \sum_{i=1}^{n} \left[\sum_{j=1}^{m} a_j f_j(x_i) - y_i \right] f_1(x_i) = 0, \tag{7.25}$$

$$\frac{\partial R}{\partial a_2} = 2 \sum_{i=1}^{n} \frac{\partial R_i}{\partial a_2} = 2 \sum_{i=1}^{n} \left[\sum_{j=1}^{m} a_j f_j(x_i) - y_i \right] f_2(x_i) = 0, \tag{7.26}$$

$$\vdots \tag{7.27}$$

$$\frac{\partial R}{\partial a_m} = 2 \sum_{i=1}^{n} \frac{\partial R_i}{\partial a_m} = 2 \sum_{i=1}^{n} \left[\sum_{j=1}^{m} a_j f_j(x_i) - y_i \right] f_m(x_i) = 0. \tag{7.28}$$

Dividindo cada equação por 2 e escrevendo na forma matricial, obtemos $Ma = w$, onde a matriz M é dada por:

$$M = \begin{bmatrix} \sum_{i=1}^{n} f_1(x_i)^2 & \sum_{i=1}^{n} f_2(x_i)f_1(x_i) & \cdots & \sum_{i=1}^{n} f_m(x_i)f_1(x_i) \\ \sum_{i=1}^{n} f_1(x_i)f_2(x_i) & \sum_{i=1}^{n} f_2(x_i)^2 & \cdots & \sum_{i=1}^{n} f_m(x_i)f_2(x_i) \\ \sum_{i=1}^{n} f_1(x_i)f_3(x_i) & \sum_{i=1}^{n} f_2(x_i)f_3(x_i) & \cdots & \sum_{i=1}^{n} f_m(x_i)f_3(x_i) \\ \vdots & \vdots & \ddots & \vdots \\ \sum_{i=1}^{n} f_1(x_i)f_m(x_i) & \sum_{i=1}^{n} f_2(x_i)f_m(x_i) & \cdots & \sum_{i=1}^{n} f_m(x_i)^2 \end{bmatrix}. \tag{7.29}$$

E os vetores a e w são dados por:

$$a = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} \quad \text{e} \quad w = \begin{bmatrix} \sum_{i=1}^{n} f_1(x_i)y_i \\ \sum_{i=1}^{n} f_2(x_i)y_i \\ \sum_{i=1}^{n} f_3(x_i)y_i \\ \vdots \\ \sum_{i=1}^{n} f_m(x_i)y_i \end{bmatrix}. \tag{7.30}$$

Agora, observamos que $M = V^T V$ e $w = V^T y$, onde a matriz V é dada por:

$$V = \begin{bmatrix} f_1(x_1) & f_2(x_1) & \cdots & f_m(x_1) \\ f_1(x_2) & f_2(x_2) & \cdots & f_m(x_2) \\ f_1(x_3) & f_2(x_3) & \cdots & f_m(x_3) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_n) & f_2(x_n) & \cdots & f_m(x_n) \end{bmatrix} \tag{7.31}$$

e y é o vetor coluna $y = (y_1, y_2, \ldots, y_N)$.

Assim, o problema de ajuste se reduz a resolver o sistema linear $Ma = w$, ou $V^T V a = V^T y$. Este sistema linear tem solução única se a matriz M for inversível. O teorema a seguir mostra que isto acontece sempre a matriz V possui posto m, ou seja, o número de linhas linearmente independentes for igual ao número de colunas.[1]

[1] Nota-se que o posto não pode ultrapassar o número de colunas.

Teorema 7.2.1. *A matriz $M = V^TV$ é quadrada de ordem m e é inversível sempre que o posto da matriz V é igual a número de colunas m.*

Demonstração. Para provar que M é inversível, precisamos mostrar que se v é um vetor de ordem m e $Mv = 0$, então $v = 0$. Suponha, então, que $Mv = 0$, isto é, $V^TVv = 0$. Tomando o produto interno da expressão $V^TVv = 0$ com v, temos:

$$0 = \left\langle V^TVv,v \right\rangle = \langle Vv,Vv \rangle = \|Vv\|^2 \tag{7.32}$$

Portanto $Mv = 0$ implica obrigatoriamente $Vv = 0$. Como o posto de V é igual ao número de colunas, v precisar ser o vetor nulo. □

Observação 7.2.1. Este problema é equivalente a resolver pelo métodos dos mínimos quadrados o seguinte sistema linear:

$$\begin{bmatrix} f_1(x_1) & f_2(x_1) & \cdots & f_m(x_1) \\ f_1(x_2) & f_2(x_2) & \cdots & f_m(x_2) \\ f_1(x_3) & f_2(x_3) & \cdots & f_m(x_3) \\ \vdots & \vdots & \ddots & \vdots \\ f_1(x_n) & f_2(x_n) & \cdots & f_m(x_n) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} \tag{7.33}$$

Observação 7.2.2. O caso de ajuste de um reta para um conjunto de pontos é um caso particular de ajuste linear.

Exemplo 7.2.1. Encontre a reta que melhor se ajusta aos pontos dados na seguinte tabela:

i	1	2	3	4	5
x_i	0,01	1,02	2,04	2,95	3,55
y_i	1,99	4,55	7,20	9,51	10,82

Solução. O problema consiste em ajustar uma função da forma $f(x) = a_1 + a_2x$ no conjunto de pontos dados. Notamos que $f(x)$ é uma função da família gerada pelo conjunto de funções $\{f_1(x) = 1, f_2(x) = x\}$. Então, aplicando o procedimento acima, temos que o vetor dos coeficientes $a = (a_1, a_2)$ é solução por mínimos

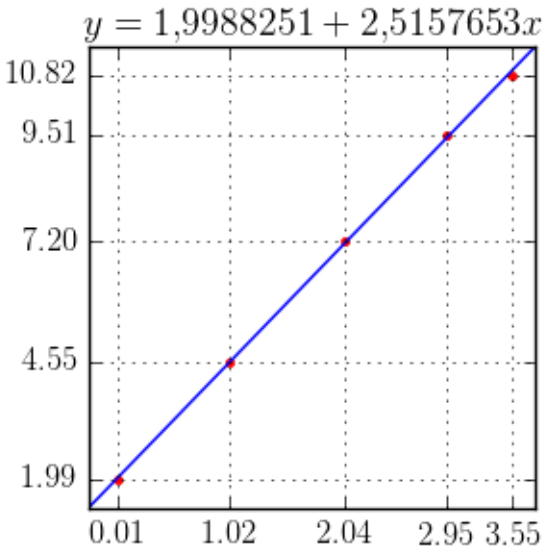


Figura 7.2: Gráfico da solução do problema apresentado no Exemplo 7.2.1.

quadrados do sistema linear $Va = y$, onde:

$$V = \begin{bmatrix} f_1(x_1) & f_2(x_1) \\ f_1(x_2) & f_2(x_2) \\ f_1(x_3) & f_2(x_3) \\ f_1(x_4) & f_2(x_4) \\ f_1(x_5) & f_2(x_5) \end{bmatrix} = \begin{bmatrix} 1 & 0{,}01 \\ 1 & 1{,}02 \\ 1 & 2{,}04 \\ 1 & 2{,}95 \\ 1 & 3{,}55 \end{bmatrix}. \tag{7.34}$$

Ou seja, é a solução do sistema $V^T Va = V^T y$ dado por

$$\begin{bmatrix} 5 & 9{,}57 \\ 9{,}57 & 26{,}5071 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} 34{,}07 \\ 85{,}8144 \end{bmatrix} \tag{7.35}$$

A solução desse sistema é $a_1 = 1{,}9988251$ e $a_2 = 2{,}5157653$. A Figura 7.2, apresenta um gráfico dos pontos e da reta ajustada.

$\Diamond$

Exemplo 7.2.2. Encontre a função $f(x) = a_1 \operatorname{sen}(\pi x) + a_2 \cos(\pi x)$ que melhor se ajusta pelo critérios dos mínimos quadrados aos seguintes pontos dados

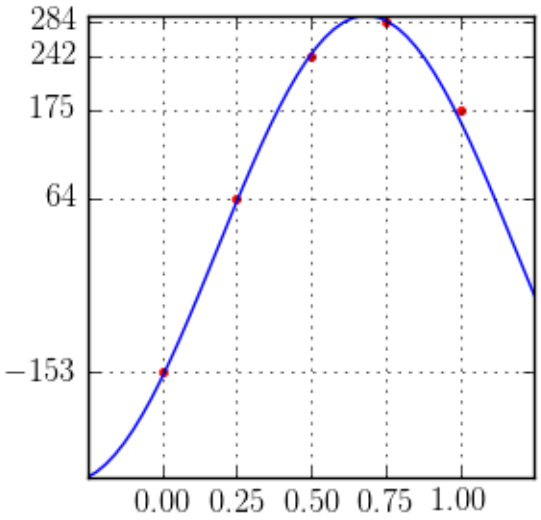


Figura 7.3: Gráfico da solução do problema apresentado no Exemplo 7.2.2.

i	1	2	3	4	5
x_i	0,00	0,25	0,50	0,75	1,00
y_i	−153	64	242	284	175

Solução. Pelo procedimento visto nesta seção, temos que os coeficientes a_1 e a_2 são dados pela solução por mínimos quadrados do seguinte sistema linear $Va = y$

$$\begin{aligned} a_1 \operatorname{sen}(\pi x_1) + a_2 \cos(\pi x_1) &= y_1 \\ a_1 \operatorname{sen}(\pi x_2) + a_2 \cos(\pi x_2) &= y_2 \\ a_1 \operatorname{sen}(\pi x_3) + a_2 \cos(\pi x_3) &= y_3 \\ a_1 \operatorname{sen}(\pi x_4) + a_2 \cos(\pi x_4) &= y_4 \\ a_1 \operatorname{sen}(\pi x_5) + a_2 \cos(\pi x_5) &= y_5 \end{aligned} \tag{7.36}$$

cuja matriz de coeficientes V é:

$$V = \begin{bmatrix} \operatorname{sen}(0) & \cos(0) \\ \operatorname{sen}(0{,}25\pi) & \cos(0{,}25\pi) \\ \operatorname{sen}(0{,}5\pi) & \cos(0{,}5\pi) \\ \operatorname{sen}(0{,}75\pi) & \cos(0{,}75\pi) \\ \operatorname{sen}(\pi) & \cos(\pi) \end{bmatrix} \tag{7.37}$$

Então, a solução por mínimos quadrados é

$$a = (V^T V)^{-1} V^T y = \begin{bmatrix} 244{,}03658 \\ -161{,}18783 \end{bmatrix}. \tag{7.38}$$

Ou seja, $f(x) = 244{,}03658\,\text{sen}\,(\pi x) - 161{,}18783\cos(\pi x)$ é a função ajustada ao conjunto de pontos dados. A Figura 7.3 apresenta o gráfica de $f(x)$ e dos pontos dados.

Em `Python`, podemos computar os coeficientes da função $f(x)$ da seguinte forma:

```
>>> xi = np.array([0,0.25,0.5,0.75,1])
>>> yi = np.array([-153,64,242,284,175])
>>> V = np.array([np.sin(np.pi*xi),np.cos(np.pi*xi)]).transpose()
>>> a = ((np.linalg.inv((V.transpose()).dot(V))).dot(V.transpose())).dot(yi)
```

$\Diamond$

Observação 7.2.3. Em `Python`, quando resolvemos um sistema $Ax = b$ usando

```
>>> x = np.linalg.inv(A).dot(b)
```

estamos computando a inversa da matriz A e multiplicando por b. Dde forma mais eficiente, podemos usar a função numpy.linalg.solve, digitando:

```
>>> x = np.linalg.solve(A,b)
```

Isto requer que a matriz A seja quadrada e de posto completo. Alternativamente, para obtermos a solução por mínimos quadrados, podemos usar a função numpy.linalg.lstsq. Neste caso, digitamos:

```
>>> np.linalg.lstsq(A,b)
```

7.2.1 Ajuste polinomial

O **ajuste polinomial** é o caso particular do ajuste linear para **funções polinomiais**, isto é, funções do tipo

$$p(x) = a_1 + a_2 x + \cdots + a_m x^{m-1}. \tag{7.39}$$

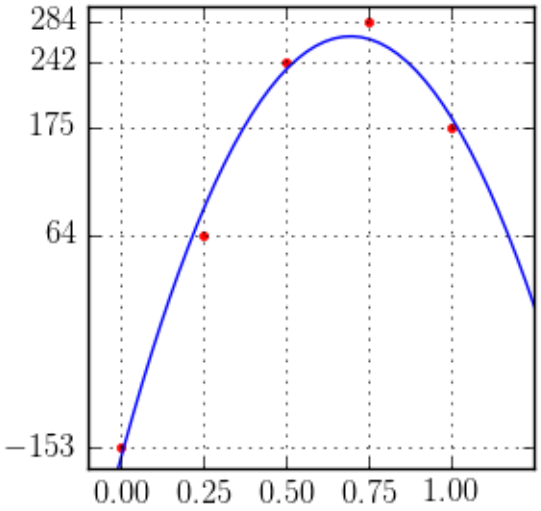


Figura 7.4: Gráfico da solução do problema apresentado no Exemplo 7.2.3.

Neste caso, a matriz V associada ao ajuste dos pontos $\{(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots, (x_n, y_n)\}$ é dada por:

$$V = \begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{m-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{m-1} \\ 1 & x_3 & x_3^2 & \cdots & x_3^{m-1} \\ \vdots & \vdots & & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{m-1} \end{bmatrix} \tag{7.40}$$

Então, os coeficientes a_i, $i = 1, 2, \ldots, m$, são dados pela solução do sistema linear $V^T V a = v^T y$:

$$\underbrace{\begin{bmatrix} n & \sum_{j=1}^{n} x_j & \cdots & \sum_{j=1}^{n} x_j^{m-1} \\ \sum_{j=1}^{n} x_j & \sum_{j=1}^{n} x_j^2 & & \sum_{j=1}^{n} x_j^m \\ \vdots & & \ddots & \vdots \\ \sum_{j=1}^{n} x_j^{m-1} & \sum_{j=1}^{n} x_j^m & \cdots & \sum_{j=1}^{n} x_j^{2m-1} \end{bmatrix}}_{V^T V} \underbrace{\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_{p+1} \end{bmatrix}}_{a} = \underbrace{\begin{bmatrix} \sum_{j=1}^{n} y_j \\ \sum_{j=1}^{n} x_j y_j \\ \vdots \\ \sum_{j=1}^{n} x_j^{m-1} y_j \end{bmatrix}}_{V^T y} \tag{7.41}$$

Exemplo 7.2.3. Entre o polinômio de grau 2 que melhor se ajusta aos pontos dados na seguinte tabela:

i	1	2	3	4	5
x_i	0,00	0,25	0,50	0,75	1,00
y_i	-153	64	242	284	175

Solução. Um polinômio de grau 2 pode ser escrito na seguinte forma:

$$p(x) = a_1 + a_2 x + a_3 x^2. \tag{7.42}$$

Assim, o problema se resume em encontrarmos a solução por mínimos quadrados do seguinte sistema linear:

$$\begin{aligned} a_1 + a_2 x_1 + a_3 x_1^2 &= y_1 \\ a_2 + a_2 x_2 + a_3 x_2^2 &= y_2 \\ a_3 + a_2 x_3 + a_3 x_3^2 &= y_3 \\ a_4 + a_2 x_4 + a_3 x_4^2 &= y_4 \\ a_5 + a_2 x_5 + a_3 x_5^2 &= y_5 \end{aligned} \tag{7.43}$$

Ou, escrita na forma matricial, $Va = y$, onde:

$$V = \begin{bmatrix} 1 & x_1 & x_1^2 \\ 1 & x_2 & x_2^2 \\ 1 & x_3 & x_3^2 \\ 1 & x_4 & x_4^2 \\ 1 & x_5 & x_5^2 \end{bmatrix} \tag{7.44}$$

A solução por mínimos quadrados é, então:

$$a = (V^T V)^{-1} V^T y = \begin{bmatrix} -165{,}37143 \\ 1250{,}9714 \\ -900{,}57143 \end{bmatrix} \tag{7.45}$$

Ou seja, o polinômio de grau 2 que melhor ajusta os pontos dados no sentido de mínimos quadrados é $p(x) = -165{,}37143 + 1250{,}9714x - 900{,}57143x^2$. A Figura 7.4 mostra o gráfico do polinômio ajustado e os pontos dados.

Em `Python`, podemos computar os coeficientes do polinômio $p(x)$ da seguinte forma:

```
>>> xi = np.array([0,0.25,0.5,0.75,1])
>>> yi = np.array([-153,64,242,284,175])
>>> V = np.array([xi**2,xi**1,xi**0]).transpose()
>>> a = ((np.linalg.inv((V.transpose()).dot(V))).dot(V.transpose())).dot(yi)
```

Para fazermos o gráfico do polinômio e dos pontos, digitamos:

```
>>> xx = np.linspace(-0.25,1.25)
>>> plt.plot(xi,yi,'ro',xx,np.polyval(a,xx),'b-')
>>> plt.grid();plt.show()
```

$\Diamond$

Exercícios

E 7.2.1. Encontre o polinômio $p(x) = a_1 + a_2x + a_3x^2$ que melhor se ajusta no sentido de mínimos quadrados aos pontos:

i	1	2	3	4
x_i	$-1{,}50$	$-0{,}50$	1,25	1,50
y_i	1,15	$-0{,}37$	0,17	0,94

E 7.2.2. Encontrar a parábola $y = ax^2+bx+c$ que melhor aproxima o seguinte conjunto de dados:

i	1	2	3	4	5
x_i	0,01	1,02	2,04	2,95	3,55
y_i	1,99	4,55	7,20	9,51	10,82

E 7.2.3. Dado o seguinte conjunto de dados

x_i	0,0	0,1	0,2	0,3	0,4	0,5	0,6	0,7	0,8	0,9	1,0
y_i	31	35	37	33	28	20	16	15	18	23	31

a) Encontre a função do tipo $f(x) = a + b\,\mathrm{sen}\,(2\pi x) + c\cos(2\pi x)$ que melhor aproxima os valores dados.

b) Encontre a função do tipo $f(x) = a + bx + cx^2 + dx^3$ que melhor aproxima os valores dados.

7.3 Aproximando problemas não lineares por problemas lineares

Eventualmente, problemas de ajuste de curvas podem recair em um sistema não linear. Por exemplo, para ajustar função $y = Ae^{bx}$ ao conjunto de pontos (x_1,y_1), (x_2,y_2) e (x_3,y_3), temos que minimizar o resíduo[2]

$$R = (Ae^{x_1 b} - y_1)^2 + (Ae^{x_2 b} - y_2)^2 + (Ae^{x_3 b} - y_3)^2 \tag{7.46}$$

ou seja, resolver o sistema

$$\begin{aligned} \frac{\partial R}{\partial A} &= 2(Ae^{x_1 b} - y_1)e^{x_1 b} + 2(Ae^{x_2 b} - y_2)e^{x_2 b} + 2(Ae^{x_3 b} - y_3)e^{x_3 b} = 0 && (7.47)\\ \frac{\partial R}{\partial b} &= 2Ax_1(Ae^{x_1 b} - y_1)e^{x_1 b} + 2Ax_2(Ae^{x_2 b} - y_2)e^{x_2 b} && (7.48)\\ &+ 2Ax_3(Ae^{x_3 b} - y_3)e^{x_3 b} = 0 && (7.49) \end{aligned}$$

que é não linear em A e b. Esse sistema pode ser resolvido pelo método de Newton-Raphson, o que pode se tornar custoso, ou mesmo inviável quando não dispomos de uma boa aproximação da solução para inicializar o método.

Felizmente, algumas famílias de curvas admitem uma transformação que nos leva a um problema linear. No caso da curva $y = Ae^{bx}$, observe que $\ln y = \ln A + bx$. Assim, em vez de ajustar a curva original $y = Ae^{bx}$ a tabela de pontos, ajustamos a curva submetida a transformação logarítmica

$$\tilde{y} := a_1 + a_2\tilde{x} = \ln A + bx. \tag{7.50}$$

Usamos os pontos $(\tilde{x}_j,\tilde{y}_j) := (x_j, \ln y_j)$, $j = 1,2,3$ e resolvemos o sistema linear

$$V^T V \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = V^T \begin{bmatrix} \tilde{y}_1 \\ \tilde{y}_2 \\ \tilde{y}_3 \end{bmatrix}, \tag{7.51}$$

onde

$$V = \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \end{bmatrix}. \tag{7.52}$$

[2] A soma do quadrado dos resíduos.

Exemplo 7.3.1. Encontre uma curva da forma $y = Ae^{bx}$ que melhor ajusta os pontos $(1, 2)$, $(2, 3)$ e $(3, 5)$.

Solução. Aplicando o logaritmo natural de ambos os lados da equação $y = Ae^{bx}$, temos

$$\ln y = \ln A + bx. \tag{7.53}$$

Então, denotando $\tilde{y} := \ln y$, $a_1 := \ln A$ e $a_2 := b$, o problema reduz-se a ajustar a reta $\tilde{y} = a_1 + a_2 x$ aos pontos $(1, \ln 2)$, $(2, \ln 3)$ e $(3, \ln 5)$. Para tanto, resolvemos o sistema

$$\underbrace{\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix}}_{V} \underbrace{\begin{bmatrix} a_1 \\ a_2 \end{bmatrix}}_{a} = \underbrace{\begin{bmatrix} \ln 2 \\ \ln 3 \\ \ln 5 \end{bmatrix}}_{\tilde{y}} \tag{7.54}$$

por mínimos quadrados, isto é,

$$V^T V a = V^T \tilde{y} \Rightarrow a = \left(V^T V\right)^{-1} V^T \tilde{y}. \tag{7.55}$$

A solução do sistema é, então, $a_1 = 0{,}217442$ e $a_2 = 0{,}458145$. Portanto, $A = e^{a_1} = 1{,}24289$ e $b = a_2 = 0{,}458145$.

Em `Python`, podemos resolver este problema com o seguinte código:

```
x = np.array([1,2,3])
y = np.array([2,3,5])
V = np.array([np.ones(3),x]).transpose()
a = np.linalg.lstsq(V,np.log(y))[0]
A = np.exp(a[0])
b = a[1]
```

◊

Observação 7.3.1. Os coeficientes obtidos a partir dessa linearização são aproximados, ou seja, são diferentes daqueles obtidos quando aplicamos mínimos quadrados não linear. Observe que estamos minimizando $\sum_i [\ln y_i - \ln(f(x_i))]^2$ em vez de $\sum_i [y_i - f(x_i)]^2$. No exemplo resolvido, a solução do sistema não linear original seria $A = 1{,}19789$ e $b = 0{,}474348$

Observação 7.3.2. Mesmo quando se deseja resolver o sistema não linear, a solução do problema linearizado pode ser usada para construir condições iniciais para o problema não linear.

A próxima tabela apresenta algumas curvas e transformações que linearizam o problema de ajuste.

Curva	Transformação	Problema Linearizado
$y = ae^{bx}$	$\tilde{y} = \ln y$	$\tilde{y} = \ln a + bx$
$y = ax^b$	$\tilde{y} = \ln y$	$\tilde{y} = \ln a + b \ln x$
$y = ax^b e^{cx}$	$\tilde{y} = \ln y$	$\tilde{y} = \ln a + b \ln x + cx$
$y = ae^{(b+cx)^2}$	$\tilde{y} = \ln y$	$\tilde{y} = \ln a + b^2 + bcx + c^2x^2$
$y = \dfrac{a}{b+x}$	$\tilde{y} = \dfrac{1}{y}$	$\tilde{y} = \dfrac{b}{a} + \dfrac{1}{a}x$
$y = A\cos(\omega x + \phi)$	$-x-$	$y = a\cos(\omega x) - b\,\mathrm{sen}\,(\omega x)$
ω conhecido		$a = A\cos(\phi)$, $b = A\,\mathrm{sen}\,(\phi)$

Exemplo 7.3.2. Encontre a função f da forma $y = f(x) = A\cos(2\pi x + \phi)$ que ajusta a tabela de pontos

x_i	y_i
0,0	9,12
0,1	1,42
0,2	- 7,76
0,3	- 11,13
0,4	- 11,6
0,5	- 6,44
0,6	1,41
0,7	11,01
0,8	14,73
0,9	13,22
1,0	9,93

Solução. Usando o fato que $y = A\cos(2\pi x + \phi) = a\cos(2\pi x) - b\,\mathrm{sen}\,(2\pi x)$, onde $a = A\cos(\phi)$ e $b = A\,\mathrm{sen}\,(\phi)$, $z = \begin{bmatrix} a & b \end{bmatrix}^T$ é solução do problema

$$B^T B z = B^T y, \tag{7.56}$$

onde

$$B = \begin{bmatrix} \cos(2\pi x_0) & -\operatorname{sen}(2\pi x_0) \\ \cos(2\pi x_1) & -\operatorname{sen}(2\pi x_1) \\ \vdots & \\ \cos(2\pi x_{10}) & -\operatorname{sen}(2\pi x_{10}) \end{bmatrix} = \begin{bmatrix} 1. & 0. \\ 0{,}8090170 & -0{,}5877853 \\ 0{,}3090170 & -0{,}9510565 \\ -0{,}3090170 & -0{,}9510565 \\ -0{,}8090170 & -0{,}5877853 \\ -1{,}0000000 & 0{,}0000000 \\ -0{,}8090170 & 0{,}5877853 \\ -0{,}3090170 & 0{,}9510565 \\ 0{,}3090170 & 0{,}9510565 \\ 0{,}8090170 & 0{,}5877853 \\ 1{,}0000000 & 0{,}0000000 \end{bmatrix}. \tag{7.57}$$

Assim, $a = 7{,}9614704$ e $b = 11{,}405721$ e obtemos o seguinte sistema:

$$\begin{cases} A\cos(\phi) = 7{,}9614704 \\ A\operatorname{sen}(\phi) = 11{,}405721 \end{cases}. \tag{7.58}$$

Observe que

$$A^2 = 7{,}9614704^2 + 11{,}405721^2 \tag{7.59}$$

e, escolhendo $A > 0$, $A = 13{,}909546$ e

$$\operatorname{sen}(\phi) = \frac{11{,}405721}{13{,}909546} = 0{,}8199923 \tag{7.60}$$

Assim, como $\cos\phi$ também é positivo, ϕ é um ângulo do primeiro quadrante:

$$\phi = 0{,}9613976 \tag{7.61}$$

Portanto $f(x) = 13{,}909546\cos(2\pi x + 0{,}9613976)$. Observe que nesse exemplo a solução do problema linear é a mesma do problema não linear. ♢

Exercícios resolvidos

ER 7.3.1. Encontre a função f da forma $y = f(x) = \frac{a}{b+x}$ que ajusta a seguinte tabela de pontos usando uma das transformações tabeladas.

i	x_i	y_i
1	0,0	101
2	0,2	85
3	0,4	75
4	0,6	66
5	0,8	60
6	1,0	55

Solução. Usando o fato que $Y = \frac{1}{y} = \frac{b}{a} + \frac{1}{a}x$, $z = \left[\begin{array}{cc} \frac{b}{a} & \frac{1}{a} \end{array}\right]^T$ é solução do problema

$$A^T A z = A^T Y, \tag{7.62}$$

onde

$$A = \left[\begin{array}{cc} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ 1 & x_4 \\ 1 & x_5 \\ 1 & x_6 \end{array}\right] = \left[\begin{array}{cc} 1 & 0{,}0 \\ 1 & 0{,}2 \\ 1 & 0{,}4 \\ 1 & 0{,}6 \\ 1 & 0{,}8 \\ 1 & 1{,}0 \end{array}\right] \tag{7.63}$$

e

$$Y = \left[\begin{array}{c} 1/y_1 \\ 1/y_2 \\ 1/y_3 \\ 1/y_4 \\ 1/y_5 \\ 1/y_6 \end{array}\right] = \left[\begin{array}{c} 0{,}0099010 \\ 0{,}0117647 \\ 0{,}0133333 \\ 0{,}0151515 \\ 0{,}0166667 \\ 0{,}0181818 \end{array}\right] \tag{7.64}$$

Assim, $\frac{1}{a} = 0{,}0082755$ e $\frac{b}{a} = 0{,}0100288$ e, então, $a = 120{,}83924$ e $b = 1{,}2118696$, ou seja, $f(x) = \frac{120{,}83924}{1{,}2118696+x}$. ♢

Exercícios

Esta seção carece de exercícios. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

Capítulo 8

Derivação numérica

Nesta seção, trataremos das estratégias numéricas para aproximação de derivadas de funções reais. Com as técnicas abordadas, é possível calcular aproximadamente a derivada de uma função a partir de um conjunto discreto de pontos $\{(x_i, y_i)\}_{i=1}^n$. Começamos discutindo as chamadas **aproximações por diferenças finitas** e, então, as aproximações de derivadas via ajuste ou interpolação.

Ao longo deste capítulo, assumiremos que as seguintes bibliotecas e módulos `Python` estão importados:

```
from __future__ import division
import numpy as np
import matplotlib.pyplot as plt
```

8.1 Diferenças finitas

Uma diferença finita é uma expressão da forma $f(x+b) - f(x+a)$, que ao ser dividida por $(b-a)$ chama-se um quociente de diferenças. A técnica de **diferenças finitas** consiste em aproximar a derivada de uma função via fórmulas discretas que requerem apenas um conjunto finito de pares ordenados $\{(x_i, y_i)\}_{i=1}^n$, onde geralmente denotamos $y_i = f(x_i)$.

Essas fórmulas podem ser obtidas de várias maneiras. Começamos com a fórmula mais simples que pode ser obtida do cálculo diferencial. Seja f uma função diferenciável, a derivada de f no ponto x_0 é, por definição,

$$f'(x_0) = \lim_{h \to 0} \frac{f(x_0 + h) - f(x_0)}{h}. \tag{8.1}$$

Deste limite, tomando $h \neq 0$ pequeno (não muito pequeno para evitar o cancelamento catastrófico), é esperado que possamos obter uma aproximação razoável

para $f'(x_0)$. Assim, a **diferença finita progressiva** de ordem 1

$$D_{+,h}f(x_0) := \frac{f(x_0+h)-f(x_0)}{h} \approx f'(x_0) \tag{8.2}$$

é uma aproximação para $f'(x_0)$.

Exemplo 8.1.1. Usando a diferença finita progressiva de ordem 1, calcule aproximações da derivada de $f(x) = \cos(x)$ no ponto $x = 1$ usando $h = 10^{-1}$, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-12} e 10^{-14}. Calcule o erro $|D_{+,h}f(1) - f'(1)|$ obtido para cada valor de h.

Solução. Usando a diferença progressiva em (8.2), devemos calcular

$$D_{+,h}f(1) = \frac{\cos(1+h)-\cos(1)}{h} \tag{8.3}$$

Fazendo isso, obtemos:

h	$D_{+,h}f(1)$	$\lvert f'(1) - D_{+,h}f(1)\rvert$
10^{-1}	−8,67062E−01	2,55909E−02
10^{-2}	−8,44158E−01	2,68746E−03
10^{-3}	−8,41741E−01	2,70011E−04
10^{-4}	−8,41498E−01	2,70137E−05
10^{-12}	−8,41549E−01	7,80679E−05
10^{-14}	−8,43769E−01	2,29851E−03

Em `Python`, podemos calcular a aproximação da derivada $f'(1)$ com $h = 0{,}1$ usando as seguintes linhas de código:

```
>>> def f(x):
...     return np.cos(x)
...
>>> x0=1
>>> h=0.1
>>> df = (f(x0+h)-f(x0))/h
```

E, similarmente, para outros valores de x_0 e h. ◊

Exploremos o Exemplo 8.1.1 um pouco mais. Observamos que, para valores moderados de h, o erro $|f'(1) - D_{+,h}f(1)|$ diminui linearmente com h (veja Figura 8.1). Isto é consequência da ordem de truncamento da fórmula de diferenças finitas aplicada (que é de ordem 1). Porém, para valores muito pequenos de $h < 10^{-8}$, o erro passa a aumentar quando diminuímos h. Isto é devido ao efeito de cancelamento catastrófico.

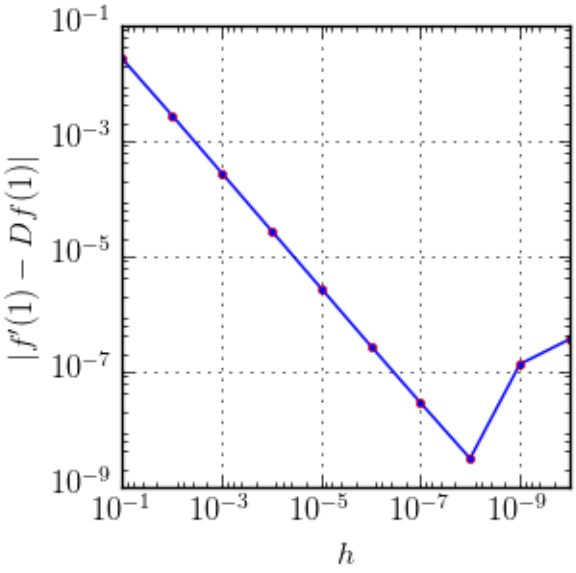


Figura 8.1: Erro absoluto das derivadas numéricas no Exemplo 8.1.1.

8.1.1 Diferenças finitas via série de Taylor

Podemos construir fórmulas de diferenças finitas para uma função $f(x)$ (suave[1]) no ponto $x = x_0$ a partir de seu polinômio de Taylor. Em alguns casos, este procedimento acaba por nos fornecer, também, a ordem de truncamento da fórmula.

Diferença finita progressiva de ordem 1

Podemos obter uma aproximação para $f'(x_0)$ a partir da série de Taylor

$$f(x_0 + h) = f(x_0) + hf'(x_0) + h^2\frac{f''(\xi)}{2}, \quad h > 0, \xi \in (x_0, x_0 + h). \tag{8.4}$$

Isolando $f'(x_0)$, obtemos

$$f'(x_0) = \underbrace{\frac{f(x_0 + h) - f(x_0)}{h}}_{D_{+,h}} - \underbrace{h\frac{f''(\xi)}{2}}_{\mathcal{O}(h)}, \tag{8.5}$$

[1]Uma função suave é uma função infinitamente continuamente diferenciável, isto é, $f \in C^\infty(\mathbb{R})$. Uma análise mais cuidadosa, revela que hipóteses mais fracas podem ser assumidas.

o que mostra que o erro de truncamento da **diferença finita progressiva**[2]

$$D_{+,h}f(x_0) := \frac{f(x_0+h)-f(x_0)}{h} \tag{8.6}$$

é de ordem h.

Diferença finita regressiva de ordem 1

Outra aproximação para a derivada primeira pode ser obtida da série de Taylor de f em torno de (x_0-h) dada por

$$f(x_0-h) = f(x_0) - hf'(x_0) + h^2\frac{f''(\xi)}{2}, \quad h>0, \xi\in(x_0,x_0+h). \tag{8.7}$$

Isolando $f'(x_0)$, obtemos

$$f'(x_0) = \underbrace{\frac{f(x_0)-f(x_0-h)}{h}}_{D_{-,h}} + \underbrace{h\frac{f''(\xi)}{2}}_{\mathcal{O}(h)}. \tag{8.8}$$

que fornece a **diferença finita regressiva**[3]

$$D_{-,h}f(x_0) := \frac{f(x_0)-f(x_0-h)}{h}, \tag{8.9}$$

que possui erro de truncamento de ordem h.

Diferença finita central de ordem 2

Para obter uma aproximação para a derivada primeira com um erro menor, podemos utilizar as séries de Taylor:

$$f(x_0+h) = f(x_0) + hf'(x_0) + h^2f''(x_0) + h^3\frac{f'''(\xi_+)}{3!}, \tag{8.10}$$

$$f(x_0-h) = f(x_0) - hf'(x_0) + h^2f''(x_0) + h^3\frac{f'''(\xi_-)}{3!} \tag{8.11}$$

Fazendo a primeira equação menos a segunda, obtemos

$$f(x_0+h)-f(x_0-h) = 2hf'(x_0) + h^3\left(\frac{f'''(\xi_+)-f'''(\xi_-)}{3!}\right). \tag{8.12}$$

[2]Também chamada de diferença finita progressiva de dois pontos ou diferença pra frente.
[3]Também chamada de diferença regressiva de dois pontos ou diferença pra trás.

Dividindo por $2h$ e isolando $f'(x_0)$ obtemos

$$f'(x_0) = \underbrace{\frac{f(x_0+h)-f(x_0-h)}{2h}}_{D_{0,h}} - \underbrace{h^2\left(\frac{f'''(\xi_+)-f'''(\xi_-)}{2\cdot 3!}\right)}_{\mathcal{O}(h^2)}. \tag{8.13}$$

Assim, a **diferença finita central**[4]

$$D_{0,h}f(x_0) := \frac{f(x_0+h)-f(x_0-h)}{2h}, \tag{8.14}$$

é uma aproximação para $f'(x_0)$ com erro de truncamento de ordem h^2, ou simplesmente ordem 2.

Exemplo 8.1.2. Calcule a derivada numérica da função $f(x) = e^{\frac{1}{2}x}$ no ponto $x = 2$ usando a diferença progressiva, diferença regressiva e diferença central com $h = 10^{-1}$, $h = 10^{-2}$ e $h = 10^{-4}$. Também, calcule o erro absoluto da aproximação obtida em cada caso.

Solução. Usando a diferença progressiva, devemos calcular

$$D_{+,h} = \frac{f(x+h)-f(x)}{h} = \frac{e^{\frac{1}{2}(x+h)}-e^{\frac{1}{2}x}}{h}. \tag{8.15}$$

Com a diferença regressiva, calculamos

$$D_{-,h} = \frac{f(x)-f(x-h)}{h} = \frac{e^{\frac{1}{2}x}-e^{\frac{1}{2}(x-h)}}{h}. \tag{8.16}$$

Por fim, usando a diferença central temos

$$D_{0,h} = \frac{f(x+h)-f(x-h)}{2h} = \frac{e^{\frac{1}{2}(x+h)}-e^{\frac{1}{2}(x-h)}}{2h}. \tag{8.17}$$

As aproximações e os erros absolutos calculados em cada caso estão apresentados na seguinte tabela:

h	$D_{+,h}f(2)$	Erro	$D_{-,h}$	Erro	$D_{0,h}$	Erro
10^{-1}	1,39369	3,5E−02	1,32572	3,3E−02	1,35971	5,7E−04
10^{-2}	1,36254	3,4E−03	1,35575	3,4E−03	1,35915	5,7E−06
10^{-4}	1,35917	3,4E−05	1,35911	3,4E−05	1,35914	5,7E−10

◊

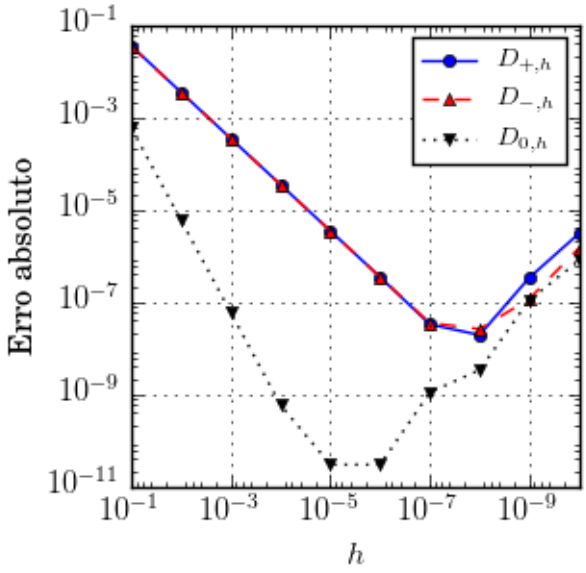


Figura 8.2: Erro absoluto das derivadas numéricas no Exemplo 8.1.2.

Observação 8.1.1. O experimento numérico realizado no Exemplo 8.1.2, nos mostra que o erro absoluto na derivação numérica não é da ordem do erro de truncamento. Entretanto, este erro tende a variar com h na mesma ordem do erro de truncamento. A Figura 8.1.2 apresenta o erro absoluto das derivadas numéricas computadas para o Exemplo 8.1.2. Note que, devido ao efeito de cancelamento catastrófico, o erro absoluto deixa de variar na ordem do erro de truncamento para valores muito pequenos de h.

Exemplo 8.1.3. Estime o erro absoluto no cálculo da derivada de $f(x) = e^{-x}$ para $x > 0$ utilizando a diferença progressiva.

Solução. Da Equação 8.5, temos:

$$f'(x) = D_{+,h}f(x) - h\frac{f''(\xi)}{2}, \quad \xi > 0, \tag{8.18}$$

ou seja:

$$|f'(x) - D_{+,h}f(x)| = \left|\frac{f''(\xi)}{2}\right| h, \quad \xi > 0. \tag{8.19}$$

Agora, como $|f''(x)| = |e^{-x}| < 1$ para $x > 0$, concluímos que:

$$|f'(x) - D_{+,h}f(x)| \leq \frac{1}{2}h, \quad x > 0. \tag{8.20}$$

[4]Também chamada de diferença finita central de três pontos. Note que o ponto $f(x_0)$ possui coeficiente 0, por isso 3 pontos.

$\Diamond$

8.1.2 Erros de arredondamento

Para entender como os erros de arredondamento se propagam ao calcular as derivadas numéricas vamos analisar a fórmula de diferenças finitas progressiva

$$D_{+,h}f(x) = \frac{f(x+h) - f(x)}{h}. \tag{8.21}$$

Nesse contexto temos o valor exato $f'(x)$ para a derivada, a sua aproximação numérica $D_{+,h}f(x)$ e a representação em número de máquina do operador $D_{+,h}f(x)$ que denotaremos por $\overline{D_{+,h}f(x)}$. Denotando por $\varepsilon(x,h)$ o erro de arredondamento ao calcularmos a derivada, vamos assumir que

$$\overline{D_{+,h}f(x)} = D_{+,h}f(x)(1+\varepsilon(x,h)) = \frac{\overline{f(x+h)} - \overline{f(x)}}{h}(1+\varepsilon(x,h)). \tag{8.22}$$

Também, consideremos

$$|\overline{f(x+h)} - f(x+h)| = \delta(x,h) \leq \delta \tag{8.23}$$

e

$$|\overline{f(x)} - f(x)| = \delta(x,0) \leq \delta, \tag{8.24}$$

onde $\overline{f(x+h)}$ e $\overline{f(x)}$ são as representações em ponto flutuante dos números $f(x+h)$ e $f(x)$, respectivamente.

Então, da Equação (8.22), a diferença do valor da derivada e sua aproximação representada em ponto flutuante pode ser estimada por:

$$\left|f'(x) - \overline{D_{+,h}f(x)}\right| = \left|f'(x) - \frac{\overline{f(x+h)} - \overline{f(x)}}{h}(1+\varepsilon(x,h))\right|. \tag{8.25}$$

Podemos reescrever o lado direito desta equação, da seguinte forma

$$\begin{aligned}
\left|f'(x) - \overline{D_{+,h}f(x)}\right| &= \left|f'(x) - \left(\frac{\overline{f(x+h)} - \overline{f(x)}}{h} + \frac{f(x+h) - f(x+h)}{h}\right.\right. && (8.26)\\
&+ \left.\left.\frac{f(x) - f(x)}{h}\right)(1+\varepsilon)\right| && (8.27)\\
&= \left|f'(x) + \left(-\frac{f(x+h) - f(x)}{h} - \frac{\overline{f(x+h)} - f(x+h)}{h}\right.\right. && (8.28)\\
&+ \left.\left.\frac{\overline{f(x)} - f(x)}{h}\right)(1+\varepsilon)\right|. && (8.29)
\end{aligned}$$

Então, separando os termos e estimando, obtemos:

$$\left|f'(x) - \overline{D_{+,h}f(x)}\right| \leq \left|f'(x) - \frac{f(x+h)-f(x)}{h}\right| + \left(\left|\frac{\overline{f(x+h)} - f(x+h)}{h}\right| \right. \tag{8.30}$$

$$\left. + \left|\frac{\overline{f(x)} - f(x)}{h}\right|\right)|1+\varepsilon| + \left|\frac{f(x+h)-f(x)}{h}\right|\varepsilon \tag{8.31}$$

$$\leq Mh + \left(\left|\frac{\delta}{h}\right| + \left|\frac{\delta}{h}\right|\right)|1+\varepsilon| + |f'(x)|\varepsilon \tag{8.32}$$

$$\leq Mh + \left(\frac{2\delta}{h}\right)|1+\varepsilon| + |f'(x)|\varepsilon \tag{8.33}$$

onde

$$M = \frac{1}{2}\max_{x\leq y\leq x+h}|f''(y)| \tag{8.34}$$

está relacionado com o erro de truncamento.

Por fim, obtemos a seguinte estimativa para o erro absoluto na computação da derivada numérica:

$$\left|f'(x) - \overline{D_{+,h}f(x)}\right| \leq Mh + \left(\frac{2\delta}{h}\right)|1+\varepsilon| + |f'(x)|\varepsilon. \tag{8.35}$$

Esta estimativa mostra que se o valor de h for muito pequeno o erro ao calcular a aproximação numérica cresce. Isso nos motiva a procurar o valor ótimo de h que minimiza o erro.

Exemplo 8.1.4. No Exemplo 8.1.2, computamos a derivada numérica da função $f(x) = e^{\frac{1}{2}x}$ no ponto $x = 2$ usando as fórmulas de diferenças finitas progressivas, regressivas e central. A Figura 8.2, mostra que, para valores h muito pequenos, os erros de arredondamento passam a dominar os cálculos e, por consequência, o erro da derivada numérica passa a aumentar. Pela figura, podemos inferir que a escolha ótima de h para as fórmulas progressiva e regressivas é $h \approx 10^{-7}$. Agora, para a fórmula central, $h \approx 10^{-5}$ parece ser a melhor escolha.

Observação 8.1.2. Note que a estimativa (8.35), mostra que o erro na computação da derivada numérica depende da função que está sendo derivada. Assim, o h ótimo depende não somente da fórmula de diferenças finitas, mas também da função a ser derivada.

Exercícios resolvidos

Esta seção carece de exercícios resolvidos. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

ER 8.1.1. Aproxime a derivada de $f(x) = \operatorname{sen}(2x) - x^2$ no ponto $x = 2$ usando a fórmula de diferenças finitas progressiva de ordem 1 com: a) $h = 0{,}1$ e b) $h = 0{,}01$. Compute, também, o erro absoluto de cada aproximação computada.

Solução. A fórmula de diferenças finitas de ordem 1 para uma função $y = f(x)$ em um ponto $x = x_0$ é dada por:

$$D_{+,h}f(x_0) = \frac{f(x_0 + h) - f(x_0)}{h}. \tag{8.36}$$

Substituindo $f(x) = \operatorname{sen}(2x) - x^2$ e $x_0 = 2$, obtemos:

$$\begin{aligned} D_{+,h}f(x_0) &= \frac{(\operatorname{sen}(2(x_0 + h)) - (x_0 + h)^2) - (\operatorname{sen}(2x_0) - x_0^2)}{h} \\ &= \frac{\operatorname{sen}(2(x_0 + h)) - x_0^2 + 2x_0h + h^2 - \operatorname{sen}(2x_0) + x_0^2)}{h} \\ &= \frac{\operatorname{sen}(4 + 2h) + 4h + h^2 - \operatorname{sen}(4))}{h}. \end{aligned} \tag{8.37}$$

Então, tomando $h = 0{,}1$, podemos computar a derivada numérica e o erro associado:

$$D_{+,0,1}f(2) = -5{,}247733, \quad |f'(2) - D_{+,0,1}f(2)| = 5{,}96 \times 10^{-2}, \tag{8.38}$$

onde $f'(x) = 2\operatorname{sen}(2x) - 2x$ é a derivada analítica. Tomando $h = 0{,}01$ temos:

$$D_{+,0,1}f(2) = -5{,}302065, \quad |f'(2) - D_{+,0,1}f(2)| = 5{,}22 \times 10^{-3}. \tag{8.39}$$

Em `Python`, podemos fazer os cálculos com o seguinte código:

```
#funcao
def f(x):
    return np.sin(2*x) - x**2

#derivada analitica
def fl(x):
    return 2*np.cos(2*x) - 2*x

#d.f. progressiva de ordem 1
def dp1(f,x,h=0.1):
    return (f(x+h)-f(x))/h

#h=0.1
dy = dp1(f,2)
```

```
print("D.F. Progressiva de ordem 1 com h = %f" % 1e-1)
print("Df = %f" % dy)
print("Erro = %1.2e" % np.abs(fl(2)-dy))

#h=0.01
dy = dp1(f,2,1e-2)
print("D.F. Progressiva de ordem 1 com h = %f" % 1e-2)
print("Df = %f" % dy)
print("Erro = %1.2e" % np.abs(fl(2)-dy))
```

$\Diamond$

Exercícios

E 8.1.1. Use os esquemas numéricos de diferença finita regressiva de ordem 1, diferença finita progressiva de ordem 1 e diferença finita central de ordem 2 para aproximar as seguintes derivadas:

a) $f'(x)$ onde $f(x) = \text{sen}\,(x)$ e $x = 2$.

b) $f'(x)$ onde $f(x) = e^{-x}$ e $x = 1$.

Use $h = 10^{-2}$ e $h = 10^{-3}$ e compare com os valores obtidos através da avaliação numérica das derivadas exatas.

E 8.1.2. Expanda a função suave $f(x)$ em um polinômio de Taylor adequado para obter as seguintes aproximações:

a) $f'(x) = \frac{f(x+h)-f(x)}{h} + \mathcal{O}(h)$

b) $f'(x) = \frac{f(x)-f(x-h)}{h} + \mathcal{O}(h)$

c) $f'(x) = \frac{f(x+h)-f(x-h)}{2h} + \mathcal{O}(h^2)$

E 8.1.3. Use a expansão da função $f(x)$ em torno de $x = 0$ em polinômios de Taylor para encontrar os coeficientes a_1, a_2 e a_3 tais que

a) $f'(0) = a_1 f(0) + a_2 f(h) + a_3 f(2h) + \mathcal{O}(h^2)$

b) $f'(0) = a_1 f(0) + a_2 f(-h) + a_3 f(-2h) + \mathcal{O}(h^2)$

c) $f'(0) = a_1 f(-h_1) + a_2 f(0) + a_3 f(h_2) + \mathcal{O}(h^2),\ \ |h_1|, |h_2| = \mathcal{O}(h)$

E 8.1.4. As tensões na entrada, v_i, e saída, v_o, de um amplificador foram medidas em regime estacionário conforme tabela abaixo.

0,0	0,50	1,00	1,50	2,00	2,50	3,00	3,50	4,00	4,50	5,00
0,0	1,05	1,83	2,69	3,83	4,56	5,49	6,56	6,11	7,06	8,29

onde a primeira linha é a tensão de entrada em volts e a segunda linha é tensão de saída em volts. Sabendo que o ganho é definido como

$$\frac{\partial v_o}{\partial v_i}. \tag{8.40}$$

Calcule o ganho quando $v_i = 1$ e $v_i = 4.5$ usando as seguintes técnicas:

a) Derivada primeira numérica de primeira ordem usando o próprio ponto e o próximo.

b) Derivada primeira numérica de primeira ordem usando o próprio ponto e o anterior.

c) Derivada primeira numérica de segunda ordem usando o ponto anterior e o próximo.

d) Derivada primeira analítica da função do tipo $v_0 = a_1 v_i + a_3 v_i^3$ que melhor se ajusta aos pontos pelo critério dos mínimos quadrados.

Caso	a	b	c	d
$v_i = 1$				
$v_i = 4.5$				

E 8.1.5. Estude o comportamento da derivada de $f(x) = e^{-x^2}$ no ponto $x = 1{,}5$ quando h fica pequeno.

8.2 Diferença finita para derivada segunda

Para aproximar a derivada segunda, considere as expansões em série de Taylor

$$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2}{2}f''(x_0) + \frac{h^3}{6}f'''(x_0) + O(h^4) \tag{8.44}$$

$$f(x_0 - h) = f(x_0) - hf'(x_0) + \frac{h^2}{2}f''(x_0) - \frac{h^3}{6}f'''(x_0) + O(h^4). \tag{8.45}$$

Somando as duas expressões, temos:

$$f(x_0 + h) + f(x_0 - h) = 2f(x_0) + h^2 f''(x_0) + O(h^4) \tag{8.46}$$

ou seja, uma aproximação de segunda ordem para a derivada segunda em x_0 é

$$f''(x_0) = \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2} + \mathcal{O}(h^2) := D^2_{0,h}f(x_0) + \mathcal{O}(h^2), \tag{8.47}$$

onde

$$D^2_{0,h}f(x_0) = \frac{f(x_0 + h) - 2f(x_0) + f(x_0 - h)}{h^2}. \tag{8.48}$$

Exemplo 8.2.1. Calcule a derivada segunda numérica de $f(x) = e^{-x^2}$ em $x = 1{,}5$ para $h = 0{,}1$, $h = 0{,}01$ e $h = 0{,}001$.

Solução. A tabela mostra os resultados:

h	$h = 0{,}1$	$h = 0{,}01$	$h = 0{,}001$
$D^2_{0,h}f(1{,}5)$	0,7364712	0,7377814	0,7377944

Observe que $f''(x) = (4x^2 - 2)e^{-x^2}$ e $f''(1{,}5) = 0{,}7377946$.

◊

Exercícios resolvidos

Esta seção carece de exercícios resolvidos. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

Exercícios

Esta seção carece de exercícios. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

E 8.2.1. Use a expansão da função $f(x)$ em torno de $x = 0$ em polinômios de Taylor para encontrar os coeficientes a_1, a_2 e a_3 tais que

a) $f''(0) = a_1 f(0) + a_2 f(h) + a_3 f(2h) + \mathcal{O}(h)$

b) $f''(0) = a_1 f(0) + a_2 f(-h) + a_3 f(-2h) + \mathcal{O}(h)$

8.3 Obtenção de fórmulas por polinômios interpoladores

Para aproximar a derivada de uma função $f(x)$ em x_0, x_1 ou x_2 usaremos os três pontos vizinhos $(x_0,f(x_0))$, $(x_1,f(x_1))$ e $(x_2,f(x_2))$. Uma interpolação usando polinômios de Lagrange para esses três pontos é da forma:

$$\begin{aligned} f(x) = f(x_0)\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)} + f(x_1)\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)} \\ + f(x_2)\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)} + \frac{f'''(\xi(x))}{6}(x-x_0)(x-x_1)(x-x_2). \end{aligned} \tag{8.49}$$

A derivada de $f(x)$ é

$$\begin{aligned} f'(x) = f(x_0)\frac{2x-x_1-x_2}{(x_0-x_1)(x_0-x_2)} + f(x_1)\frac{2x-x_0-x_2}{(x_1-x_0)(x_1-x_2)} \\ + f(x_2)\frac{2x-x_0-x_1}{(x_2-x_0)(x_2-x_1)} \\ + \frac{f'''(\xi(x))}{6}\left((x-x_1)(x-x_2)+(x-x_0)(2x-x_1-x_2)\right) \\ + D_x\left(\frac{f'''(\xi(x))}{6}\right)(x-x_0)(x-x_1)(x-x_2). \end{aligned} \tag{8.50}$$

Trocando x por x_0, temos

$$\begin{aligned} f'(x_0) = f(x_0)\frac{2x_0-x_1-x_2}{(x_0-x_1)(x_0-x_2)} + f(x_1)\frac{2x_0-x_0-x_2}{(x_1-x_0)(x_1-x_2)} \\ + f(x_2)\frac{2x_0-x_0-x_1}{(x_2-x_0)(x_2-x_1)} \\ + \frac{f'''(\xi(x_0))}{6}\left((x_0-x_1)(x_0-x_2)+(x_0-x_0)(2x_0-x_1-x_2)\right) \\ + D_x\left(\frac{f'''(\xi(x_0))}{6}\right)(x_0-x_0)(x_0-x_1)(x_0-x_2). \end{aligned} \tag{8.51}$$

Considerando uma malha equiespaçada onde $x_1 = x_0 + h$ e $x_2 = x_0 + 2h$, temos:

$$\begin{aligned} f'(x_0) &= f(x_0)\frac{-3h}{(-h)(-2h)} + f(x_1)\frac{-2h}{(h)(-h)} \\ &\quad + f(x_2)\frac{-h}{(2h)(h)} + \frac{f'''(\xi(x_0))}{6}\left((-h)(-2h)\right) \\ &= \frac{1}{h}\left[-\frac{3}{2}f(x_0) + 2f(x_1) - \frac{1}{2}f(x_2)\right] + h^2\frac{f'''(\xi(x_0))}{3} \end{aligned} \tag{8.52}$$

Similarmente, trocando x por x_1 ou trocando x por x_2 na expressão (8.50), temos outras duas expressões

$$f'(x_1) = \frac{1}{h}\left[-\frac{1}{2}f(x_0) + \frac{1}{2}f(x_2)\right] + h^2\frac{f'''(\xi(x_1))}{6} \quad (8.53)$$

$$f'(x_2) = \frac{1}{h}\left[\frac{1}{2}f(x_0) - 2f(x_1) + \frac{3}{2}f(x_2)\right] + h^2\frac{f'''(\xi(x_2))}{3} \quad (8.54)$$

Podemos reescrever as três fórmulas da seguinte forma:

$$f'(x_0) = \frac{1}{h}\left[-\frac{3}{2}f(x_0) + 2f(x_0+h) - \frac{1}{2}f(x_0+2h)\right] + h^2\frac{f'''(\xi(x_0))}{3} \quad (8.55)$$

$$f'(x_0+h) = \frac{1}{h}\left[-\frac{1}{2}f(x_0) + \frac{1}{2}f(x_0+2h)\right] + h^2\frac{f'''(\xi(x_0+h))}{6} \quad (8.56)$$

$$f'(x_0+2h) = \frac{1}{h}\left[\frac{1}{2}f(x_0) - 2f(x_0+h) + \frac{3}{2}f(x_0+2h)\right] + h^2\frac{f'''(\xi(x_0+2h))}{3} \quad (8.57)$$

ou ainda

$$f'(x_0) = \frac{1}{2h}\left[-3f(x_0) + 4f(x_0+h) - f(x_0+2h)\right] + h^2\frac{f'''(\xi(x_0))}{3} \quad (8.58)$$

$$f'(x_0) = \frac{1}{2h}\left[f(x_0+h) - f(x_0-h)\right] + h^2\frac{f'''(\xi(x_0))}{6} \quad (8.59)$$

$$f'(x_0) = \frac{1}{2h}\left[f(x_0-2h) - 4f(x_0-h) + 3f(x_0)\right] + h^2\frac{f'''(\xi(x_0))}{3} \quad (8.60)$$

Observe que uma das fórmulas é exatamente as diferenças centrais obtida anteriormente.

Analogamente, para construir as fórmulas de cinco pontos tomamos o polinômio de Lagrange para cinco pontos e chegamos a cinco fórmulas, sendo uma delas a seguinte:

$$f'(x_0) = \frac{1}{12h}\left[f(x_0-2h) - 8f(x_0-h) + 8f(x_0+h) - f(x_0+2h)\right] + \frac{h^4}{30}f^{(5)}(\xi(x_0)) \quad (8.61)$$

Exemplo 8.3.1. Calcule a derivada numérica de $f(x) = e^{-x^2}$ em $x = 1{,}5$ pelas fórmulas de três e cinco pontos para $h = 0{,}1$, $h = 0{,}01$ e $h = 0{,}001$.

Solução. Em `Python`, podemos computar estas derivadas numéricas com $h = 0.1$ da seguinte forma:

```
>>> def f(x):
>>> ...     return np.exp(-x**2)
>>> x=1.5
```

Diferenças Finitas	$h = 0{,}1$	$0{,}01$	$0{,}001$
Progressiva $\mathcal{O}(h)$	$-0{,}2809448$	$-0{,}3125246$	$-0{,}3158289$
Regressiva $\mathcal{O}(h)$	$-0{,}3545920$	$-0{,}3199024$	$-0{,}3165667$
Progressiva $\mathcal{O}(h^2)$	$-0{,}3127746$	$-0{,}3161657$	$-0{,}3161974$
Central $\mathcal{O}(h^2)$	$-0{,}3177684$	$-0{,}3162135$	$-0{,}3161978$
Regressiva $\mathcal{O}(h^2)$	$-0{,}3135824$	$-0{,}3161665$	$-0{,}3161974$
Central $O(h^4)$	$-0{,}3162384$	$-0{,}3161977$	$-0{,}31619767$

Tabela 8.1: Derivadas numéricas de $f(x) = e^{-x^2}$ em $x = 1{,}5$. Veja o Exemplo 8.3.1.

```
>>> h=0.1
>>> #progressiva de ordem 1
>>> dp1 = (f(x+h)-f(x))/h
>>> #regressiva de ordem 1
>>> dr1 = (f(x)-f(x-h))/h
>>> #central de ordem 2
>>> dc2 = (f(x+h)-f(x-h))/(2*h)
>>> #progressiva de ordem 2
>>> dp2 = (-3*f(x)+4*f(x+h)-f(x+2*h))/(2*h)
>>> #regressiva de ordem 2
>>> dr2 = (f(x-2*h)-4*f(x-h)+3*f(x))/(2*h)
>>> #central de ordem 4
>>> dc4 = (f(x-2*h)-8*f(x-h)+8*f(x+h)-f(x+2*h))/(12*h)
```

e, análogo, para $h = 0.01$ e $h = 0.001$. O valor analítico da derivada é $f'(1{,}5) \approx -0{,}3161976736856$. A Tabela 8.1 mostra os resultados computados com as derivadas numéricas.

$\Diamond$

8.3.1 Exercícios resolvidos

Esta seção carece de exercícios resolvidos. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

Exercícios

Esta seção carece de exercícios. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

8.4 Fórmulas de diferenças finitas

Veremos nessa seção uma outra maneira de obter fórmulas de diferenças finitas para derivadas de qualquer ordem de tal forma que elas possuam alta ordem de precisão.

Dados $n+1$ pontos $\{x_1, x_2, \ldots, x_n\}$, queremos obter uma aproximação para a derivada de $f(x)$ calculada em x^* do tipo

$$f'(x^*) \approx c_1 f(x_1) + c_2 f(x_2) + \ldots + c_n f(x_n) \tag{8.62}$$

que seja exata para polinômios até ordem $n-1$.

Seja $q(x) = c_1\phi_1(x) + c_2\phi_2(x) + \ldots + c_n\phi_n(x)$ o polinômio de ordem n que aproxima $f(x)$. Fixe a base $\phi_k(x) = x^k$. Como a regra (8.62) deve ser exata para qualquer $q(x)$ até ordem $n-1$, então também deve ser exata para qualquer função da base. Substituindo $f(x)$ por $\phi_1(x) = 1$ em (8.62) obtemos

$$\begin{aligned}
\phi_1'(x)|_{x^*} &= (1)'|_{x^*} = & (8.63)\\
0 &= c_1\phi_1(x_1) + c_2\phi_1(x_2) + \ldots + c_n\phi_1(x_n) & (8.64)\\
0 &= c_1 + c_2 + \ldots + c_n & (8.65)
\end{aligned}$$

Da mesma forma para $k = 1, \ldots, n-1$, obtemos

$$\begin{aligned}
(x)'_{x^*} = 1 &= c_1x_1 + c_2x_2 + \ldots + c_nx_n & (8.66)\\
(x^2)'_{x^*} = 2x^* &= c_1x_1^2 + c_2x_2^2 + \ldots + c_nx_n^2 & (8.67)\\
(x^3)'_{x^*} = 3(x^*)^2 &= c_1x_1^3 + c_2x_2^3 + \ldots + c_nx_n^3 & (8.68)\\
\vdots &= \vdots & (8.69)\\
(x^{n-1})'_{x^*} = (n-1)(x^*)^{n-2} &= c_1x_1^{n-1} + c_1x_1^{n-1} + \ldots + c_nx_n^{n-1} & (8.70)
\end{aligned}$$

que pode ser escrito na forma matricial

$$\begin{bmatrix} 1 & 1 & \dots & 1 \\ x_1 & x_2 & \dots & x_n \\ x_1^2 & x_2^2 & \dots & x_n^2 \\ \vdots & \vdots & & \vdots \\ x_1^{n-1} & x_2^{n-1} & \dots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_n \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 2x^* \\ \vdots \\ (n-1)(x^*)^{n-2} \end{bmatrix} \tag{8.71}$$

Resolvendo o sistema, obtemos os coeficientes c_k para a regra de diferenciação.

Exemplo 8.4.1. Sejam $\{x_1, x_2, x_3\} = \{-h, 0, h\}$ e $x^* = x_2 = 0$, obtenha uma regra de diferenciação para aproximar $f'(x^*)$.

Solução. A regra terá a forma

$$f'(x^*) \approx c_1 f(x_1) + c_2 f(x_2) + c_3 f(x_3) \tag{8.72}$$

Considere a base polinomial $\{\phi_1(x), \phi_2(x), \phi_3(x)\} = \{1, x, x^2\}$ e substitua $f(x)$ por $\phi_k(x)$ obtendo

$$(1)'_{x=0} = 0 \quad = c_1(1) + c_2(1) + c_3(1) \tag{8.73}$$
$$(x)'_{x=0} = 1 \quad = c_1(-h) + c_2(0) + c_3(h) \tag{8.74}$$
$$(x^2)'_{x=0} = 0 \quad = c_1(-h)^2 + c_2(0)^2 + c_3(h)^2 \tag{8.75}$$

que pode ser escrito na forma matricial

$$\begin{bmatrix} 1 & 1 & 1 \\ -h & 0 & h \\ h^2 & 0 & h^2 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{8.76}$$

Resolvendo o sistema, obtemos $\{c_1, c_2, c_3\} = \{-\frac{1}{2h}, 0, \frac{1}{2h}\}$ fornecendo a regra

$$f'|_{x=x_1} \approx -\frac{1}{2h} f(x_1) + \frac{1}{2h} f(x_3) \tag{8.77}$$
$$\approx \frac{f(x_3) - f(x_1)}{2h} \tag{8.78}$$

$\Diamond$

Exercícios resolvidos

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:

https://www.ufrgs.br/reamat/participe.html

Exercícios

E 8.4.1. Seja $\{x_0, x_1, x_2\} = \{0, h, 2h\}$ e $x^* = x_0 = 0$, obtenha uma regra unilateral de diferenciação para aproximar $f'(x_0)$.

E 8.4.2. Seja $\{x_0, x_1, x_2\} = \{-h, 0, h\}$ e $x^* = x_1 = 0$, obtenha uma regra de diferenciação para aproximar $f''(x^*)$.

E 8.4.3. Seja $\{x_0, x_1, \ldots, x_4\} = \{-2h, -h, 0, h, 2h\}$ e $x^* = 0$, obtenha uma regra de diferenciação para aproximar $f'(x^*)$.

E 8.4.4. Seja $[x_0,x_1, \ldots ,x_4] = [-2h, -h,0,h,2h]$ e $x^* = 0$, obtenha uma regra de diferenciação para aproximar $f''(x^*)$.

E 8.4.5. Seja $[x_0,x_1, \ldots ,x_4] = [0,h,3h,6h,10h]$ e $x^* = 0$, obtenha uma regra de diferenciação para aproximar $f'(x^*)$.

8.5 Derivada via ajuste ou interpolação

Dados os valores de uma função em um conjuntos de pontos $\{(x_i,y_i)\}_{i=1}^N$, as derivadas $\left(\frac{dy}{dx}\right)_i$ podem ser obtidas através da derivada de uma curva que melhor ajusta ou interpola os pontos. Esse tipo de técnica é necessário quando os pontos são muito espaçados entre si ou quando a função oscila muito. Por exemplo, dados os pontos (0,1), (1,2), (2,5), (3,9), a parábola que melhor ajusta os pontos é

$$Q(x) = 0{,}95 + 0{,}45x + 0{,}75x^2. \tag{8.79}$$

Usando esse ajuste para calcular as derivadas, temos:

$$Q'(x) = 0{,}45 + 1{,}5x \tag{8.80}$$

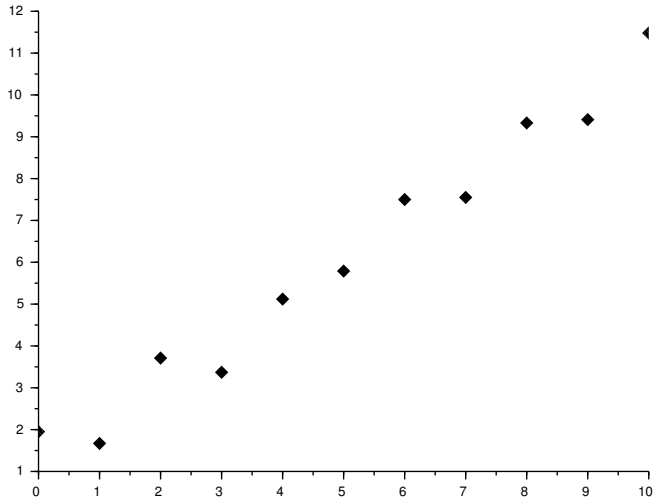
e

$$y'(x_1) \approx Q'(x_1) = 0{,}45, \qquad y'(x_2) \approx Q'(x_2) = 1{,}95, \tag{8.81}$$

$$y'(x_3) \approx Q'(x_3) = 3{,}45 \qquad \text{e} \qquad y'(x_4) \approx Q'(x_4) = 4{,}95 \tag{8.82}$$

Agora olhe o gráfico da seguinte tabela de pontos.

x	y
0	1,95
1	1,67
2	3,71
3	3,37
4	5,12
5	5,79
6	7,50
7	7,55
8	9,33
9	9,41
10	11,48



Observe que as derivadas calculadas por diferenças finitas oscilam entre um valor pequeno e um grande em cada intervalo e além disso, a fórmula progressiva difere da regressiva significantemente. Por exemplo, por diferenças regressivas $f'(7) \approx \frac{(7{,}55-7{,}50)}{1} = 0{,}05$ e por diferenças progressivas $f'(7) \approx \frac{(9{,}33-7{,}55)}{1} = 1{,}78$. A melhor forma de calcular a derivada aqui é fazer um ajuste de curva. A reta que

melhor ajusta os dados da tabela é $y = f(x) = 1{,}2522727 + 0{,}9655455x$. Usando esse ajuste, temos $f'(7) \approx 0{,}9655455$.

Exercícios resolvidos

Esta seção carece de exercícios resolvidos. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

Exercícios

Esta seção carece de exercícios. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

8.6 Exercícios finais

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:

https://www.ufrgs.br/reamat/participe.html

Capítulo 9

Integração numérica

Neste capítulo discutiremos técnicas numéricas para aproximar **integrais** definidas de funções reais. Mais precisamente, considere o problema de calcular (ou aproximar) a integral de $f(x)$ no intervalo $[a,b]$, ou seja,

$$I = \int_a^b f(x)\, dx. \tag{9.1}$$

Geometricamente, I corresponde a área[1] entre o gráfico de $f(x)$ e o eixo das abscissas (eixo x). Uma maneira de aproximar I consiste em subdividir o intervalo $[a,b]$ em $n-1$ subintervalos a partir de um conjunto ordenado de pontos $a = x_1 < x_2 < ... < x_n = b$. Então, temos:

$$\begin{aligned} I &= \int_a^b f(x)\, dx \\ &= \int_{x_1}^{x_2} f(x)\, dx + \int_{x_2}^{x_3} f(x)\, dx + \cdots + \int_{x_{n-1}}^{x_n} f(x)\, dx \\ &= \sum_{i=1}^{n-1} \int_{x_i}^{x_{i+1}} f(x)\, dx \end{aligned} \tag{9.2}$$

Agora, supondo que o tamanho de cada cada subintervalo $h_i = x_{i+1} - x_i$ é suficientemente pequeno, podemos aproximar $f(x)$ no intervalo (x_i, x_{i+1}) por $f(x_i^*)$ escolhendo arbitrariamente $x_i^* \in [x_i, x_{i+1}]$. Desta forma, temos

$$\int_{x_i}^{x_{i+1}} f(x)\, dx \approx f(x_i^*)h_i. \tag{9.3}$$

Isto é equivalente a aproximar a área entre o gráfico de $f(x)$ e o eixo x restrito ao intervalo $[x_i, x_{i+1}]$ pelo retângulo de base h_i e altura $f(x_i^*)$ (veja Figura 9.1).

[1] área líquida

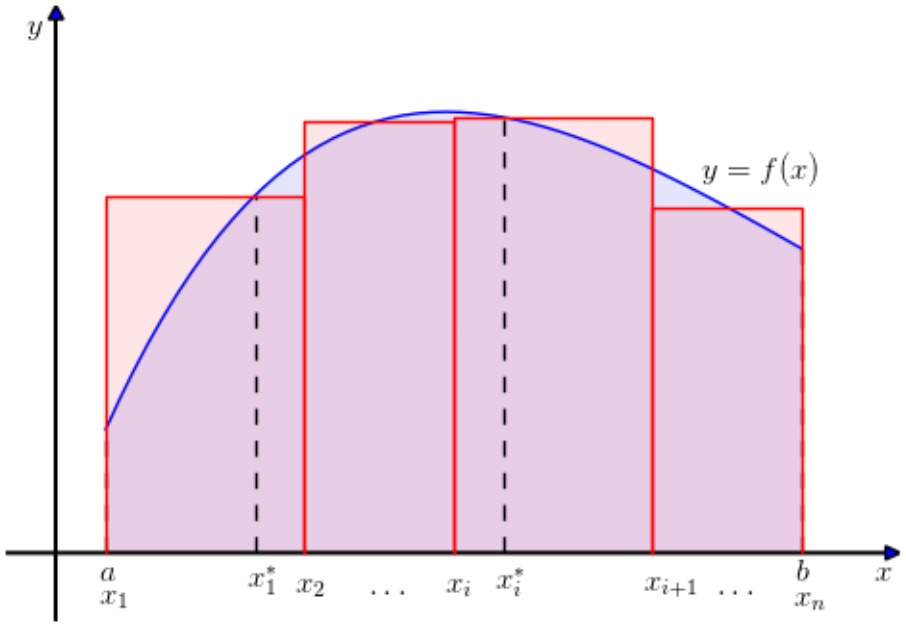


Figura 9.1: Aproximação da integral definida de uma função.

Consequentemente, de (9.2) temos

$$I = \int_a^b f(x)\,dx = \sum_{i=1}^{n-1} \int_{x_i}^{x_{i+1}} f(x)\,dx \tag{9.4}$$

$$I \approx \sum_{i=1}^{n-1} f(x_i^*)h_i. \tag{9.5}$$

Exemplo 9.0.1. A Figura 9.2 mostra um exemplo quando $f(x) = x^2 + 1$, $0 \leq x \leq 2$. Temos a aproximação por um retângulo com base $h_1 = 2$, depois com dois retângulos de base $h_2 = 1$ e, finalmente com quatro retângulos de bases $h_3 = 0{,}5$. Os valores aproximados para a integral são dados na seguinte tabela:

	$\int_0^2 (x^2+1)\,dx$
$h_1 = 2$	$h_1 f(1) = 4$
$h_2 = 1$	$h_2 f(0{,}5) + h_2 f(1{,}5) = 4{,}5$
$h_3 = 0{,}5$	4,625
$h_4 = 0{,}25$	4,65625

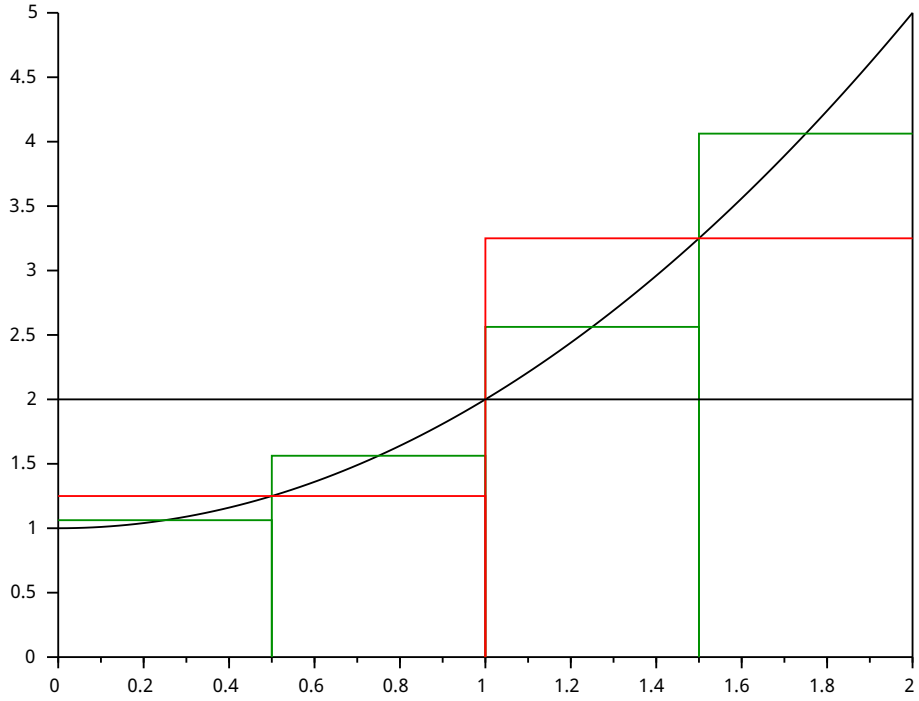


Figura 9.2: Aproximação por retângulos.

Observe que:

$$\int_0^2 (x^2+1)\,dx = \left[\frac{x^3}{3}+x\right]_0^2 = \frac{8}{3}+2 = 4{,}6666667. \tag{9.6}$$

Uma tal aproximação de uma integral definida

$$\int_a^b f(x)\,dx \approx \sum_i f(x_i)w_i, \tag{9.7}$$

é chamada de quadratura numérica, onde os números x_i denota seu i-ésimo ponto e w_i seu i-ésimo peso. Nas próximas seções, mostraremos como obter diferentes quadraturas numéricas e discutiremos sobre suas características.

Nos códigos `Python` apresentados ao longo deste capítulo, assumiremos o seguinte:

```
>>> from __future__ import division
>>> import numpy as np
```

9.1 Somas de Riemann

O método mais simples de aproximar

$$I = \int_a^b f(x)\,dx. \tag{9.8}$$

com apenas um intervalo, é aproximar $f(x)$ por um polinômio constante no intervalo $[a,b]$, ou seja, $f(x) = c$. Se aproximarmos $f(x)$ pelo ponto a esquerda do intervalo temos que $f(x) \approx f(a)$ e

$$\begin{aligned} I &= \int_a^b f(x)\,dx && (9.9)\\ &\approx \int_a^b f(a)\,dx && (9.10)\\ &= f(a)\int_a^b dx && (9.11)\\ &= f(a)(b-a) && (9.12) \end{aligned}$$

Esta é a regra de quadratura local para 1 intervalo.

Quando subdividimos $[a,b]$ em n intervalos com tamanho $h = (b-a)/n$ nos pontos $x_i = a + (i-1)h$, em cada intervalo i aproximamos a área por

$$\Delta S_i \approx f(x_i)h \tag{9.13}$$

tal que a área total será aproximada pelas **somas de Riemann à esquerda**

$$S = \sum_{i=1}^{n-1} \Delta S_i = \sum_{i=1}^{n-1} f(x_i)h \tag{9.14}$$

Podemos obter uma fórmula similar se usarmos os pontos a direita do intervalo, ou seja, as **somas de Riemann à direita**

$$S = \sum_{i=1}^{n-1} f(x_{i+1})h \tag{9.15}$$

Uma terceira opção é utilizar o ponto médio do intervalo $[x_i,x_{i+1}]$ o qual fornece a **regra do ponto médio**

$$S = \sum_{i=1}^{n-1} f(\xi_i)h, \qquad \xi_i = \frac{x_i + x_{i+1}}{2}. \tag{9.16}$$

Exemplo 9.1.1. A integral de $f(x) = e^{-x}\,\mathrm{sen}\,(x)$ no intervalo $[0{,}1, 0{,}2]$ é

$$\int_0^1 f(x)\,dx \approx 2{,}45837\mathrm{E}{-1}. \tag{9.17}$$

Usando somas de Riemann à esquerda com 10 intervalos, obtemos

$$\int_0^1 f(x)\,dx \approx \sum_{i=1}^{10} f(x_i)h = 2{,}29433 \times 10^{-1}. \tag{9.18}$$

onde $h = 0{,}1$ e $x_i = (i-1)h$. Analogamente, usando somas de Riemann à direita, obtemos

$$\int_0^1 f(x)\,dx \approx \sum_{i=1}^{10} f(x_{i+1})h = 2{,}60389 \times 10^{-1}. \tag{9.19}$$

E, usando a regra do ponto médio, temos

$$\int_0^1 f(x)\,dx \approx \sum_{i=1}^{10} f\left(\frac{x_i + x_{i+1}}{2}\right)h = 2{,}46300 \times 10^{-1}. \tag{9.20}$$

Pode-se, no `Python`, implementar as somas de Riemman da seguinte forma

```
f = lambda x: np.exp(-x)*np.sin(x)

a = 0
b = 1
n = 10
```

```
h = (b-a)/n
x = np.linspace(a,b,n+1)

S_esq = 0
S_dir = 0
S_med = 0

print("Soma de Riemman de {} a {} com {} intervalos:\n".format(a, b, n))

for i in range(n):
    S_esq += f(x[i])*h
print("A esquerda: {:.5e}".format(S_esq))

for i in range(n):
    S_dir += f(x[i+1])*h
print("A direita: {:.5e}".format(S_dir))

for i in range(n):
    S_med += f(((x[i]) + (x[i+1]))/2)*h
print("Pelo ponto medio: {:.5e}".format(S_med))
```

Exercícios resolvidos

Esta seção carece de exercícios resolvidos. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

Exercícios

Esta seção carece de exercícios. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

9.2 Regras de Newton-Cotes

O método básico para encontrar as regras de integração consiste em aproximar a integral de f por uma combinação linear de n valores de $y_i := f(x_i)$, ou seja,

$$I = \int_a^b f(x)\, dx \approx \sum_{i=1}^{n} A_i y_i. \tag{9.21}$$

Podemos obter os coeficientes A_i aproximando a função f pelo polinômio de Lagrange p_{n-1} que interpola $\{(x_i,y_i)\}_{i=1}^n$, tal que,

$$\begin{aligned} f(x) &= p_n(x) + E^n_{LAG}(x) && (9.22)\\ &= \sum_{i=1}^{n} y_i L_i(x) + E^n_{LAG}(x) && (9.23) \end{aligned}$$

onde o erro na interpolação de Lagrange é

$$E^n_{LAG}(x) = \frac{f^{(n)}(\xi(x))}{n!} \prod_{i=1}^{n} (x - x_i). \tag{9.24}$$

Substituindo na integral, obtemos:

$$\int_a^b f(x)\, dx = \sum_{i=1}^{n} \left[y_i \int_a^b L_i(x)\, dx \right] + \int_a^b E^n_{LAG}(x)\, dx. \tag{9.25}$$

A fórmula de quadratura é então

$$\int_a^b f(x)\, dx \approx \sum_{i=1}^{n} A_i y_i, \tag{9.26}$$

onde

$$A_i = \int_a^b L_i(x)\, dx. \tag{9.27}$$

9.2.1 Regra do ponto médio

A regra do ponto médio (9.16) é uma quadratura de Newton-Cotes de um ponto. Neste caso, temos $x_1 = (a+b)/2$ e o polinômio interpolador é o polinômio de grau zero

$$p(x) = f(x_1)L_1(x) = f(x_1), \tag{9.28}$$

uma vez que $L_1(x) \equiv 1$. Então, temos

$$\begin{aligned} \int_a^b f(x)\,dx &\approx \int_a^b p(x)\,dx \\ &= \int_a^b f(x_1)\,dx \\ &= f(x_1)\int_a^b dx \\ &= hf(\frac{a+b}{2}), \end{aligned} \tag{9.29}$$

onde $h = b - a$.

Exemplo 9.2.1. Usando a regra do ponto médio, temos

$$\begin{aligned} \int_{0,1}^{0,3} e^{-x}\,\text{sen}\,(x)\,dx &\approx f(\frac{a+b}{2}) \\ &= 0{,}2e^{-0{,}2}\,\text{sen}\,(0{,}2) = 3{,}25313 \times 10^{-2}. \end{aligned} \tag{9.30}$$

No `Python`, computamos:

```
f = lambda x: np.exp(-x)*np.sin(x)

a=0.1
b=0.3
h=b-a

I = h*f((a+b)/2)
```

9.2.2 Regra do trapézio

A regra do trapézio consiste em aproximar a função $f(x)$ por um polinômio de grau 1. O nome do método vem do fato que a região entre o eixo x e a reta que liga o pontos sobre o gráfico da função nos extremos do intervalo forma um trapézio.

Aqui, utilizamos $x_1 := a$, $x_2 := b$, $h = x_2 - x_1$ e a notação $y_i = f(x_i)$, obtemos através da interpolação de Lagrange o polinômio

$$p_1(x) \quad = \quad y_1 L_1(x) + y_2 L_2(x) \tag{9.31}$$

Aproximando $f(x)$ por $p_1(x)$ e integrando, obtemos:

$$\begin{aligned}\int_a^b f(x)\,dx &\approx \int_a^b p_1(x)\,dx &(9.32)\\ &= \int_a^b y_1L_1(x)+y_2L_2(x)\,dx &(9.33)\\ &= y_1\int_a^b L_1(x)\,dx + y_2\int_a^b L_2(x)\,dx &(9.34)\\ &= A_1y_1+A_2y_2, &(9.35)\end{aligned}$$

onde

$$\begin{aligned}A_1 &= \int_a^b \frac{x-x_1}{x_2-x_1}dx = \left[\frac{(x-x_1)^2}{2h}\right]_{x_1}^{x_2} &(9.36)\\ &= \frac{(x_2-x_1)^2}{2h} = \frac{h^2}{2h} = \frac{1}{2}h. &(9.37)\end{aligned}$$

Da mesma forma,

$$A_2 = \int_a^b \frac{(x-x_2)}{(x_1-x_2)}\,dx = \frac{1}{2}h, \tag{9.38}$$

de onde obtemos a **regra do trapézio** dada por:

$$\int_a^b f(x)\,dx \approx \left(\frac{1}{2}f(a)+\frac{1}{2}f(b)\right)h. \tag{9.39}$$

Erro na regra do trapézio

O *erro na regra do trapézio* pode ser obtido integrando o erro da interpolação de Lagrange,

$$E_{TRAP} = \int_a^b E^2_{LAG}(x)\,dx = \int_a^b \frac{f''(\xi(x))}{2!}(x-x_1)(x-x_2)\,dx. \tag{9.40}$$

Pelo teorema do valor médio, existe $a \le \eta \le b$ tal que

$$E_{TRAP} = \frac{f''(\eta)}{2!}\int_a^b (x-x_1)(x-x_2)\,dx, \tag{9.41}$$

portanto

$$E_{TRAP} = \frac{f''(\eta)}{2}\left[\frac{x^3}{3} - \frac{x^2}{2}(x_2+x_1) + x_1x_2x\right]_{x_1}^{x_2} \tag{9.42}$$

$$= \frac{f''(\eta)}{2}\left(\frac{x_2^3}{3} - \frac{x_2^2}{2}(x_2+x_1) + x_1x_2x_2 - \frac{x_1^3}{3} + \frac{x_1^2}{2}(x_2+x_1) - x_1x_2x_1\right) \tag{9.43}$$

$$= \frac{f''(\eta)}{2}\frac{2x_2^3 - 3x_2^2(x_2+x_1) + 6x_2^2x_1 - 2x_1^3 + 3x_1^2(x_2+x_1) - 6x_2x_1^2}{6} \tag{9.44}$$

$$= \frac{f''(\eta)}{12}\left(x_1^3 - 3x_1^2x_2 + 3x_2^2x_1 - x_2^3\right) = \frac{f''(\eta)}{12}(x_1-x_2)^3 \tag{9.45}$$

$$= -\frac{f''(\eta)}{12}h^3. \tag{9.46}$$

Assim, o erro na regra do trapézio é

$$E_{TRAP} = -\frac{f''(\eta)}{12}h^3 = \mathcal{O}(h^3). \tag{9.47}$$

Exemplo 9.2.2. Use a regra do trapézio para aproximar a integral

$$\int_0^1 e^{-x^2}\,dx. \tag{9.48}$$

Depois divida a integral em duas

$$\int_0^{1/2} e^{-x^2}\,dx + \int_{1/2}^1 e^{-x^2}\,dx. \tag{9.49}$$

e aplique a regra do trapézio em cada uma delas. Finalmente, repita o processo dividindo em quatro integrais.

Usando o intervalo [0,1], temos $h = 1$, $x_0 = 0$ e $x_1 = 1$. A regra do trapézio resulta em

$$\int_0^1 e^{-x^2}\,dx \approx \frac{1}{2}(e^0 + e^{-1}) = 0{,}6839397. \tag{9.50}$$

Usando dois intervalos, [0,1/2] e [1/2,1] e usando a regra de Simpson em cada um dos intervalos, temos:

$$\int_0^1 e^{-x^2}\,dx \approx \frac{0{,}5}{2}\left(e^0 + e^{-1/4}\right) + \frac{0{,}5}{2}\left(e^{-1/4} + e^{-1}\right) \tag{9.51}$$

$$= 0{,}4447002 + 0{,}2866701 = 0{,}7313703. \tag{9.52}$$

Agora, usando quatro intervalos, temos

$$\int_0^1 e^{-x^2}\,dx \approx \frac{0{,}25}{2}\left(e^0 + e^{-1/16}\right) + \frac{0{,}25}{2}\left(e^{-1/16} + e^{-1/4}\right) \tag{9.53}$$

$$+ \frac{0{,}25}{2}\left(e^{-1/4} + e^{-9/16}\right) + \frac{0{,}25}{2}\left(e^{-9/16} + e^{-1}\right) \tag{9.54}$$

$$= 0{,}7429841. \tag{9.55}$$

9.2.3 Regra de Simpson

Na regra de Simpson aproximamos f por um polinômio de grau 2, portanto precisamos de três pontos do intervalo $[a,b]$. Utilizando, por definição,

$$x_1 := a, \qquad x_2 := \frac{a+b}{2} \qquad \text{e} \qquad x_3 := b \tag{9.56}$$

com $h = \frac{x_3 - x_1}{2}$, isto é, a distância entre dois pontos consecutivos, podemos obter o polinômio de Lagrange

$$p_2(x) = y_1 L_1(x) + y_2 L_2(x) + y_3 L_3(x), \tag{9.57}$$

onde $y_i = f(x_i)$, $i = 1,2,3$.

Aproximando $f(x)$ por $p_2(x)$ e integrando temos

$$\begin{aligned} \int_a^b f(x)\,dx &\approx \int_a^b p_2(x)\,dx && (9.58)\\ &= \int_a^b y_1 L_1(x) + y_2 L_2(x) + y_3 L_3(x)\,dx && (9.59)\\ &= y_1 A_1 + y_2 A_2 + y_3 A_3 && (9.60) \end{aligned}$$

onde

$$A_i = \int_a^b L_i(x)\,dx \tag{9.61}$$

Calculando essas integrais obtemos **a regra de Simpson**:

$$\int_a^b f(x)\,dx = \left(\frac{1}{3}f(a) + \frac{4}{3}f\left(\frac{a+b}{2}\right) + \frac{1}{3}f(b)\right) h. \tag{9.62}$$

Exemplo 9.2.3. Obtenha os coeficientes A_i do método de Simpson integrando os polinômios de Lagrange $L_i(x)$.

Fazendo uma translação para a origem (subtraindo x_1 de x_2 e x_3)

$$\begin{aligned} A_1 &= \int_{x_1}^{x_3} \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)}\,dx && (9.63)\\ &= \int_0^{2h} \frac{(x-h)(x-2h)}{(0-h)(0-2h)}\,dx = \frac{1}{2h^2}\int_0^{2h}(x-h)(x-2h)\,dx && (9.64)\\ &= \frac{1}{2h^2}\int_0^{2h}\left(x^2 - 3hx + 2h^2\right)dx = \frac{1}{2h^2}\left(\frac{1}{3}x^3 - \frac{3}{2}hx^2 + 2h^2x\right)\Big|_0^h && (9.65)\\ &= \frac{1}{2h^2}\left(\frac{1}{3}h^3 - \frac{3}{2}h^3 + 2h^3\right) = \frac{h}{3}. && (9.66) \end{aligned}$$

Apesar de longa, é apenas a integral de um polinômio de grau 2. De forma semelhante podemos obter

$$A_2 = \frac{4}{3}h, \quad A_3 = \frac{1}{3}h \tag{9.67}$$

Assim, lembrando que $h = \frac{b-a}{2}$, temos:

$$\int_a^b f(x)dx \approx \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]. \tag{9.68}$$

Erro na regra de Simpson

Se usarmos a mesma metodologia da regra dos trapézios, teremos

$$\int_a^b f(x)\,dx = \int_a^b p_2(x)\,dx + \int_a^b \frac{(x-x_1)(x-x_2)(x-x_3)}{6} f'''(\xi(x))\,dx \tag{9.69}$$

e obteremos o fórmula de Simpson com um erro de quarta ordem. O fato é que a regra de Simpson tem ordem cinco e, para isso, usaremos uma abordagem alternativa.

Considere o polinômio de Taylor em x_2,

$$f(x) = f(x_2)+f'(x_2)(x-x_2)+\frac{f''(x_2)}{2}(x-x_2)^2+\frac{f'''(x_2)}{6}(x-x_2)^3+\frac{f^{(4)}(\xi(x))}{24}(x-x_2)^4, \tag{9.70}$$

onde $x_1 \leq \xi(x) \leq x_3$ e integre no intervalo $[a,b] = [x_1,x_3]$:

$$\begin{aligned}\int_a^b f(x)\,dx = &\left[f(x_2)(x-x_2) + f'(x_2)\frac{(x-x_2)^2}{2} + \frac{f''(x_2)}{6}(x-x_2)^3\right.\\ &\left.+\frac{f'''(x_2)}{24}(x-x_2)^4\right]_{x_1}^{x_3}\\ &+\frac{1}{24}\int_{x_1}^{x_3} f^{(4)}(\xi(x))(x-x_2)^4\,dx,\end{aligned} \tag{9.71}$$

Pelo teorema do valor médio, existe $x_1 \leq \eta \leq x_3$ tal que

$$\begin{aligned}\int_a^b f(x)\,dx &= \left[f(x_2)(x-x_2) + f'(x_2)\frac{(x-x_2)^2}{2} + \frac{f''(x_2)}{6}(x-x_2)^3\right.\\ &\quad \left. + \frac{f'''(x_2)}{24}(x-x_2)^4\right]_{x_1}^{x_3}\\ &\quad + \frac{f^{(4)}(\eta)}{24}\int_{x_1}^{x_3}(x-x_2)^4\,dx\\ &= \left[f(x_2)(x-x_2) + f'(x_2)\frac{(x-x_2)^2}{2} + \frac{f''(x_2)}{6}(x-x_2)^3\right.\\ &\quad \left. + \frac{f'''(x_2)}{24}(x-x_2)^4\right]_{x_1}^{x_3}\\ &\quad + \frac{f^{(4)}(\eta)}{120}\left[(x-x_2)^5\right]_{x_1}^{x_3}.\end{aligned} \tag{9.72}$$

Usando o fato que

$$(x_3-x_2)^3 - (x_1-x_2)^3 = 2h^3, \tag{9.73}$$

$$(x_3-x_2)^4 - (x_1-x_2)^4 = 0 \tag{9.74}$$

e

$$(x_3-x_2)^5 - (x_1-x_2)^5 = 2h^5, \tag{9.75}$$

temos

$$\int_a^b f(x)\,dx = hf(x_2) + \frac{h^3}{3}f''(x_2) + \frac{h^5 f^{(4)}(\eta)}{60}. \tag{9.76}$$

Usando a fórmula de diferenças finitas centrais para a derivada segunda:

$$f''(x_2) = \frac{f(x_1) - 2f(x_2) + f(x_3)}{h^2} + \frac{h^2}{12}f^{(4)}(\eta_2), \tag{9.77}$$

$x_1 \leq \eta_2 \leq x_3$, temos

$$\int_a^b f(x)\,dx = 2hf(x_2) + \frac{h^3}{3}\left(\frac{f(x_1) - 2f(x_2) + f(x_3)}{h^2} + \frac{h^2}{12}f^{(4)}(\eta_2)\right) \tag{9.78}$$

$$+ \frac{h^5 f^{(4)}(\eta)}{60} \tag{9.79}$$

$$= \frac{h}{3}\left(f(x_1) + 4f(x_2) + f(x_3)\right) - \frac{h^5}{12}\left(\frac{1}{3}f^{(4)}(\eta_2) - \frac{1}{5}f^{(4)}(\eta)\right) \tag{9.80}$$

Pode-se mostrar que é possível escolher η_3 que substitua η e η_2 com a seguinte estimativa

$$\int_a^b f(x)\,dx = \frac{h}{3}\left(f(x_1) + 4f(x_2) + f(x_3)\right) - \frac{h^5}{90}f^{(4)}(\eta_3). \tag{9.81}$$

Exemplo 9.2.4. Use a regra de Simpson para aproximar a integral

$$\int_0^1 e^{-x^2}\,dx. \tag{9.82}$$

Depois divida a integral em duas

$$\int_0^{1/2} e^{-x^2}\,dx + \int_{1/2}^1 e^{-x^2}\,dx. \tag{9.83}$$

e aplica a regra de Simpson em cada uma delas.

Usando o intervalo [0,1], temos $h = 1/2$, $x_0 = 0$, $x_1 = 1/2$ e $x_2 = 1$. A regra de Simpson resulta em

$$\int_0^1 e^{-x^2}\,dx \approx \frac{0{,}5}{3}(e^0 + 4e^{-1/4} + e^{-1}) = 0{,}7471804. \tag{9.84}$$

Usando dois intervalos, [0,1/2] e [1/2,1] e usando a regra do trapézio em cada um dos intervalos, temos:

$$\int_0^1 e^{-x^2}\,dx \approx \frac{0{,}25}{3}(e^0 + 4e^{-1/16} + e^{-1/4}) + \frac{0{,}25}{3}(e^{-1/4} + 4e^{-9/16} + e^{-1}) = 0{,}7468554. \tag{9.85}$$

Exercícios resolvidos

Esta seção carece de exercícios resolvidos. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

Exercícios

E 9.2.1. Calcule numericamente as seguintes integrais:

$$\text{a) } \int_0^1 e^{-x}\,dx \qquad \text{b) } \int_0^1 x^2\,dx \tag{9.86}$$

$$\text{c) } \int_0^1 x^3\,dx \quad \text{d) } \int_0^1 xe^{-x^2}\,dx \tag{9.87}$$

$$\text{e) } \int_0^1 \frac{1}{x^2+1}\,dx \quad \text{e) } \int_0^1 \frac{x}{x^2+1}\,dx \tag{9.88}$$

usando os métodos simples do ponto médio, Trapézio e Simpson. Calcule, também, o valor analítico destas integrais e o erro nas aproximações dadas pelas quadraturas numéricas.

E 9.2.2. Dê a interpretação geométrica dos métodos do ponto médio, trapézio e Simpson. A partir desta construção geométrica, deduza as fórmulas para aproximar

$$\int_a^b f(x)\,dx. \tag{9.89}$$

Verifique o método de Simpson pode ser entendido como uma média aritmética ponderada entre os métodos de trapézio e ponto médio. Encontre os pesos envolvidos. Explique o que são os métodos compostos.

E 9.2.3. Calcule numericamente o valor de $\int_2^5 e^{4-x^2}\,dx$ usando os métodos compostos do ponto médio, trapézio e Simpson. Obtenha os resultados utilizando, em cada quadratura, o número de pontos indicado.

n	Ponto médio	Trapézios	Simpson
3			
5			
7			
9			

9.3 Obtenção das regras de quadratura

Na seção anterior, obtivemos as regras de quadraturas pela aproximação do integrando por polinômios interpoladores de Lagrange. Aqui, veremos um outro método para obter regras de quadratura, que torna-se bastante útil para quando temos muitos pontos ou quando o intervalo entre os pontos não é uniforme.

Dados n pontos $[t_1, t_2, \ldots, t_n]$, queremos obter uma aproximação para

$$\int_a^b f(t)\,dt \approx w_1 f(t_1) + w_2 f(t_2) + \ldots + w_n f(t_n) \tag{9.92}$$

que seja exata para polinômios[2] até ordem $n-1$.

Aproxime $f(t)$ pelo polinômio $p(t) = w_1\phi_1(t) + \ldots + w_n\phi_n(t)$ de ordem $n-1$. Escolha uma base, como por exemplo $\phi_k(t) = t^{k-1}$. Como a regra de quadratura deve ser exata para qualquer polinômio até ordem $n-1$, então também deve ser

[2] Por exemplo, se $n = 2$, então a regra é exata para retas.

exata para qualquer função da base. Substituindo $f(t)$ por $\phi_1(t) = 1$ em (9.92). obtemos:

$$\int_a^b \phi_1(t)\,dt = t|_a^b = w_1\phi_1(t_1) + w_2\phi_1(t_2) + \ldots + w_n\phi_1(t_n) \tag{9.93}$$

$$b - a = w_1 + w_2 + \ldots + w_n. \tag{9.94}$$

Da mesma forma para $\phi_k(t)$, $k = 2, \ldots, n$, obtemos:

$$(t^2/2)|_a^b = \frac{b^2 - a^2}{2} = w_1t_1 + w_2t_2 + \ldots + w_nt_n \tag{9.95}$$

$$(t^3/3)|_a^b = \frac{b^3 - a^3}{3} = w_1t_1^2 + w_2t_2^2 + \ldots + w_nt_n^2 \tag{9.96}$$

$$\vdots \tag{9.97}$$

$$\frac{b^n - a^n}{n} = w_1t_1^{n-1} + w_2t_2^{n-1} + \ldots + w_nt_n^{n-1}, \tag{9.98}$$

que pode ser escrito na forma matricial a seguir:

$$\begin{bmatrix} 1 & 1 & \ldots & 1 \\ t_1 & t_2 & \ldots & t_n \\ t_1^2 & t_2^2 & \ldots & t_n^2 \\ \vdots & \vdots & & \vdots \\ t_1^{n-1} & t_2^{n-1} & \ldots & t_n^{n-1} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} b-a \\ \frac{b^2-a^2}{2} \\ \frac{b^3-a^3}{3} \\ \vdots \\ \frac{b^n-a^n}{n} \end{bmatrix}. \tag{9.99}$$

Resolvendo o sistema, obtemos os coeficientes w_k para a regra de integração.

Exemplo 9.3.1. Seja $n = 3$, $[a, b] = [0, h]$, onde $(t_1, t_2, t_3) = (0, h/2, h)$. Obtenha uma regra de integração para aproximar $\int_a^b f(t)\,dt$.

Solução. A regra terá a forma

$$\int_a^b f(t)\,dt \approx w_1f(t_1) + w_2f(t_2) + w_3f(t_3) \tag{9.100}$$

$$\approx w_1y_1 + w_2y_2 + w_3y_3. \tag{9.101}$$

Considere a base polinomial $[\phi_1(t), \phi_2(t), \phi_3(t)] = [1, t, t^2]$ e substitua $f(t)$ por $\phi_k(t)$ obtendo

$$\int_0^h 1\,dt = h = w_1(1) + w_2(1) + w_3(1) \tag{9.102}$$

$$\int_0^h t\,dt = h^2/2 = w_1(0) + w_2(h/2) + w_3(h) \tag{9.103}$$

$$\int_0^h t^2\,dt = h^3/3 = w_1(0)^2 + w_2(h/2)^2 + w_3(h)^2 \tag{9.104}$$

que pode ser escrito na forma matricial

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & h/2 & h \\ 0 & h^2/4 & h^2 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} h \\ h^2/2 \\ h^3/3 \end{bmatrix} \tag{9.105}$$

Note que podemos simplificar h tal que o sistema fique

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1/2 & 1 \\ 0 & 1/4 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = h \begin{bmatrix} 1 \\ 1/2 \\ 1/3 \end{bmatrix} \tag{9.106}$$

Resolvendo o sistema, obtemos $(w_1, w_2, w_3) = h\left(\frac{1}{6}, \frac{4}{6}, \frac{1}{6}\right)$, o que fornece a regra de Simpson:

$$\int_0^h f(t)\, dt \approx \frac{h}{6} f(0) + \frac{4h}{6} f(h/2) + \frac{h}{6} f(h). \tag{9.107}$$

$\Diamond$

Exercícios resolvidos

Esta seção carece de exercícios resolvidos. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

Exercícios

Esta seção carece de exercícios. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

9.4 Regras compostas

Em todas as estimativas de erro que derivamos, o erro depende do tamanho do intervalo de integração. Uma estratégia para reduzir o erro consiste em particionar

o intervalo de integração em diversos subintervalos menores de forma que

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} \int_{x_i}^{x_{i+1}} f(x)\,dx \tag{9.108}$$

onde $a = x_1 < ... < x_{n+1} = b$, sendo n o número de subintervalos da partição do intervalo de integração. No caso uniforme $x_i = a + (i-1)h$, $h = (b-a)/n$.

Depois, aplica-se um método simples de integração em cada subintervalo,

$$\int_{x_i}^{x_{i+1}} f(x)\,dx \approx \Delta S_i \tag{9.109}$$

e a integral será aproximada por

$$\int_a^b f(x)\,dx \approx S = \sum_{i=1}^{n} \Delta S_i. \tag{9.110}$$

9.4.1 Método composto dos trapézios

A **regra composta dos trapézios** assume a seguinte forma:

$$\begin{aligned} \int_a^b f(x)\,dx &= \sum_{i=1}^{n} \int_{x_i}^{x_{i+1}} f(x)\,dx && (9.111)\\ &\approx \sum_{i=1}^{n} \frac{x_{i+1}-x_i}{2}\left[f(x_i)+f(x_{i+1})\right]. && (9.112) \end{aligned}$$

Como $h = x_{i+1} - x_i$, temos:

$$\begin{aligned} \int_a^b f(x)\,dx &\approx \frac{h}{2}\sum_{k=1}^{N_i}\left[f(x_k)+f(x_{k+1})\right] && (9.113)\\ &= \frac{h}{2}\left[f(x_1)+2f(x_2)+2f(x_3)+\cdots+2f(x_{N_i})+f(x_{N_i+1})\right] && (9.114)\\ &= \frac{h}{2}\left[f(x_1)+f(x_{N_i+1})\right]+h\sum_{i=2}^{N_i} f(x_i) && (9.115) \end{aligned}$$

9.4.2 Código Python: trapézio composto

O código `Python` abaixo é uma implementação do método do trapézio composto para calcular:

$$\int_a^b f(x)\,dx = \frac{h}{2}\left[f(x_1)+f(x_{n+1})\right]+h\sum_{i=2}^{n} f(x_i)+O(h^3), \tag{9.116}$$

onde $h = (b-a)/n$ e $x_i = a+(i-1)h$, $i = 1,2,\ldots,n+1$. Os parâmetros de entrada são: `f` o integrando definido como uma função, `a` o limite inferior de integração, `b` o limite superior de integração, `n` o número de subintervalos desejado. A variável de saída é `y` e corresponde a aproximação calculada de $\int_a^b f(x)\,dx$.

Aqui, cabe um código Python explicativo. Escreva você mesmo o código.

Veja como participar da escrita do livro em:

https://github.com/livroscolaborativos/CalculoNumerico

9.4.3 Método composto de Simpson

Já a regra composta de Simpson assume a seguinte forma:

$$\int_a^b f(x)\,dx = \sum_{k=1}^{n} \int_{x_k}^{x_{k+1}} f(x)\,dx \tag{9.117}$$

$$\approx \sum_{k=1}^{n} \frac{x_{x+1}-x_k}{6}\left[f(x_k) + 4f\left(\frac{x_{k+1}+x_k}{2}\right) + f(x_{k+1})\right] \tag{9.118}$$

onde, como anteriormente, $x_k = a+(k-1)h$, $h = (b-a)/n$ e $i = 1,2,\ldots,n+1$, sendo n o número de subintervalos da partição do intervalo de integração. Podemos simplificar o somatório acima, escrevendo:

$$\int_a^b f(x)\,dx \approx \frac{h}{3}\left[f(x_1) + 2\sum_{i=1}^{n-1} f(x_{2i+1}) + 4\sum_{i=1}^{n} f(x_{2i}) + f(x_{2n+1})\right] + O(h^5) \tag{9.119}$$

onde, agora, $h = (b-a)/(2n)$, $x_i = a+(i-1)h$, $i = 1,2,\ldots,2n+1$.

9.4.4 Código em Python: Simpson composto

O código em `GNU Octave` abaixo é uma implementação do método de Simpson composto para calcular:

$$\int_a^b f(x)\,dx = \frac{h}{3}\left[f(x_1) + 2\sum_{i=1}^{n-1} f(x_{2i+1}) + 4\sum_{i=1}^{n} f(x_{2i}) + f(x_{2n+1})\right] + O(h^3), \tag{9.120}$$

onde $h = (b-a)/(2n)$ e $x_i = a+(i-1)h$, $i = 1,2,\ldots,2n+1$. Os parâmetros de entrada são: `f` o integrando definido como uma função, `a` o limite inferior de integração, `b` o limite superior de integração, `n` o número de subintervalos desejado. A variável de saída é `y` e corresponde a aproximação calculada de $\int_a^b f(x)\,dx$.

Aqui, cabe um código Python explicativo. Escreva você mesmo o código.

Veja como participar da escrita do livro em:

https://github.com/livroscolaborativos/CalculoNumerico

Exemplo 9.4.1. Calcule numericamente a integral

$$\int_0^2 x^2 e^{x^2}\, dx \tag{9.121}$$

pelas regras compostas do ponto médio, trapézio e Simpson variando o número de intervalos $n = 1$, 2, 3, 6, 12, 24, 48 e 96.

Solução. As aproximações calculadas são apresentadas na seguinte tabela:

n	Ponto médio	Trapézios	Simpson
1	5,4365637	218,3926	76,421909
2	21,668412	111,91458	51,750469
3	31,678746	80,272022	47,876505
6	41,755985	55,975384	46,495785
12	45,137529	48,865685	46,380248
24	46,057757	47,001607	46,372373
48	46,292964	46,529682	46,37187
96	46,352096	46,411323	46,371838

◊

Exercícios resolvidos

Esta seção carece de exercícios resolvidos. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

Exercícios

E 9.4.1. Use as rotinas computacionais para calcular numericamente o valor das seguintes integrais usando o método composto dos trapézios para os seguintes números de pontos:

n	$\int_0^1 e^{-4x^2}\,dx$	$\int_0^1 \frac{1}{1+x^2}dx$	$\int_0^1 x^4(1-x)^4\,dx$	$\int_0^1 e^{-\frac{1}{x^2+1}}\,dx$
17	0,4409931			
33	0,4410288			
65	0,4410377			
129	0,4410400			
257	0,4410405			
513	0,4410406			
1025	0,4410407	0,7853981	1,5873015873016E3	4,6191723776309E3

E 9.4.2. O valor exato da integral imprópria $\int_0^1 x\ln(x)\,dx$ é dado por

$$\int_0^1 x\ln(x)\,dx = \left(\frac{x^2}{2}\ln x - \frac{x^2}{4}\right)\Bigg|_0^1 = -1/4. \tag{9.122}$$

Aproxime o valor desta integral usando a regra de Simpson para $n = 3$, $n = 5$ e $n = 7$. Como você avalia a qualidade do resultado obtido? Por que isso acontece.

E 9.4.3. O valor exato da integral imprópria $\int_0^\infty e^{-x^2}\,dx$ é dado por $\frac{\sqrt{\pi}}{2}$. Escreva esta integral como

$$I = \int_0^1 e^{-x^2}\,dx + \int_0^1 u^{-2}e^{-1/u^2}du = \int_0^1 \left(e^{-x^2} + x^{-2}e^{-1/x^2}\right)\,dx \tag{9.123}$$

e aproxime seu valor usando o esquema de trapézios e Simpson para $n = 5$, $n = 7$ e $n = 9$.

E 9.4.4. Estamos interessados em avaliar numericamente a seguinte integral:

$$\int_0^1 \ln(x)\,\mathrm{sen}\,(x)\,dx \tag{9.124}$$

cujo valor com 10 casas decimais corretas é $-.2398117420$.

a) Aproxime esta integral via Gauss-Legendre com $n = 2$, $n = 3$, $n = 4$, $n = 5$, $n = 6$ e $n = 7$.

b) Use a identidade

$$\int_0^1 \ln(x)\,\text{sen}\,(x)\;dx = \int_0^1 \ln(x)x\;dx + \int_0^1 \ln(x)\left[\text{sen}\,(x)-x\right]\;dx \quad (9.125)$$

$$= \left(\frac{x^2}{2}\ln x - \frac{x^2}{4}\right)\Big|_0^1 + \int_0^1 \ln(x)\left[\text{sen}\,(x)-x\right]\;dx \quad (9.126)$$

$$= -\frac{1}{4} + \int_0^1 \ln(x)\left[\text{sen}\,(x)-x\right]\;dx \quad (9.127)$$

e aproxime a integral $\int_0^1 \ln(x)\left[\text{sen}\,(x)-x\right]\;dx$ numericamente via Gauss-Legendre com $n=2$, $n=3$, $n=4$, $n=5$, $n=6$ e $n=7$.

c) Compare os resultados e discuta levando em consideração as respostas às seguintes perguntas: 1)Qual função é mais bem-comportada na origem? 2)Na segunda formulação, qual porção da solução foi obtida analiticamente e, portanto, sem erro de truncamento?

9.5 Método de Romberg

O método de Romberg é um algoritmo projetado para construir quadraturas de alta ordem de forma iterativa a partir do método dos trapézios.

Considere o método de trapézios composto aplicado à integral

$$\int_a^b f(x)\;dx. \quad (9.128)$$

Defina $I(h)$ a aproximação desta integral pelo método dos trapézios composto com malha de largura constante igual a h. Aqui $h=\frac{b-a}{N_i}$ para algum N_i inteiro, isto é:

$$I(h) = \frac{h}{2}\left[f(a) + 2\sum_{j=2}^{N_i} f(x_j) + f(b)\right], \quad N_i = \frac{b-a}{h} \quad (9.129)$$

Teorema 9.5.1. *Se $f(x)$ é uma função analítica no intervalo (a,b), então a função $I(h)$ admite uma representação na forma*

$$I(h) = I_0 + I_2h^2 + I_4h^4 + I_6h^6 + \ldots \quad (9.130)$$

Para um demonstração, veja [4]. Em especial observamos que

$$\int_a^b f(x)\;dx = \lim_{h\to 0} I(h) = I_0 \quad (9.131)$$

Ou seja, o valor exato da integral procurada é dado pelo coeficiente I_0.

A ideia central do método de Romberg, agora, consiste em usar a extrapolação de Richardson para construir métodos de maior ordem a partir do métodos dos trapézios para o intervalo (a,b)

Exemplo 9.5.1. Construção do método de quarta ordem.

$$I(h) = I_0 + I_2 h^2 + I_4 h^4 + I_6 h^6 + \dots \tag{9.132}$$

$$\tag{9.133}$$

$$I\left(\frac{h}{2}\right) = I_0 + I_2\frac{h^2}{4} + I_4\frac{h^4}{16} + I_6\frac{h^6}{64} + \dots \tag{9.134}$$

$$\tag{9.135}$$

Usamos agora uma eliminação gaussiana para obter o termo I_0:

$$\frac{4I(h/2) - I(h)}{3} = I_0 - \frac{1}{4}I_4 h^4 - \frac{5}{16}I_6 h^6 + \dots \tag{9.136}$$

Vamos agora aplicar a fórmula para $h = b - a$,

$$I(h) = \frac{h}{2}\left[f(a) + f(b)\right] \tag{9.137}$$

$$I(h/2) = \frac{h}{4}\left[f(a) + 2f\left(c\right) + f(b)\right], \quad c = \frac{a+b}{2}. \tag{9.138}$$

$$\tag{9.139}$$

$$\frac{4I(h/2) - I(h)}{3} = \frac{h}{3}\left[f(a) + 2f\left(c\right) + f(b)\right] - \frac{h}{6}\left[f(a) + f(b)\right] \tag{9.140}$$

$$= \frac{h}{6}\left[f(a) + 4f\left(c\right) + f(b)\right]. \tag{9.141}$$

Note que este esquema obtido coincide com o método de Simpson.

A partir de agora, a fim de deduzir o caso geral, utilizaremos a seguinte notação:

$$R_{1,1} = I(h), \tag{9.142}$$

$$R_{2,1} = I(h/2), \tag{9.143}$$

$$R_{3,1} = I(h/4), \tag{9.144}$$

$$\vdots \tag{9.145}$$

$$R_{n,1} = I(h/2^{n-1}). \tag{9.146}$$

Observamos que os pontos envolvidos na quadratura $R_{k,1}$ são os mesmos pontos envolvidos na quadratura $R(k-1,1)$ acrescidos dos pontos centrais, assim, temos a seguinte fórmula de recorrência:

$$R_{k,1} = \frac{1}{2}R_{k-1,1} + \frac{h}{2^{k-1}} \sum_{i=1}^{2^{k-2}} f\left(a + (2i-1)\frac{h}{2^{k-1}}\right) \tag{9.147}$$

Definimos $R_{k,2}$ para $k \geq 2$ como o esquema de ordem quatro obtido da fórmula do Exemplo 9.5.1:

$$R_{k,2} = \frac{4R_{k,1} - R_{k-1,1}}{3} \tag{9.148}$$

Os valores $R_{k,2}$ representam então os valores obtidos pelo método de Simpson composto aplicado a uma malha composta de $2^{k-1}+1$ pontos.

Similarmente os valores de $R_{k,j}$ são os valores obtidos pela quadratura de ordem $2j$ obtida via extrapolação de Richardson. Pode-se mostrar que

$$R_{k,j} = R_{k,j-1} + \frac{R_{k,j-1} - R_{k-1,j-1}}{4^{j-1}-1}. \tag{9.149}$$

Exemplo 9.5.2. Construa o esquema de Romberg para aproximar o valor de $\int_0^2 e^{x^2}\,dx$ com erro de ordem 8.

O que nos fornece os seguintes resultados:

55,59815	0,000000	0,000000	0,000000
30,517357	22,157092	0,000000	0,000000
20,644559	17,353626	17,033395	0,000000
17,565086	16,538595	16,484259	**16,475543**

Ou seja, temos:

$$\int_0^2 e^{x^2}\,dx \approx 16{,}475543 \tag{9.150}$$

usando uma aproximação de ordem 8.

Exemplo 9.5.3. Construa o esquema de Romberg para aproximar o valor de $\int_0^2 x^2 e^{x^2}\,dx$ com erro de ordem 12.

O que nos fornece:

218,3926					
111,91458	76,421909				
66,791497	51,750469	50,105706			
51,892538	46,926218	46,604601	46,549028		
47,782846	46,412949	46,378731	46,375146	46,374464	
46,72661	46,374531	46,37197	46,371863	46,37185	**46,371847**

Ou seja, temos:

$$\int_0^2 x^2 e^{x^2}\, dx \approx 46{,}371847 \tag{9.151}$$

com uma aproximação de ordem 12.

Exercícios

E 9.5.1. Para cada integrando, encontre o função $I(h) = a_0 + a_1 h + a_2 h^2 + a_3 h^3 + a_4 h^4$ que melhor se ajusta aos dados, onde $h = \frac{1}{n-1}$. Discuta os resultados com base no teorema envolvido na construção do método de Romberg.

E 9.5.2. Calcule os valores da quadratura de Romberg de $R_{1,1}$ até $R_{4,4}$ para $\int_0^\pi \operatorname{sen}(x)\, dx$. Não use rotinas prontas neste problema.

E 9.5.3. Sem usar rotinas prontas, use o método de integração de Romberg para obter a aproximação $R_{3,3}$ das seguintes integrais:

a) $\int_0^1 e^{-x^2}\, dx$

b) $\int_0^2 \sqrt{2 - \cos(x)}\, dx$

c) $\int_0^2 \frac{1}{\sqrt{2-\cos(x)}}\, dx$

E 9.5.4. Encontre uma expressão para $R_{2,2}$ em termos de $f(x)$ e verifique o método de Romberg $R_{2,2}$ é equivalente ao método de Simpson.

E 9.5.5. Considere o problema de aproximar numericamente o valor de

$$\int_0^{100} \left(e^{\frac{1}{2}\cos(x)} - 1\right)\, dx \tag{9.156}$$

pelo método de Romberg. Usando rotinas prontas, faça o que se pede.

a) Calcule $R(6,k)$, $k = 1,\ldots,6$ e observe os valores obtidos.

b) Calcule $R(7,k)$, $k = 1,\ldots,6$ e observe os valores obtidos.

c) Calcule $R(8,k)$, $k = 1,\ldots,6$ e observe os valores obtidos.

d) Discuta os resultados anteriores e proponha uma estratégia mais eficiente para calcular o valor da integral.

9.6 Ordem de precisão

Todos os métodos de quadratura que vimos até o momento são da forma

$$\int_a^b f(x)\, dx \approx \sum_{j=1}^{N} w_j f(x_j) \tag{9.157}$$

Exemplo 9.6.1. a) Método do trapézio

$$\begin{aligned}
\int_a^b f(x)\, dx &\approx [f(a) + f(b)]\, \frac{b-a}{2} && (9.158)\\
&= \frac{b-a}{2} f(a) + \frac{b-a}{2} f(b) && (9.159)\\
&:= w_1 f(x_1) + w_2 f(x_2) = \sum_{j=1}^{2} w_j f(x_j) && (9.160)
\end{aligned}$$

b) Método do trapézio com dois intervalos

$$\begin{aligned}
\int_a^b f(x)\, dx &\approx \left[f(a) + 2f\left(\frac{a+b}{2}\right) + f(b)\right] \frac{b-a}{4} && (9.161)\\
&= \frac{b-a}{4} f(a) + \frac{b-a}{2} f\left(\frac{a+b}{2}\right) + \frac{b-a}{4} f(b) && (9.162)\\
&:= w_1 f(x_1) + w_2 f(x_2) + w_3 f(x_3) = \sum_{j=1}^{3} w_j f(x_j) && (9.163)
\end{aligned}$$

c) Método de Simpson

$$\int_a^b f(x)\,dx \approx \left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right]\frac{b-a}{6} \tag{9.164}$$

$$= \frac{b-a}{6}f(a)+\frac{2(b-a)}{3}f\left(\frac{a+b}{2}\right)+\frac{b-a}{6}f(b) \tag{9.165}$$

$$:= \sum_{j=1}^{3} w_j f(x_j) \tag{9.166}$$

d) Método de Simpson com dois intervalos

$$\int_a^b f(x)\,dx \approx \left[f(a)+4f\left(\frac{3a+b}{4}\right)+2f\left(\frac{a+b}{2}\right)\right. \tag{9.167}$$

$$+ \left. 4f\left(\frac{a+3b}{4}\right)+f(b)\right]\frac{b-a}{12} \tag{9.168}$$

$$= \frac{b-a}{12}f(a)+\frac{b-a}{3}f\left(\frac{3a+b}{4}\right)+\frac{b-a}{6}f\left(\frac{a+b}{2}\right) \tag{9.169}$$

$$+ \frac{b-a}{3}f\left(\frac{a+3b}{4}\right)+\frac{b-a}{12}f(b) \tag{9.170}$$

$$:= \sum_{j=1}^{5} w_j f(x_j) \tag{9.171}$$

A principal técnica que temos usado para desenvolver os métodos numéricos é o **polinômio de Taylor**:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n + R_n(x) \tag{9.172}$$

Integrando termo a termo, temos:

$$\int_a^b f(x)\,dx = \int_a^b a_0\,dx + \int_a^b a_1 x\,dx + \int_a^b a_2 x^2\,dx + \ldots + \tag{9.173}$$

$$\int_a^b a_n x^n\,dx + \int_a^b R_n(x)\,dx \tag{9.174}$$

$$= a_0(b-a) + a_1\frac{b^2-a^2}{2} + a_2\frac{b^3-a^3}{3} + \ldots + \tag{9.175}$$

$$a_n\frac{b^{n+1}-a^{n+1}}{n+1} + \int_a^b R_n(x)\,dx \tag{9.176}$$

Neste momento, é natural investigar o desempenho de um esquema numérico aplicado a funções do tipo $f(x) = x^n$.

Definição 9.6.1. *A* **ordem de precisão** *ou* **ordem de exatidão** *de um esquema de quadratura numérica é definida como o maior inteiro positivo* **n** *para o qual o esquema é exato para todas as funções do tipo* x^k *com* $0 \leq k \leq n$*, ou seja, um esquema é dito de ordem n se*

$$\sum_{j=1}^{n} w_j f(x_j) = \int_a^b f(x)\,dx, \quad f(x) = x^k,\ k = 0,1,\ldots n \tag{9.177}$$

ou, equivalentemente:

$$\sum_{j=1}^{n} w_j x_j^k = \int_a^b x^k\,dx = \frac{b^{k+1} - a^{k+1}}{k+1}, \quad k = 0,1,\ldots n \tag{9.178}$$

Observação 9.6.1. Se o método tem ordem 0 ou mais, então

$$\sum_{j=1}^{n} w_j = b - a \tag{9.179}$$

Exemplo 9.6.2. A ordem de precisão do esquema de trapézios é 1:

$$\int_a^b f(x)\,dx \approx [f(a) + f(b)]\,\frac{b-a}{2} = \sum_{j=1}^{2} w_j f(x_j) \tag{9.180}$$

onde $w_j = \frac{b-a}{2}$, $x_1 = a$ e $x_2 = b$.

$$(k=0): \quad \textstyle\sum_{j=1}^{n} w_j = b - a \tag{9.181}$$

$$(k=1): \quad \textstyle\sum_{j=1}^{n} w_j x_j = (a+b)\frac{b-a}{2} = \frac{b^2-a^2}{2} \tag{9.182}$$

$$(k=2): \quad \textstyle\sum_{j=1}^{n} w_j x_j^2 = (a^2+b^2)\frac{b-a}{2} \neq \frac{b^3-a^3}{3} \tag{9.183}$$

Exemplo 9.6.3. A ordem de precisão do esquema de Simpson é 3:

$$\int_a^b f(x)\,dx \approx \left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right]\frac{b-a}{6} = \sum_{j=1}^{3} w_j f(x_j) \tag{9.184}$$

onde $w_1 = w_3 = \frac{b-a}{6}$, $w_2 = 4\frac{b-a}{6}$, $x_1 = a$, $x_2 = \frac{a+b}{2}$ e $x_3 = b$

$$(k=0): \quad \textstyle\sum_{j=1}^{n} w_j = (1+4+1)\frac{b-a}{6} = b - a \tag{9.185}$$

$$(k=1): \quad \textstyle\sum_{j=1}^{n} w_j x_j = (a + 4\frac{a+b}{2} + b)\frac{b-a}{6} = (a+b)\frac{b-a}{2} = \frac{b^2-a^2}{2} \tag{9.186}$$

$$(k=2): \quad \textstyle\sum_{j=1}^{n} w_j x_j^2 = (a^2 + 4\left(\frac{a+b}{2}\right)^2 + b^2)\frac{b-a}{6} = \frac{b^3-a^3}{3} \tag{9.187}$$

$$(k=3): \quad \textstyle\sum_{j=1}^{n} w_j x_j^3 = (a^3 + 4\left(\frac{a+b}{2}\right)^3 + b^3)\frac{b-a}{6} = \frac{b^4-a^4}{4} \tag{9.188}$$

$$(k=4): \quad \textstyle\sum_{j=1}^{n} w_j x_j^4 = (a^4 + 4\left(\frac{a+b}{2}\right)^4 + b^4)\frac{b-a}{6} \neq \frac{b^5-a^5}{4} \tag{9.189}$$

Exemplo 9.6.4. Encontre os pesos w_j e as abscissas x_j tais que o esquema de dois pontos

$$\int_{-1}^{1} f(x)\,dx = w_1 f(x_1) + w_2 f(x_2) \tag{9.190}$$

é de ordem 3.

Solução. Temos um sistema de quatro equações e quatro incógnitas dado por:

$$\begin{aligned} w_1 + w_2 &= 2 && (9.191)\\ x_1 w_1 + x_2 w_2 &= 0 && (9.192)\\ x_1^2 w_1 + x_2^2 w_2 &= \frac{2}{3} && (9.193)\\ x_1^3 w_1 + x_2^3 w_2 &= 0 && (9.194)\\ & && (9.195) \end{aligned}$$

Da segunda e quarta equação, temos:

$$\frac{w_1}{w_2} = -\frac{x_2}{x_1} = -\frac{x_2^3}{x_1^3} \tag{9.196}$$

Como $x_1 \neq x_2$, temos $x_1 = -x_2$ e $w_1 = w_2$. Da primeira equação, temos $w_1 = w_2 = 1$. Da terceira equação, temos $-x_1 = x_2 = \frac{\sqrt{3}}{3}$.

Esse esquema de ordem de precisão três e dois pontos chama-se quadratura de Gauss-Legendre com dois pontos:

$$\int_{-1}^{1} f(x)\,dx = f\left(\frac{\sqrt{3}}{3}\right) + f\left(-\frac{\sqrt{3}}{3}\right) \tag{9.197}$$

$\Diamond$

Exemplo 9.6.5. Comparação

$f(x)$	Exato	Trapézio	Simpson	Gauss-Le
e^x	$e - e^{-1}$ $\approx 2{,}35040$	$e^{-1} + e$ $\approx 3{,}08616$	$\frac{e^{-1} + 4e^0 + e^1}{3}$ $\approx 2{,}36205$	$e^{-\frac{-\sqrt{}}{3}}$ $\approx 2{,}$
$x^2\sqrt{3+x^3}$	$\frac{16}{9} - \frac{4}{9}\sqrt{2}$ $\approx 1{,}14924$	3,41421	1,13807	1,1
$x^2e^{x^3}$	$\frac{e-e^{-1}}{3} \approx 0{,}78347$	3,08616	1,02872	0,6

Exercícios

E 9.6.1. Encontre os pesos w_1, w_2 e w_3 tais que o esquema de quadratura dado por

$$\int_0^1 f(x)\,dx \approx w_1 f(0) + w_2 f(1/2) + w_3 f(1) \tag{9.198}$$

apresente máxima ordem de exatidão. Qual a ordem obtida?

E 9.6.2. Encontre a ordem de exatidão do seguinte método de integração:

$$\int_{-1}^1 f(x)\,dx \approx \frac{2}{3}\left[f\left(\frac{-\sqrt{2}}{2}\right) + f(0) + f\left(\frac{\sqrt{2}}{2}\right)\right] \tag{9.199}$$

E 9.6.3. Encontre a ordem de exatidão do seguinte método de integração:

$$\int_{-1}^1 f(x)\,dx = -\frac{1}{210}f'(-1) + \frac{136}{105}f(-1/2) - \frac{62}{105}f(0) + \frac{136}{105}f(1/2) + \frac{1}{210}f'(1) \tag{9.200}$$

E 9.6.4. Encontre os pesos w_1, w_2 e w_3 tal que o método de integração

$$\int_0^1 f(x)\,dx \approx w_1 f(1/3) + w_2 f(1/2) + w_3 f(2/3) \tag{9.201}$$

tenha ordem de exatidão máxima. Qual é ordem obtida?

E 9.6.5. Quantos pontos são envolvidos no esquema de quadratura $R_{3,2}$? Qual a ordem do erro deste esquema de quadratura? Qual a ordem de exatidão desta quadratura?

9.7 Quadratura de Gauss-Legendre

Utilizando n pontos para aproximar a integral de $f(x)$ em $[-1,1]$ podemos encontrar a regra de quadratura de Gauss-Legendre

$$\int_{-1}^{1} f(t)\, dt \approx \sum_{j=1}^{n} w_j f(t_j) \tag{9.202}$$

cuja ordem de exatidão é $2n-1$.

- Note que temos n coeficientes w_j e n pontos t_j para determinar. O problema de encontrar os n pesos e n abscissas é equivalente a um sistema não linear com $2n$ equações e $2n$ incógnitas.
- Pode-se mostrar que este problema sempre tem solução e que a solução é única se $t_1 < t_2 < \ldots < t_n$
- Os nós x_j são dados pelos zeros do polinômio de Legendre, $P_n(t)$.
- Os pesos são dados por

$$w_j = \frac{2}{\left(1-t_j^2\right)\left[P_n'(t_j)\right]^2}. \tag{9.203}$$

A Tabela 9.1 lista os nós e os pesos da quadratura de Gauss-Legendre para $n = 1$, 2, 3, 4 e 5.

Exemplo 9.7.1. Aproxime

$$I = \int_{-1}^{1} \sqrt{1+x^2}\, dx \tag{9.204}$$

pelo método de Gauss-Legendre com 2, 3, 4 e 5 pontos.

Solução. A aproximação desta integral usando o método de Gauss-Legendre consiste em computar

$$I = \int_{-1}^{1} f(x)\, dx \approx \sum_{i=1}^{n} w_i f(t_i), \tag{9.205}$$

onde $f(x) = \sqrt{1+x^2}$, w_i é o i-ésimo peso, t_i é o i-ésimo nodo, $i = 1, \ldots, n$, e n é o número de nodos (ou pesos) da quadratura. Usando os nodos e pesos dados na Tabela 9.1, obtemos os seguintes resultados:

Tabela 9.1: Nodos e pesos para quadratura de Gauss-Legendre.

n	t_j	w_j
1	0	2
2	$\pm\frac{\sqrt{3}}{3}$	1
3	0	$\frac{8}{9}$
	$\pm\sqrt{\frac{3}{5}}$	$\frac{5}{9}$
4	$\pm\sqrt{\left(3-2\sqrt{6/5}\right)/7}$	$\frac{18+\sqrt{30}}{36}$
	$\pm\sqrt{\left(3+2\sqrt{6/5}\right)/7}$	$\frac{18-\sqrt{30}}{36}$
5	0	$\frac{128}{225}$
	$\pm\frac{1}{3}\sqrt{5-2\sqrt{\frac{10}{7}}}$	$\frac{322+13\sqrt{70}}{900}$
	$\pm\frac{1}{3}\sqrt{5+2\sqrt{\frac{10}{7}}}$	$\frac{322-13\sqrt{70}}{900}$

n	I
2	2,3094011
3	2,2943456
4	2,2957234
5	2,2955705

Em Python, temos:

```
def f(x):
    return np.sqrt(1+x**2)

#G-L n=2
x2 = [np.sqrt(3)/3]
w2 = [1]
I2 = w2[0]*f(x2[0]) + w2[0]*f(-x2[0])
print("Para n = 2, I = %1.7f" % I2)

#G-L n=3
x3 = [0, np.sqrt(3/5)]
w3 = [8/9, 5/9]
I3 = (w3[0]*f(x3[0]) +
      w3[1]*f(x3[1]) + w3[1]*f(-x3[1]))
print("Para n = 3, I = %1.7f" % I3)

#G-L n=4
x4 = [np.sqrt((3-2*np.sqrt(6/5))/7),
      np.sqrt((3+2*np.sqrt(6/5))/7)]
w4 = [(18+np.sqrt(30))/36, (18-np.sqrt(30))/36]
I4 = (w4[0]*f(x4[0]) + w4[0]*f(-x4[0]) +
      w4[1]*f(x4[1]) + w4[1]*f(-x4[1]))
print("Para n = 4, I = %1.7f" % I4)

#G-L n=5
x5 = [0,
      1/3*np.sqrt(5-2*np.sqrt(10/7)),
      1/3*np.sqrt(5+2*np.sqrt(10/7))]
w5 = [128/225, (322+13*np.sqrt(70))/900,
      (322-13*np.sqrt(70))/900]
I5 = (w5[0]*f(x5[0]) +
      w5[1]*f(x5[1]) + w5[1]*f(-x5[1]) +
      w5[2]*f(x5[2]) + w5[2]*f(-x5[2]))
```

```
print("Para n = 5, I = %1.7f" % I5)
```

$\Diamond$

Mudança de intervalo

Os coeficientes da quadratura de Gauss-Legendre foram obtidos no intervalo $[-1,1]$. Para aproximar a integral de $f(x)$ no intervalo $[a,b]$ devemos fazer a mudança de variável

$$\bar{x}_i = \alpha t_i + \beta, \quad \alpha = (b-a)/2, \quad \beta = (b+a)/2 \tag{9.206}$$

tal que

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{n} w_i f(\bar{x}_i)(b-a)/2 \tag{9.207}$$

Quando subdividimos o intervalo inicial $[a,b]$ em N intervalos com extremos $[x_i,x_{i+1}]$ a transformação torna-se

$$\bar{x}_i = \alpha t_i + \beta, \quad \alpha = (x_{i+1}-x_i)/2, \quad \beta = (x_{i+1}+x_i)/2 \tag{9.208}$$

e

$$\int_{x_i}^{x_{i+1}} f(x)\,dx \approx \sum_{i=1}^{n} w_i f(\bar{x}_i)(x_{i+1}-x_i)/2 \tag{9.209}$$

Exemplo 9.7.2. Aproximar

$$I = \int_0^1 \sqrt{1+x^2}dx \tag{9.210}$$

pelo método de Gauss-Legendre com 3 pontos.

Solução. Para tanto, fazemos a mudança de variáveis $u = 2x-1$:

$$\begin{aligned} I &= \int_0^1 \sqrt{1+x^2}dx \\ &= \frac{1}{2}\int_{-1}^1 \sqrt{1+\left(\frac{u+1}{2}\right)^2}du. \end{aligned} \tag{9.211}$$

E, então aplicamos a quadratura gaussiana nesta última integral, o que nos fornece $I \approx 1{,}1478011$. Em `Python`, podemos computar estas aproximações com o seguinte código:

```
def f(u):
    return np.sqrt(1+(u+1)**2/4)/2

#G-L n=3
x3 = [0, np.sqrt(3/5)]
w3 = [8/9, 5/9]
I3 = (w3[0]*f(x3[0])
      + w3[1]*f(x3[1]) + w3[1]*f(-x3[1]))
print("Para n = 3, I = %1.7f" %(I3))
```

◊

Exercícios resolvidos

Esta seção carece de exercícios resolvidos. Participe da sua escrita.
Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

Exercícios

E 9.7.1. Encontre aproximações para a integral

$$\int_{-1}^{1} x^4 e^{x^5} dx \tag{9.212}$$

usando a quadratura de Gauss-Legendre com 2, 3, 4 e 5 pontos. Então, compare com o seu valor exato.

E 9.7.2. Encontre aproximações para as seguintes integrais via Gauss-Legendre com 4 e 5 pontos:

a) $\int_0^1 e^{-x^4} dx$

b) $\int_1^4 \log(x + e^x) dx$

c) $\int_0^1 e^{-x^2} dx$

9.8 Integrais impróprias

A aplicação de quadraturas numéricas para integrais impróprias geralmente demanda alguns cuidados adicionais. Aqui, abordaremos apenas alguns aspectos, começando por integrandos com singularidade no intervalo de integração.

9.8.1 Integrandos com singularidade do tipo $1/(x-a)^n$

Consideremos a integral imprópria[3]

$$\int_a^b \frac{f(x)}{(x-a)^p}\,dx, \quad 0 < p < 1, \tag{9.213}$$

Observamos, que para uma tal integral, não é possível aplicar, diretamente, as regras do trapézio e de Simpson. Alternativamente, podemos aplicar a regra do ponto médio e quadraturas gaussianas, por exemplo. Entretanto, aplicações diretas de tais quadraturas fornecem resultados pouco precisos (veja o Exemplo 9.8.1).

Exemplo 9.8.1. Aplicando as regras compostas do ponto médio e quadratura gaussiana com dois pontos à integral

$$\int_0^1 \frac{e^{-x}}{x^{1/2}}\,dx \tag{9.214}$$

obtemos os seguintes resultados (n número de subintervalos):

n	h	Ponto Médio	G-L(2)
1	1	0,8577	1,1363
10	10^{-1}	1,3007	1,3829
10^2	10^{-2}	1,4331	1,4587
10^3	10^{-3}	1,4745	1,4826
10^4	10^{-4}	1,4876	1,4902

No `Python`, podemos computar os valores apresentados na tabela acima da seguinte forma:

```
def f(x):
    return np.exp(-x)/np.sqrt(x)
def F(u):
    return (x[i+1]-x[i])/2*f((x[i+1]-x[i])/2*(u+1)+x[i])
```

[3] convergente com $f(x)$ suficientemente suave, por hipótese.

```
a = 0
b = 1
n = 10
h = (b-a)/n
x = np.linspace(a,b,n+1)

#regra do ponto medio
s_med = 0;
for i in range(n):
    s_med += f((x[i]+x[i+1])/2)*h

#quadratura gaussiana (2 pontos)
s_gl=0
for i in range(n):
   s_gl += F(np.sqrt(3)/3) + F(-np.sqrt(3)/3)

print('''Para {} subintervalos tem-se:
\tPonto medio =~ {:.4e} e
\tquadratura gaussiada (2 pontos) =~ {:.4e}'''.format(n, s_med, s_gl))
```

Uma estratégia para se computar uma tal integral imprópria

$$I = \int_a^b \frac{f(x)}{(x-a)^p}\,dx \tag{9.215}$$

é reescrevê-la da forma

$$I = \underbrace{\int_a^b \frac{f(x)-p(x)}{(x-a)^n}\,dx}_{I_1} + \underbrace{\int_a^b \frac{p(x)}{(x-a)^n}\,dx}_{I_2} \tag{9.216}$$

onde $p(x)$ é escolhida de forma que a singularidade esteja presente somente em I_2 e esta possa ser calculada de forma analítica, restando computar I_1 numericamente. Isto pode ser feito, escolhendo $p(x)$ como a expansão em polinômio de Taylor da função $f(x)$ em torno do ponto $x = a$, i.e.

$$p(x) = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(m)}(a)}{m!}(x-a)^m \tag{9.217}$$

Com esta escolha, o integrando de I_1 passa a ter uma singularidade removível

$$\lim_{x\to a} \frac{f(x)-p(x)}{(x-a)^p} = 0. \tag{9.218}$$

e pode ser computada numericamente. A integral I_2 pode ser calculada analiticamente, de fato

$$\int_a^b \frac{p(x)}{(x-a)^p}\,dx = \frac{f(a)}{1!(1-p)}(x-a)^{1-p} + \cdots + \frac{f^{(m)}(a)}{m!(m-p)}(x-a)^{m-p}\Big|_a^b. \quad (9.219)$$

Exemplo 9.8.2. Consideremos a integral imprópria

$$I = \int_0^1 \frac{e^{-x}}{\sqrt{x}}\,dx. \quad (9.220)$$

Computando o polinômio de Taylor de grau 4 de $f(x) = e^{-x}$ em torno de $x = 0$, obtemos

$$p(x) = 1 - x + \frac{x^2}{2} - \frac{x^3}{3} + \frac{x^4}{4}. \quad (9.221)$$

Então, escrevemos

$$I = \underbrace{\int_0^1 \frac{e^{-x} - p(x)}{\sqrt{x}}\,dx}_{I_1} + \underbrace{\int_0^1 \frac{p(x)}{\sqrt{x}}\,dx}_{I_2}. \quad (9.222)$$

Calculando I_2 analiticamente, temos

$$\begin{aligned} I_2 &= \int_0^1 \frac{p(x)}{\sqrt{x}}\,dx \\ &= 2x^{1/2} - \frac{2}{3}x^{3/2} + \frac{2}{10}x^{(5/2)} - \frac{2}{42}x^{7/2} + \frac{2}{216}x^{9/2}\Big|_0^1 \\ &= 2 - \frac{2}{3} + \frac{2}{10} - \frac{2}{42} + \frac{2}{216} \\ &= 1{,}4950. \end{aligned} \quad (9.223)$$

Agora, computamos a integral I_1 numericamente usando a regra composta do ponto médio. A seguinte tabela apresenta os resultados para as aproximações obtidas para I_1 e consequentemente para $I = I_1 + I_2$:

n	h	I_2	I
1	1	−3,39657E−4	1,49463
10	10^{-1}	−1,31240E−3	1,49366
10^2	10^{-2}	−1,32515E−3	1,49365
10^3	10^{-3}	−1,32528E−3	1,49365

Exercícios resolvidos

Esta seção carece de exercícios resolvidos. Participe da sua escrita.
Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

Exercícios

Esta seção carece de exercícios. Participe da sua escrita.
Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

9.9 Exercícios finais

E 9.9.1. Considere o problema de calcular numericamente a integral $I = \int_{-1}^{1} f(x)dx$ quando $f(x) = \frac{\cos(x)}{\sqrt{|x|}}$.

a) O que acontece quando se aplica diretamente a quadratura gaussiana com um número impar de abscissas?

b) Calcule o valor aproximado por quadratura gaussiana com $n = 2$, $n = 4$, $n = 6$ e $n = 8$.

c) Calcule o valor aproximado da integral removendo a singularidade

$$I = \int_{-1}^{1} \frac{\cos(x)}{\sqrt{|x|}}dx = \int_{-1}^{1} \frac{\cos(x)-1}{\sqrt{|x|}}dx + \int_{-1}^{1} \frac{1}{\sqrt{|x|}}dx \quad (9.224)$$

$$= \int_{-1}^{1} \frac{\cos(x)-1}{\sqrt{|x|}}dx + 2\int_{0}^{1} \frac{1}{\sqrt{x}}dx = \int_{-1}^{1} \frac{\cos(x)-1}{\sqrt{|x|}}dx + 4 \quad (9.225)$$

e aplicando quadratura gaussiana com $n = 2$, $n = 4$, $n = 6$ e $n = 8$.

d) Calcule o valor aproximado da integral removendo a singularidade, considerando a paridade da função

$$I = 4 + \int_{-1}^{1} \frac{\cos(x)-1}{\sqrt{|x|}}dx = 4 + 2\int_{0}^{1} \frac{\cos(x)-1}{\sqrt{x}}dx = 4 + \sqrt{2}\int_{-1}^{1} \frac{\cos\left(\frac{1+u}{2}\right)-1}{\sqrt{1+u}} \quad (9.226)$$

e aplicando quadratura gaussiana com $n = 2$, $n = 4$, $n = 6$ e $n = 8$.

e) Expandindo a função $\cos(x)$ em série de Taylor, truncando a série depois do n-ésimo termos não nulo e integrando analiticamente.

f) Aproximando a função $\cos(x)$ pelo polinômio de Taylor de grau 4 dado por

$$P_4(x) = 1 - \frac{x^2}{2} + \frac{x^4}{24} \tag{9.227}$$

e escrevendo

$$I = \int_{-1}^{1} \frac{\cos(x)}{\sqrt{|x|}}dx = \int_{-1}^{1} \frac{\cos(x) - P_4(x)}{\sqrt{|x|}}dx + \int_{-1}^{1} \frac{P_4(x)}{\sqrt{|x|}}dx \tag{9.228}$$

$$= \underbrace{2\int_0^1 \frac{\cos(x) - P_4(x)}{\sqrt{x}}dx}_{\text{Resolver numericamente}} + \underbrace{2\int_0^1 \left(x^{-1/2} - \frac{x^{3/2}}{2} + \frac{x^{7/2}}{24}\right)dx}_{\text{Resolver analiticamente}} \tag{9.229}$$

E 9.9.2. Calcule numericamente o valor das seguintes integrais com um erro relativo inferior a 10^{-4}.

a) $\displaystyle\int_0^1 \frac{\operatorname{sen}(\pi x)}{x}dx$

b) $\displaystyle\int_0^1 \frac{\operatorname{sen}(\pi x)}{x(1-x)}dx$

c) $\displaystyle\int_0^1 \frac{\operatorname{sen}\left(\frac{\pi}{2}x\right)}{\sqrt{x(1-x)}}dx$

d) $\displaystyle\int_0^1 \ln(x)\cos(x)dx$

E 9.9.3. Calcule as integrais $\int_0^1 \frac{e^x}{|x|^{1/4}}dx$ e $\int_0^1 \frac{e^{-x}}{|x|^{4/5}}dx$ usando procedimentos analíticos e numéricos.

E 9.9.4. Use a técnica de integração por partes para obter a seguinte identidade envolvendo integrais impróprias:

$$I = \int_0^\infty \frac{\cos(x)}{1+x}dx = \int_0^\infty \frac{\operatorname{sen}(x)}{(1+x)^2}dx. \tag{9.236}$$

Aplique as técnicas estudadas para aproximar o valor de I e explique por que a integral da direita é mais bem comportada.

E 9.9.5. Resolva a equação

$$x + \int_0^x e^{-y^2} dy = 5 \tag{9.237}$$

com 5 dígitos significativos.

E 9.9.6. (Ciência dos materiais) O calor específico (molar) de um sólido pode ser aproximado pela teoria de Debye usando a seguinte expressão

$$C_V = 9Nk_B \left(\frac{T}{T_D}\right)^3 \int_0^{T_D/T} \frac{y^4 e^y}{(e^y - 1)^2} dy \tag{9.238}$$

onde N é a constante de Avogrado dado por $N = 6{,}022 \times 10^{23}$ e k_B é a constante de Boltzmann dada por $k_B = 1{,}38 \times 10^{-23}$. T_D é temperatura de Debye do sólido.

a) Calcule o calor específico do ferro em quando $T = 200K$, $T = 300K$ e $T = 400K$ supondo $T_D = 470K$.

b) Calcule a temperatura de Debye de um sólido cujo calor específico a temperatura de $300K$ é $24J/K/mol$. Dica: aproxime a integral por um esquema numérico com um número fixo de pontos.

c) Melhore sua cultura geral: A lei de Dulong-Petit para o calor específico dos sólidos precede a teoria de Debye. Verifique que a equação de Debye é consistente com Dulong-Petit, ou seja:

$$\lim_{T \to \infty} C_v = 3Nk_B. \tag{9.239}$$

Dica: use $e^y \approx 1 + y$ quando $y \approx 0$

E 9.9.7. Explique por quê quando um método simples tem estimativa de erro de truncamento local de ordem h^n, então o método composto associado tem estimativa de erro de ordem h^{n-1}.

E 9.9.8. Encontre os pesos w_1 e w_2 e as abcissas x_1 e x_2 tais que

$$\int_{-1}^{1} f(x) = w_1 f(x_1) + w_2 f(x_2) \tag{9.240}$$

quando $f(x) = x^k$, $k = 0{,}1{,}2{,}3$, isto é, o método que apresente máxima ordem de exatidão possível com dois pontos.

Use esse método para avaliar o valor da integral das seguintes integrais e compare com os valores obtidos para Simpson e trapézio, bem como com o valor exato.

a) $\int_{-1}^{1} \left(2 + x - 5x^2 + x^3\right) dx$

b) $\int_{-1}^{1} e^x dx$

c) $\int_{-1}^{1} \frac{dx}{\sqrt{x^2+1}}$

E 9.9.9. Encontre os pesos w_1, w_2 e w_3 tal que o método de integração

$$\int_{-1}^{1} f(x)dx \approx w_1 f\left(-\frac{\sqrt{3}}{3}\right) + w_2 f(0) + w_3 f\left(\frac{\sqrt{3}}{3}\right) \tag{9.241}$$

tenha ordem de exatidão máxima. Qual é ordem obtida?

Capítulo 10

Problemas de valor inicial

Neste capítulo, vamos estudar metodologias numéricas para aproximar a solução de problema de valor inicial (problema de valor inicial) para equações diferenciais ordinárias. Primeiramente, daremos atenção aos problemas de primeira ordem e, depois, mostraremos que estas técnicas podem ser estendidas para problemas e sistemas de ordem superior. Considere um problema de valor inicial de primeira ordem dado por:

$$\begin{aligned} u'(t) &= f(t, u(t)), \ \ t > t^{(1)} && \text{(10.1a)} \\ u(t^{(1)}) &= a \ \ \text{(condição inicial).} && \text{(10.1b)} \end{aligned}$$

A incógnita de um problema de valor inicial é uma função que satisfaz a equação diferencial (10.1a) e a condição inicial (10.1b).

Considere os próximos três exemplos:

Exemplo 10.0.1. O seguinte problema é linear não homogêneo:

$$\begin{aligned} u'(t) &= t \\ u(0) &= 2 \end{aligned} \tag{10.2}$$

Exemplo 10.0.2. O seguinte problema é linear homogêneo:

$$\begin{aligned} u'(t) &= u(t) && \text{(10.3)} \\ u(0) &= 1 && \text{(10.4)} \end{aligned}$$

Exemplo 10.0.3. O seguinte problema é não linear e não homogêneo:

$$\begin{aligned} u'(t) &= \operatorname{sen}\left(u(t)^2 + \operatorname{sen}\left(t\right)\right) && \text{(10.5)} \\ u(0) &= a && \text{(10.6)} \end{aligned}$$

A solução do primeiro exemplo é $u(t) = t^2/2 + 2$ pois satisfaz a equação diferencial e a condição inicial. A solução do segundo também é facilmente obtida: $u(t) = e^t$. Porém como podemos resolver o terceiro problema?

Para muitos problemas de valor inicial da forma (10.1), não é possível encontrar uma expressão analítica fechada, ou seja, sabe-se que a solução existe e é única, porém não podemos expressá-la em termos de funções elementares. Por isso é necessário calcular aproximações numéricas para a solução.

Existem uma enorme família de metodologias para construir soluções numéricas para problemas de valor inicial. Aqui, vamos nos limitar a estudar métodos que aproximam $u(t)$ em um conjunto finito de valores de t. Este conjunto de valores será chamado de **malha** e será denotado por $\{t^{(i)}\}_{i=1}^{N} = \{t^{(1)}, t^{(2)}, t^{(3)}, \ldots, t^{(N)}\}$. Desta forma, aproximamos a solução $u(t^{(i)})$ por $u^{(i)}$ em cada ponto da malha usando diferentes esquemas numéricos.

Nos códgos em `Python` apresentados neste capítulo, assumiremos que as seguintes bibliotecas e módulos estão importados:

```
from __future__ import division
import numpy as np
from numpy import linalg
import matplotlib.pyplot as plt
```

10.1 Rudimentos da teoria de problemas de valor inicial

Uma questão fundamental no estudo dos problemas de valor iniciais consiste em analisar se um dado problema é um problema **bem posto**. Ou seja,

- Existe uma solução para o problema de valor inicial?
- A solução é única?
- A solução do problema de valor inicial é pouco sensível a pequenas perturbações nas condições iniciais?

A fim de responder tais questões, precisamos definir o conceito de função Lipschitz contínua, ou simplesmente, função Lipschitz

Definição 10.1.1. *Uma função $f(t,u)$ é Lipschitz contínua em um intervalo I em u se existe uma constante L, tal que $\forall t \in [a,b]$ e $u, v \in \mathbb{R}$,*

$$|f(t,u) - f(t,v)| \leq L|u - v|, \quad \forall t \in I. \tag{10.7}$$

O seguinte resultado estabelece a existência e unicidade de solução para determinada classe de problemas de valor inicial:

Teorema 10.1.1 (Teorema de Picard-Lindelöf)**.** *Seja $f(t,u)$ contínua em t e Lipschitz em u. Então o seguinte problema de valor inicial*

$$\begin{aligned} u'(t) &= f(t,u(t)), \\ u(t^{(1)}) &= a, \end{aligned} \tag{10.8}$$

Admite uma única solução em um intervalo $[t^{(1)},t^{(f)})$ com $t^{(f)} > t^{(1)}$.

Teorema 10.1.2 (Dependência contínua na condição inicial)**.** *Se $u(t)$ e $v(t)$ são soluções do problema de valor inicial (10.8), isto é, com $f(t,u)$ contínua em t e Lipschitz em u Lipschitz com $u(a) = u^{(1)}$, $v(a) = v^{(1)}$, então*

$$|u(t) - v(t)| \leq e^{L(t-t^{(1)})}|u^{(1)} - v_1|. \tag{10.9}$$

Exercícios resolvidos

ER 10.1.1. A função $f(t,u) = \sqrt{u}$, $u \geq 0$ não é uma função Lipschitz em u, pois

$$\lim_{u\to 0+} \frac{|f(t,u) - f(t,0)|}{|u-0|} = \lim_{u\to 0+} \frac{\sqrt{u}}{u} = \lim_{u\to 0+} \frac{1}{\sqrt{u}} = \infty \tag{10.10}$$

Mostre que o seguinte problema de valor inicial não admite solução única:

$$\frac{du}{dt} = \sqrt{u}, \quad u > 0, \tag{10.11}$$

$$u(0) = 0. \tag{10.12}$$

Solução. A função identicamente nula, $u(t) = 0$, satisfaz a equação diferencial e a condição de contorno, logo é uma solução do problema de valor inicial. No entanto, a função[1] $u(t) = \frac{t^2}{4}$ satisfaz a condição inicial, pois $u(0) = 0$ e a equação diferencial pois $\frac{du}{dt} = \frac{t}{2} = \sqrt{\frac{t^2}{4}}$.

De fato, qualquer função do tipo

$$u(t) = \begin{cases} 0, & 0 \leq t \leq t_0 \\ \frac{(t-t_0)^2}{4}, & t > t_0 \end{cases} \tag{10.13}$$

é solução do problema de valor inicial dado. ♢

[1] Esta solução pode ser obtida por separação de variáveis.

Exercícios

Esta seção carece de exercícios. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

10.2 Método de Euler

Nesta seção, contruiremos o mais simples dos métodos para resolver problemas de valor inicial: o método de Euler com passo constante. Por passo constante, queremos dizer que os pontos da malha estão todos igualmente espaçados, isto é:

$$t^{(i)} = (i-1)h, \quad i = 1,2,\ldots,N. \tag{10.14}$$

onde h é passo, ou seja, a distância entre dois pontos da malha.

Considere então o problema de valor inicial dado por:

$$\begin{aligned} u'(t) &= f(t,u(t)), \;\; t > t^{(1)} \\ u(t^{(1)}) &= a. \end{aligned} \tag{10.15}$$

Ao invés de tentar solucionar o problema para qualquer $t > t^{(1)}$, iremos aproximar $u(t)$ em $t = t^{(2)}$.

Integrando (10.15) de $t^{(1)}$ até $t^{(2)}$, obtemos:

$$\int_{t^{(1)}}^{t^{(2)}} u'(t)\,dt = \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt \tag{10.16}$$

$$u(t^{(2)}) - u(t^{(1)}) = \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt \tag{10.17}$$

$$u(t^{(2)}) = u(t^{(1)}) + \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt \tag{10.18}$$

Seja u_n a aproximação de $u(t_n)$. Para obter o método numérico mais simples aproximamos f em $[t^{(1)},t^{(2)}]$ pela função constante $f(t,u(t)) \approx f(t^{(1)},u^{(1)})$,

$$u^{(2)} = u^{(1)} + f(t^{(1)},u^{(1)}) \int_{t^{(1)}}^{t^{(2)}} dt \tag{10.19}$$

$$u^{(2)} = u^{(1)} + f(t^{(1)},u^{(1)})(t^{(2)} - t^{(1)}) \tag{10.20}$$

$$u^{(2)} = u^{(1)} + hf(t^{(1)},u^{(1)}) \tag{10.21}$$

Este procedimento pode ser repetido para $t^{(3)}$, $t^{(4)}$, ..., obtendo, assim, o chamado **método de Euler**:

$$\begin{aligned} u^{(n+1)} &= u^{(n)} + h\, f(t^{(n)},u^{(n)}), \\ u^{(1)} &= u^{(1)} = u(t^{(1)}) \text{ (condição inicial).} \end{aligned} \tag{10.22}$$

Exemplo 10.2.1. Considere o problema de valor inicial

$$\begin{aligned} u'(t) &= 2u(t) \\ u(0) &= 1 \end{aligned} \tag{10.23}$$

cuja solução é $u(t) = e^{2t}$. O método de Euler aplicado a este problema produz o esquema:

$$\begin{aligned} u^{(k+1)} &= u^{(k)} + 2hu^{(k)} = (1+2h)u^{(k)} \\ u^{(1)} &= 1, \end{aligned} \tag{10.24}$$

Suponha que queremos calcular o valor aproximado de $u(1)$ com $h = 0{,}2$. Então os pontos $t^{(1)} = 0$, $t^{(2)} = 0{,}2$, $t^{(3)} = 0{,}4$, $t^{(4)} = 0{,}6$, $t^{(5)} = 0{,}8$ e $t^{(6)} = 1{,}0$ formam os seis pontos da malha. As aproximações para a solução nos pontos da malha usando o método de Euler são:

$$\begin{aligned} u(0) &\approx u^{(1)} = 1 \\ u(0{,}2) &\approx u^{(2)} = (1+2h)u^{(1)} = 1{,}4u^{(1)} = 1{,}4 \\ u(0{,}4) &\approx u^{(3)} = 1{,}4u^{(2)} = 1{,}96 \\ u(0{,}6) &\approx u^{(4)} = 1{,}4u^{(3)} = 2{,}744 \\ u(0{,}8) &\approx u^{(5)} = 1{,}4u^{(4)} = 3{,}8416 \\ u(1{,}0) &\approx u^{(6)} = 1{,}4u^{(5)} = 5{,}37824 \end{aligned} \tag{10.25}$$

Essa aproximação é bem grosseira quando comparamos com a solução do problema em $t = 1$: $u(1) = e^2 \approx 7{,}38906$. Não obstante, se tivéssemos escolhido um passo menor, teríamos obtido uma aproximação melhor. Veja tabela abaixo com valores obtidos com diferentes valores de passo h.

h	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}	10^{-6}	10^{-7}
$u^{(N)}$	6,1917	6,7275	7,0400	7,2096	7,2980	7,3432	7,3660

De fato, podemos mostrar que quando h se aproxima de 0, a solução aproximada via método de Euler converge para a solução exata e^2. Para isto, basta observar que a solução da relação de recorrência (10.24) é dada por

$$u^{(k)} = (1+2h)^{k-1}. \tag{10.26}$$

Como $t^{(k)} = (k-1)h$ e queremos a solução em $t = 2$, a solução aproximada pelo método de Euler com passo h em é dada por:

$$u^{(k)} = (1+2h)^{k-1} = (1+2h)^{\frac{2}{h}}. \tag{10.27}$$

Aplicando o limite $h \to 0+$, temos:

$$\lim_{h\to 0+} (1+2h)^{\frac{2}{h}} = e^2. \tag{10.28}$$

Em Python, podemos computar a solução numérica deste problema de valor inicial via o método de Euler com o seguite código:

```
#define f(t,u)
def f(t,u):
    return 2*u

#tamanho e num. de passos
h = 0.2
N = 6

#cria vetor t e u
t = np.empty(N)
u = np.copy(t)

#C.I.
t[0] = 0
u[0] = 1

#iteracoes
for i in np.arange(N-1):
    t[i+1] = t[i] + h
    u[i+1] = u[i] + h*f(t[i],u[i])

#imprime
for i,tt in enumerate(t):
    print("%1.1f %1.4f" % (t[i],u[i]))
```

Vamos agora, analisar o desempenho do método de Euler usando um exemplo mais complicado, porém ainda simples suficiente para que possamos obter a solução exata:

Exemplo 10.2.2. Considere o problema de valor inicial relacionado à equação logística:

$$\begin{aligned} u'(t) &= u(t)(1-u(t)) \\ u(0) &= 1/2 \end{aligned} \tag{10.29}$$

Podemos obter a solução exata desta equação usando o método de separação de variáveis e o método das frações parciais. Para tal escrevemos:

$$\frac{du(t)}{u(t)(1-u(t))} = dt \tag{10.30}$$

O termo $\frac{1}{u(t)(1-u(t))}$ pode ser decomposto em frações parciais como $\frac{1}{u} + \frac{1}{1-u}$ e chegamos na seguinte equação diferencial:

$$\left(\frac{1}{u(t)} + \frac{1}{1-u(t)}\right) du = dt. \tag{10.31}$$

Integrando termo-a-termo, temos a seguinte equação algébrica relacionando $u(t)$ e t:

$$\ln(u(t)) - \ln(1-u(t)) = t + C \tag{10.32}$$

Onde C é a constante de integração, que é definida pela condição inicial, isto é, $u = 1/2$ em $t = 0$. Substituindo, temos $C = 0$. O que resulta em:

$$\ln\left(\frac{u(t)}{1-u(t)}\right) = t \tag{10.33}$$

Equivalente a

$$\frac{u(t)}{1-u(t)} = e^t \Longrightarrow u(t) = (1-u(t))e^t \Longrightarrow (1+e^t)u(t) = e^t \tag{10.34}$$

E, finalmente, encontramos a solução exata dada por $u(t) = \frac{e^t}{1+e^t}$.

Vejamos, agora, o esquema iterativo produzido pelo método de Euler:

$$\begin{aligned} u^{(k+1)} &= u^{(k)} + hu^{(k)}(1-u^{(k)}), \\ u^{(1)} &= 1/2. \end{aligned} \tag{10.35}$$

O seguinte código pode ser usado para implementar no `Python` a recursão acima:

```
def euler(h,Tmax):
u=.5
   itmax = Tmax/h;
for i in np.arange(itmax):
u = u + h*u*(1-u)
return u


h=1e-1
for t in [.5, 1, 2, 3]:
sol_euler=euler(h,t);
sol_exata=1/(1+np.exp(-t))
erro_relativo = np.fabs((sol_euler-sol_exata)/sol_exata)
```

```
print("h=%1.0e - u(%1.1f) =~ %1.7f - erro_relativo = %1.1e" % (h, t, sol_euler, erro_

h=1e-2
print;
for t in [.5, 1, 2, 3]:
sol_euler=euler(h,t);
sol_exata=1/(1+np.exp(-t))
erro_relativo = np.fabs((sol_euler-sol_exata)/sol_exata)
print("h=%1.0e - u(%1.1f) =~ %1.7f - erro_relativo = %1.1e" % (h, t, sol_euler, erro_
```

Para fins de comparação, calculamos a solução exata e aproximada para alguns valores de t e de passo h e resumimos na tabela abaixo:

t	Exato	Euler $h = 0{,}1$	Euler $h = 0{,}01$
0	1/2	0,5	0,5
1/2	$\frac{e^{1/2}}{1+e^{1/2}} \approx 0{,}6224593$	0,6231476	0,6225316
1	$\frac{e}{1+e} \approx 0{,}7310586$	0,7334030	0,7312946
2	$\frac{e^2}{1+e^2} \approx 0{,}8807971$	0,8854273	0,8812533
3	$\frac{e^3}{1+e^3} \approx 0{,}9525741$	0,9564754	0,9529609

Exercícios Resolvidos

ER 10.2.1. Aproxime a solução do problema de valor inicial

$$u'(t) = -0{,}5u(t) + 2 + t \tag{10.36}$$
$$u(0) = 8 \tag{10.37}$$

Usando os seguinte passos: $h = 10^{-1}$, $h = 10^{-2}$, $h = 10^{-3}$, $h = 10^{-4}$ e $h = 10^{-5}$ e compare a solução aproximada em $t = 1$ com a solução exata dada por:

$$u(t) = 2t + 8e^{-t/2} \Longrightarrow u(1) = 2 + 8e^{-1/2} \approx 6{,}85224527770107 \tag{10.38}$$

Solução. Primeramente itentificamos $f(t,u) = -0{,}5u + 2 + t$ e construímos o processo iterativo do método de Euler:

$$u^{(n+1)} = u^{(n)} + h(-0{,}5u^{(n)} + 2 + t^{(n)}), \quad n = 1,2,3,\ldots \tag{10.39}$$
$$u^{(1)} = 8 \tag{10.40}$$

O seguinte código pode ser usado para implementar no `Python` a recursão acima:

```
def euler(h,Tmax):
u=8
   itmax = Tmax/h;
for i in np.arange(0,itmax):
t=i*h
k1 = -0.5*u+2+t
u = u + h*k1
return u


sol_exata = 2+8*np.exp(-.5)
h=1e-1
for i in np.arange(1,5):
sol_euler=euler(h,1);
erro_relativo = np.fabs((sol_euler-sol_exata)/sol_exata)
print("h=%1.0e - u(1) =~ %1.7f - erro_relativo = %1.1e" % (h, sol_euler, erro_re
h=h/10
```

A seguinte tabela resume os resultados obtidos:

h	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
Euler	6,7898955	6,8461635	6,8516386	6,8521846	6,8522392
ε_{rel}	9,1e-03	8,9e-04	8,9e-05	8,9e-06	8,9e-07

$\Diamond$

Exercícios

E 10.2.1. Resolva o problema de valor inicial a seguir envolvendo uma equação não autônoma, isto é, quando a função $f(t,u)$ depende explicitamente do tempo. Use passo $h = 0{,}1$ e $h = 0{,}01$. Depois compare com a solução exata dada por $u(t) = 2e^{-t} + t - 1$ nos intantes $t = 0$, $t = 1$, $t = 2$ e $t = 3$.

$$\begin{aligned} u'(t) &= -u(t) + t, \\ u(0) &= 1. \end{aligned} \tag{10.41}$$

E 10.2.2. Resolva o prolema de valor inicial envolvendo uma equação não linear usando passo $h = 0{,}1$ e $h = 0{,}01$.

$$\begin{aligned} u'(t) &= \cos(u(t)) \\ u(0) &= 0, \end{aligned} \tag{10.42}$$

Depois compare com a solução exata dada por

$$u(t) = \tan^{-1}\left(\frac{e^{2t}-1}{2e^t}\right). \tag{10.43}$$

nos intantes $t = 0$, $t = 1$, $t = 2$ e $t = 3$.

E 10.2.3. Resolva a seguinte problema de valor inicial linear com passo $h = 10^{-4}$ via método de Euler e compare a solução obtida com o valor exato $y(t) = e^{\text{sen}\,(t)}$ em $t = 2$:

$$\begin{aligned} y'(t) &= \cos(t)y(t) \\ y(0) &= 1. \end{aligned} \tag{10.44}$$

10.3 Método de Euler melhorado

O método de Euler estudado na Seção 10.2 é aplicação bastante restrita devido à sua pequena precisão, isto é, normalmente precisamos escolher um passo h muito pequeno para obter soluções de boa qualidade, o que implica um número elevado de passos e, consequentemente, alto custo computacional.

Nesta seção, contruiremos o **método de Euler melhorado** ou **método de Euler modificado** ou, ainda, **método de Heun**. Para tal, considere o problema de valor inicial dado por:

$$\begin{aligned} u'(t) &= f(t,u(t)), \quad t > t^{(1)} \\ u(t^{(1)}) &= a. \end{aligned} \tag{10.45}$$

Assim como fizemos para o método de Euler, integramos (10.45) de $t^{(1)}$ até $t^{(2)}$ e obtemos:

$$\int_{t^{(1)}}^{t^{(2)}} u'(t)\,dt = \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt \tag{10.46}$$

$$u(t^{(2)}) - u(t^{(1)}) = \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt \tag{10.47}$$

$$u(t^{(2)}) = u(t^{(1)}) + \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt \tag{10.48}$$

A invés de aproximar $f(t,u(t))$ como uma constante igual ao seu valor em $t = t^{(1)}$, aplicamos a regra do trapézio (ver 9.2.2) à integral envolvida no lado direito da expressão, isto é:

$$\int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt = \left[\frac{f\left(t^{(1)},u(t^{(1)})\right) + f\left(t^{(2)},u(t^{(2)})\right)}{2}\right] h + O(h^3) \tag{10.49}$$

onde $h = t^{(2)} - t^{(1)}$. Como o valor de $u(t^{(2)})$ não é conhecido antes de o passo ser realizado, aproximamos seu valor aplicando o método de Euler:

$$\tilde{u}(t^{(2)}) = u(t^{(1)}) + hf\left(t^{(1)},u(t^{(1)})\right) \tag{10.50}$$

Assim obtemos:

$$\begin{aligned} u(t^{(2)}) &= u(t^{(1)}) + \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt && (10.51)\\ &\approx u(t^{(1)}) + \left[\frac{f\left(t^{(1)},u(t^{(1)})\right) + f\left(t^{(2)},u(t^{(2)})\right)}{2}\right] h && (10.52)\\ &\approx u(t^{(1)}) + \left[\frac{f\left(t^{(1)},u(t^{(1)})\right) + f\left(t^{(2)},\tilde{u}(t^{(2)})\right)}{2}\right] h && (10.53)\\ & && (10.54) \end{aligned}$$

Portanto, o método recursivo de Euler melhorado assume a seguinte forma:

$$\begin{aligned} \tilde{u}^{(k+1)} &= u^{(k)} + hf(t^{(k)},u^{(k)}),\\ u^{(k+1)} &= u^{(k)} + \frac{h}{2}\left(f(t^{(k)},u^{(k)}) + f(t^{(k+1)},\tilde{u}^{(k+1)})\right), \qquad (10.55)\\ u^{(1)} &= a \quad \text{(condição inicial)}. \end{aligned}$$

Que pode ser escrito equivalentemente como:

$$\begin{aligned} k_1 &= f(t^{(k)},u^{(k)}),\\ k_2 &= f(t^{(k+1)},u^{(k)} + k_1),\\ u^{(k+1)} &= u^{(k)} + h\frac{k_1 + k_2}{2}, \qquad (10.56)\\ u^{(1)} &= a \quad \text{(condição inicial)}. \end{aligned}$$

Aqui k_1 e k_2 são variáveis auxiliares que representam as inclinações e devem ser calculadas a cada passo. Esta notação é compatível com a notação usada nos métodos de Runge-Kutta, uma família de esquemas iterativos para aproximar problemas de valor inicial, da qual o método de Euler e o método de Euler melhorado são casos particulares. Veremos os métodos de Runge-Kutta na Seção 10.7.

Exercícios Resolvidos

ER 10.3.1. Resolva pelo método de Euler melhorado problema de valor inicial do Exercício Resolvido 10.2.1:

$$\begin{aligned} u'(t) &= -0{,}5u(t) + 2 + t && (10.57)\\ u(0) &= 8 && (10.58) \end{aligned}$$

Usando os seguinte passos: $h = 10^{-1}$, $h = 10^{-2}$, $h = 10^{-3}$, $h = 10^{-4}$ e $h = 10^{-5}$ e compare a solução aproximada em $t = 1$ com a solução obtida pelo método de Euler e a solução exata dada por:

$$u(t) = 2t + 8e^{-t/2} \Longrightarrow u(1) = 2 + 8e^{-1/2} \approx 6{,}85224527770107 \tag{10.59}$$

Solução. Primeramente itentificamos $f(t,u) = -0{,}5u + 2 + t$ e construímos o processo iterativo do método de Euler melhorado:

$$\begin{aligned} k_1 &= f(t^{(n)},u^{(n)}) = -0{,}5u^{(n)} + 2 + t^{(n)} && (10.60)\\ \tilde{u} &= u^{(n)} + hk_1 && (10.61)\\ k_2 &= f(t^{(n+1)},\tilde{u}) = -0{,}5\tilde{u} + 2 + t^{(n+1)} && (10.62)\\ u^{(n+1)} &= u^{(n)} + h(k_1 + k_2),\ \ n = 1{,}2{,}3,\ldots && (10.63)\\ u^{(1)} &= 8 \ \text{(condição inicial)}. && (10.64) \end{aligned}$$

O seguinte código pode ser usado para implementar no `Python` a recursão acima:

```
def euler_mod(h,Tmax):
u=8
   itmax = Tmax/h;
for i in np.arange(0,itmax):
t=i*h
k1 = -0.5*u+2+t
u_til = u + h*k1
k2 = -0.5*u_til+2+(t+h)
u=u+h*(k1+k2)/2
return u



sol_exata = 2+8*np.exp(-1/2)
for h in [1e-1, 1e-2, 1e-3, 1e-4, 1e-5]:
sol_euler=euler_mod(h,1);
erro_relativo = np.fabs((sol_euler-sol_exata)/sol_exata)
print("h=%1.0e - u(1) =~ %1.7f - erro_relativo = %1.1e" % (h, sol_euler, erro_relativ
```

A seguinte tabela resume os resultados obtidos:

h	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
Euler	6,7898955	6,8461635	6,8516386	6,8521846	6,8522392
ε_{rel}	9,1e-03	8,9e-04	8,9e-05	8,9e-06	8,9e-07
Euler mod.	6,8532949	6,8522554	6,8522454	6,8522453	6,8522453
ε_{rel}	1,5e-04	1,5e-06	1,5e-08	1,5e-10	1,5e-12

$\Diamond$

Exercícios

Esta seção carece de exercícios. Participe da sua escrita.
Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

10.4 Solução de sistemas de equações diferenciais

Nas seções 10.2 e 10.3, construimos dois métodos numéricos para resolver problemas de valor inicial. Nestas seções, sempre consideremos problemas envolvendo equações diferenciais ordinárias de primeira ordem, isto é:

$$u'(t) = f(t,u(t)), \quad t > t^{(1)} \tag{10.65}$$
$$u(t^{(1)}) = a. \tag{10.66}$$

Estas técnicas podem ser diretamente estendidas para resolver numericamente problemas de valor inicial envolvendo sistemas de equações diferenciais ordinárias de

primeira ordem, isto é:

$$\begin{aligned}
u_1'(t) &= f_1(t,u_1(t),u_2(t),u_3(t),\ldots,u_n(t)), \quad t > t^{(1)},\\
u_2'(t) &= f_2(t,u_1(t),u_2(t),u_3(t),\ldots,u_n(t)), \quad t > t^{(1)},\\
u_3'(t) &= f_3(t,u_1(t),u_2(t),u_3(t),\ldots,u_n(t)), \quad t > t^{(1)},\\
&\vdots\\
u_n'(t) &= f^{(n)}(t,u_1(t),u_2(t),u_3(t),\ldots,u_n(t)), \quad t > t^{(1)},\\
u(t^{(1)}) &= a_1,\\
u(t^{(2)}) &= a_2,\\
u(t^{(3)}) &= a_3,\\
&\vdots\\
u(t^{(n)}) &= a_n.
\end{aligned} \tag{10.67}$$

O Problema (10.67) pode ser escrito como um problema de primeira ordem envolvendo uma única incógnita, $u(t)$, dada como um vetor de funções $u_j(t)$, isto é:

$$u(t) = \begin{bmatrix} u_1(t)\\ u_2(t)\\ u_3(t)\\ \vdots\\ u_n(t) \end{bmatrix} \tag{10.68}$$

De forma que o Problema (10.67) assuma a seguinte forma:

$$u'(t) = f(t,u(t)), \quad t > t^{(1)} \tag{10.69}$$

$$u(t^{(1)}) = a. \tag{10.70}$$

onde

$$f(t,u(t)) = \begin{bmatrix} f_1(t,u_1(t),u_2(t),u_3(t),\ldots,u_n(t))\\ f_2(t,u_1(t),u_2(t),u_3(t),\ldots,u_n(t))\\ f_3(t,u_1(t),u_2(t),u_3(t),\ldots,u_n(t))\\ \vdots\\ f^{(n)}(t,u_1(t),u_2(t),u_3(t),\ldots,u_n(t)) \end{bmatrix} \tag{10.71}$$

e

$$a = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} \tag{10.72}$$

Veja o o Exemplo 10.73

Exemplo 10.4.1. Considere o problema de resolver numericamente pelo método de Euler o seguinte sistema de equações diferenciais ordinárias com valores iniciais:

$$\begin{aligned} x'(t) &= -y(t), && \text{(10.73a)} \\ y'(t) &= x(t), && \text{(10.73b)} \\ u(0) &= 1. && \text{(10.73c)} \\ v(0) &= 0. && \text{(10.73d)} \end{aligned}$$

Para aplicar o método de Euler a este sistema, devemos encarar as duas incógnitas do sistema como entradas de um vetor, ou seja, escrevemos:

$$u(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}. \tag{10.74}$$

e, portanto, o sistema pode ser escrito como:

$$\begin{bmatrix} x^{(k+1)} \\ y^{(k+1)} \end{bmatrix} = \begin{bmatrix} x^{(k)} \\ y^{(k)} \end{bmatrix} + h \begin{bmatrix} -y^{(k)} \\ x^{(k)} \end{bmatrix}. \tag{10.75}$$

Observe que este processo iterativo é equivalente a discretiza as equações do sistema uma-a-uma, isto é:

$$\begin{aligned} x^{(k+1)} &= x^{(k)} - hy^{(k)}, && \text{(10.76)} \\ y^{(k+1)} &= y^{(k)} + hx^{(k)}, && \text{(10.77)} \\ x^{(1)} &= 1, && \text{(10.78)} \\ y^{(1)} &= 0, && \text{(10.79)} \\ & && \text{(10.80)} \end{aligned}$$

Exercícios Resolvidos

ER 10.4.1. Resolva pelo método de Euler melhorado o seguinte problema de valor inicial para aproximar o valor de x e y entre $t = 0$ e $t = 1$:

$$\begin{aligned} x'(t) &= x(t) - y(t), && (10.81)\\ y'(t) &= x(t) - y(t)^3, && (10.82)\\ u(0) &= 1. && (10.83)\\ v(0) &= 0. && (10.84) \end{aligned}$$

Solução. Primeiramente, identificamos $u(t)$ como o vetor incógnita:

$$u(t) = \begin{bmatrix} x(t) \\ y(t) \end{bmatrix}. \tag{10.85}$$

Depois aplicamos a recursão do método de Euler melhorado dada por:

$$\begin{aligned} \tilde{u}^{(k+1)} &= u^{(k)} + hf(t^{(k)},u^{(k)}), && (10.86)\\ u^{(k+1)} &= u^{(k)} + \frac{h}{2}\left(f(t^{(k)},u^{(k)}) + f(t^{(k)},\tilde{u}^{(k)})\right), && (10.87)\\ & && (10.88) \end{aligned}$$

isto é:

$$\begin{aligned} \tilde{x}^{(k+1)} &= x^{(k)} + h\left(x^{(k)} - y^{(k)}\right) && (10.89)\\ \tilde{y}^{(k+1)} &= y^{(k)} + h\left(x^{(k)} - {y^{(k)}}^3\right) && (10.90)\\ x^{(k+1)} &= x^{(k)} + \frac{h}{2}\left[\left(x^{(k)} - y^{(k)}\right) + \left(\tilde{x}^{(k)} - \tilde{y}^{(k)}\right)\right] && (10.91)\\ y^{(k+1)} &= y^{(k)} + \frac{h}{2}\left[\left(x^{(k)} - {y^{(k)}}^3\right) + \left(\tilde{x}^{(k)} - {\tilde{y}^{(k)}}^3\right)\right] && (10.92)\\ & && (10.93) \end{aligned}$$

A tabela a seguir resume os resultados obtidos:

h		$t = 0{,}2$	$t = 0{,}4$	$t = 0{,}6$	$t = 0{,}8$	$t = 1{,}0$
10^{-2}	x	1.1986240	1.3890564	1.5654561	1.7287187	1.8874532
	y	0.2194288	0.4692676	0.7206154	0.9332802	1.0850012
10^{-3}	x	1.1986201	1.3890485	1.5654455	1.7287066	1.8874392
	y	0.2194293	0.4692707	0.7206252	0.9332999	1.0850259
10^{-4}	x	1.1986201	1.3890484	1.5653609	1.7287065	1.8874390
	y	0.2194293	0.4692707	0.7205062	0.9333001	1.0850262

A seguinte rotina pode ser usada para implementar a solução do sistema:

```
def euler_mod(h,Tmax,u1):
   itmax = Tmax/h;
x=np.empty(itmax+1)
y=np.empty(itmax+1)
x[0]=u1[0]
y[0]=u1[1]

for i in np.arange(0,itmax):
t=i*h
kx1 = (x[i]-y[i])
ky1 = (x[i]-y[i]**3)

x_til = x[i] + h*kx1
y_til = y[i] + h*ky1

kx2 = (x_til-y_til)
ky2 = (x_til-y_til**3)

x[i+1]=x[i]+h*(kx1+kx2)/2
y[i+1]=y[i]+h*(ky1+ky2)/2

return [x,y]


Tmax=1  #tempo maximo de simulacao
u1=np.asarray([1,0]) #condicoes iniciais na forma vetorial
h=1e-4 #passo
sol_euler=euler_mod(h,Tmax,u1);

itmax=Tmax/h

for t in [0, .2, .4, .6, .8, 1]:
k=t/h
print("h=%1.0e - x(%1.1f) =~ %1.6f - y(%1.1f) =~ %1.6f" % (h, t, sol_euler[0][k]
```

◊

Exercícios

Esta seção carece de exercícios. Participe da sua escrita.
Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

10.5 Solução de equações e sistemas de ordem superior

Na Seção 10.4, estendemos os métodos de Euler e Euler melhorado visto nas seções 10.2 e 10.3 para resolver numericamente problemas de valor inicial envolvendo sistemas de equações diferenciais ordinárias de primeira ordem. Nesta seção, estenderemos estas técnicas para resolver alguns tipos de problemas de ordem superior. Para tal, converteremos a equação diferencial em um sistema, incluindo as derivadas da incógnita como novas incógnitas. Vejamos um exemplo:

Exemplo 10.5.1. Resolva o problema de valor inicial de segunda ordem dado por

$$\begin{aligned} y'' + y' + y &= \cos(t), && (10.94)\\ y(0) &= 1, && (10.95)\\ y'(0) &= 0, && (10.96) \end{aligned}$$

A fim de transformar a equação diferencial dada em um sistema de equações de primeira ordem, introduzimos a substituição $w = y'$, de forma que obteremos o sistema:

$$\begin{aligned} y' &= w && (10.97)\\ w' &= -w - y + \cos(t) && (10.98)\\ y(0) &= 1 && (10.99)\\ w(0) &= 0 && (10.100) \end{aligned}$$

Este sistema pode ser resolvido usando as técnicas da Seção 10.4.

Exercícios resolvidos

ER 10.5.1. Considere o seguinte sistema envolvendo uma equação de segunda ordem e uma de primeira ordem:

$$\begin{aligned} x''(t) - (1 - 0{,}1z(t))x'(t) + x(t) &= 0 && (10.101)\\ 10z'(t) + z(t) &= x(t)^2 && (10.102) \end{aligned}$$

sujeito a condições iniciais dadas por:

$$\begin{aligned} x(0) &= 3 && (10.103)\\ x'(0) &= 0 && (10.104)\\ z(0) &= 10 && (10.105)\\ & && (10.106)\end{aligned}$$

Rescreva este sistema como um sistema de três equações de primeira ordem.

Solução. Definimos $y(t) = x'(t)$, pelo que o sistema se torna:

$$\begin{aligned} x'(t) &= y(t) && (10.107)\\ y'(t) - (1 - 0{,}1z(t))y(t) + x(t) &= 0 && (10.108)\\ 10z'(t) + z(t) &= x(t)^2 && (10.109)\end{aligned}$$

defina o vetor $u(t)$ como:

$$u(t) = \begin{bmatrix} x(t) \\ y(t) \\ z(t) \end{bmatrix} \tag{10.110}$$

De forma que:

$$u'(t) = \begin{bmatrix} x'(t) \\ y'(t) \\ z'(t) \end{bmatrix} = \begin{bmatrix} y(t) \\ [1 - 0{,}1z(t)]\, y(t) - x(t) \\ [x(t)^2 - z(t)]\,/10 \end{bmatrix} \tag{10.111}$$

ou (10.112)

$$u'(t) = \begin{bmatrix} u_1'(t) \\ u_2'(t) \\ u_3'(t) \end{bmatrix} = \begin{bmatrix} u_2(t) \\ [1 - 0{,}1u_3(t)]\, u_2(t) - u_1(t) \\ [u_1(t)^2 - u_3(t)]\,/10 \end{bmatrix} \tag{10.113}$$

sujeito às condições iniciais dadas por:

$$u'(0) = \begin{bmatrix} x'(0) \\ y'(0) \\ z'(0) \end{bmatrix} = \begin{bmatrix} 3 \\ 0 \\ 10 \end{bmatrix} \tag{10.114}$$

$\Diamond$

Exercícios

E 10.5.1. Resolva o problema de valor inicial dado por

$$\begin{aligned} x' &= -2x + \sqrt{y} && (10.115)\\ y' &= x - y && (10.116)\\ x(0) &= 0 && (10.117)\\ y(0) &= 2 && (10.118)\\ & && (10.119) \end{aligned}$$

com passo $h = 2 \cdot 10^{-1}$ $h = 2 \cdot 10^{-2}$, $h = 2 \cdot 10^{-3}$ e $h = 2 \cdot 10^{-4}$ para obter aproximações para $x(2)$ e $y(2)$.

E 10.5.2. Considere o problema de segunda order dado por:

$$x''(t) + x'(t) + \operatorname{sen}(x(t)) = 1, \tag{10.120}$$

sujeito às condições iniciais dadas por:

$$\begin{aligned} x(0) &= 2, && (10.121)\\ x'(0) &= 0. && (10.122) \end{aligned}$$

Resolva numericamente para obter o valor de $x(0{,}5)$, $x(1)$, $x(1{,}5)$ e $x(2)$ com passo $h = 10^{-2}$ e $h = 10^{-3}$ via método de Euler modificado.

10.6 Erro de truncamento

Nas seções 10.2 e 10.3, construimos dois métodos numéricos para resolver problemas de valor inicial. No Exercício Resolvido 10.3.1, vimos que o erro do método de Euler de do método de Euler melhorado caem quando se reduz o passo h, ademais, o erro do método de Euler melhorado cai conforme o quadrado de h, enquando o do método de Euler cai conforme h^2. Este fenômeno motiva a defingão de **ordem de precisão**.

Definição 10.6.1. *O **erro de truncamento local** é definido como o erro introduzido em cada passo pelo truncamento da equação diferencial supondo conhecida a solução exata no início do intervalo. Um método numérico é dito ter **ordem de precisão** p se o erro de truncamento local for da ordem de h^{p+1}.*

Exemplo 10.6.1. O método de Euler tem erro de truncamento local de ordem 1. Para obter este resultado, observamos via expansão de Taylor que:

$$u(t+h) = u(t) + hu'(t) + \frac{h^2}{2}u''(t) + O(h^3). \tag{10.123}$$

Se escolhermos nesta expressão $t = t^{(n)}$ e, portanto, $t + h = t^{(n)} + h = t^{(n+1)}$, temos:

$$t^{(n+1)} = t^{(n)} + hu'(t^{(n)}) + \frac{h^2}{2}u''(t^{(n)}) + O(h^3) \tag{10.124}$$

Agora notamos que o termo principal do erro é dado por $\frac{h^2}{2}u''(t^{(n)})$, como a derivada segunda da solução não depende de h, o erro local de truncamento decresce conforme h^2 e assim a ordem de precisão do método é 1.

Definição 10.6.2. *O **erro de truncamento global** é definido como erro acumulado ao longo de todos os passo de resolução, supondo a condição inicial exata.*

A relação entre o erro de truncamento global e o erro de truncamento local depende da função $f(t,u)$ envolvida. Diante de suficiente regularidade, o erro acumulado é da mesma ordem de grande do erro de truncamento local acumulado ao longo do processo, isto é, pode ser estimado multiplicando o erro local pelo número de passos. Como o número de passos N necessários para calcular a solução de um problema de valor inicial no ponto $t = t_f$ é dado por $N = \frac{t_f}{h}$, temos que a erro de truncamento global é uma ordem inferior ao erro de truncamento local e equivale à ordem de precisão do método.

Usamos também a notação ETL para o erro de truncamento local e ETG para o erro de truncamento global. De forma que, para o método de Euler, temos:

$$ETL_{Euler} = O(h^2) \quad \text{e} \quad ETG_{Euler} = O(h). \tag{10.125}$$

Exemplo 10.6.2. Vamos obter o erro de truncamento local do método de Euler melhorado. Partimos da construção do esquema iterativo de Euler melhorado:

$$\int_{t^{(1)}}^{t^{(2)}} u'(t)\,dt = \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt \tag{10.126}$$

$$u(t^{(2)}) - u(t^{(1)}) = \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt \tag{10.127}$$

$$u(t^{(2)}) = u(t^{(1)}) + \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt \tag{10.128}$$

Neste ponto, usamos o erro de truncamento do método de trapézios para aproximar a integral envolvida:

$$\int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\,dt = \frac{h}{2}\left[f\left(t^{(1)},u(t^{(1)})\right) + f\left(t^{(2)},u(t^{(2)})\right)\right] + O(h^3) \tag{10.129}$$

Assim, temos que o erro de truncamento local do método de Euler melhorado é $O(h^3)$ e, portanto, um método de ordem 2.

Exercícios resolvidos

Esta seção carece de exercícios resolvidos. Participe da sua escrita.
Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

Exercícios

Esta seção carece de exercícios. Participe da sua escrita.
Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

E 10.6.1. Aplique o método de Euler e o método de Euler melhorado para resolver o problema de valor inicial dado por

$$u' = -2u + \sqrt{u} \tag{10.130}$$
$$u(0) = 1 \tag{10.131}$$

com passo $h = 10^{-1}$, $h = 10^{-2}$, $h = 10^{-3}$, $h = 10^{-4}$ e $h = 10^{-5}$ para obter aproximações para $u(1)$. Compare com a solução exata dada do problema dada por $u(t) = \left(1 + 2e^{-t} + e^{-2t}\right)/4$ através do erro relativo e observe a ordem de precisão do método.

E 10.6.2. Resolva o problema de valor inicial dado por

$$u' = \cos(tu(t)) \tag{10.132}$$
$$u(0) = 1 \tag{10.133}$$
$$\tag{10.134}$$

com passo $h = 10^{-1}$, $h = 10^{-2}$, $h = 10^{-3}$, $h = 10^{-4}$ e $h = 10^{-5}$ para obter aproximações para $u(2)$

10.7 Métodos de Runge-Kutta explícitos

Nas seções anteriores, exploramos os métodos de Euler e Euler modificado para resolver problemas de valor inicial. Neste momento, deve estar claro ao leitor que o método de Euler melhorado produz soluções de melhor qualidade que o método de Euler para a maior parte dos problemas estudados. Isso se deve, conforme Seção 10.6, à ordem de precisão do método.

Os métodos de Runge-Kutta generalizam o esquema do método de Euler melhorado, inserindo mais estágios de cálculo e buscando ordens de precisão mais altas. Nesta seção, trataremos dos métodos de **Runge-Kutta explícitos**[2]

Para tal, considere o problema de valor inicial:

$$u'(t) = f(t,u(t)), \tag{10.135}$$
$$u(t_0) = a. \tag{10.136}$$

Integrando a EDO em $[t^{(n)},t^{(n+1)}]$ obtemos

$$u^{(n+1)} = u^{(n)} + \int_{t^{(n)}}^{t^{(n+1)}} f(t,u(t))\,dt \tag{10.137}$$

O método de Euler aproxima a integral no lado direito da expressão acima utilizando apenas o valor de $f(t,u)$ em $t = t^{(n)}$, o método de Euler melhorado aproxima esta integral utilizando os valores de $f(t,u)$ $t = t^{(n)}$ e $t = t^{(n+1)}$. Os métodos de Runge-Kutta explícitos admitem estágios intermediários, utilizando outros valores de $f(t,u)$ nos pontos $\{\tau_1,\tau_2,\ldots,\tau_\nu\}$ dentro do intervalo $[t_n,t^{(n+1)}]$. Veja esquema abaixo:

$$\begin{array}{cccc} u_n & & & u_{n+1} \\ | & & & | \\ \hline t^{(n)} & & & t^{(n+1)} \\ \tau_1 & \tau_2 & \cdots & \tau_\nu \\ t^{(n)} & t^{(n)}+c_2h & \cdots & t^{(n)}+c_\nu h \end{array}$$

Observe que $\tau_j = t^{(n)} + c_j h$ com $0 = c_1 \leq c_2 \leq \cdots \leq c_\nu \leq 1$, isto é, o primeira ponto sempre coincide com o extremo esquerdo do intervalo, mas o último ponto não precisa ser o extremo direito. Ademais, um mesmo ponto pode ser usado mais de uma vez com aproximações diferentes para $u(\tau_j)$.

Desta forma, aproximamos a integral por um esquema de quadratura com ν pontos:

$$\int_{t^{(n)}}^{t^{(n+1)}} f(t,u(t))\,dt \approx u_n + h\sum_{j=1}^{\nu} b_j f\left(\tau_j,u(\tau_j)\right) \tag{10.138}$$

onde b_j são os pesos da quadratura numérica que aproxima a integral. Assim como na construção do método de Euler melhorado, não dispomos dos valores $u(\tau_j)$ antes

[2]Existem também os métodos implícitos que serão abordados na Seção 10.8. Ver Observação 10.7.1.

de calculá-los e, por isso, precisamos estimá-los com base nos estágios anteriores:

$$\begin{aligned}
\tilde{u}_1 &= u^{(n)} \\
\tilde{u}_2 &= u_n + h a_{21} k_1 \\
\tilde{u}_3 &= u_n + h\left[a_{31}k_1 + a_{32}k_2\right] \\
\tilde{u}_4 &= u_n + h\left[a_{41}k_1 + a_{42}k_2 + a_{43}k_3\right] \\
&\vdots \\
\tilde{u}_\nu &= u_n + h\left[a_{\nu 1}k_1 + a_{\nu 2}k_2 + \cdots + a_{\nu\nu}k_\nu\right] \\
u^{(n+1)} &= u_n + h[b_1k_1 + b_2k_2 + \cdots + b_\nu k_\nu],
\end{aligned} \tag{10.139}$$

onde $k_j = f\left(\tau_j, \tilde{u}_j\right)$, $A := (a_{ij})$ é a matriz Runge-Kutta (triangular inferior com diagonal zero), b_j são os pesos Runge-Kutta e c_j são os nós Runge-Kutta. Estes coeficientes podem ser escritos de forma compacta em uma tabela conforme a seguir:

$$\begin{array}{c|c} c & A \\ \hline & b^T \end{array} = \begin{array}{c|ccc} c_1 & 0 & 0 & 0 \\ c_2 & a_{21} & 0 & 0 \\ c_3 & a_{31} & a_{22} & 0 \\ \hline & b_1 & b_2 & b_3. \end{array} = \begin{array}{c|ccc} c_1 & & & \\ c_2 & a_{21} & & \\ c_3 & a_{31} & a_{22} & \\ \hline & b_1 & b_2 & b_3. \end{array}$$

Na tabela mais à direita, omitimos os termos obrigatoriamente nulos.

Exemplo 10.7.1. O método de Euler modificado pode ser escrito conforme:

$$\tilde{u}_1 = u^{(n)} \tag{10.140}$$
$$\tilde{u}_2 = u^{(n)} + hk_1 \tag{10.141}$$
$$u^{(n+1)} = u^{(n)} + h\left[\frac{1}{2}k_1 + \frac{1}{2}k_2\right] \tag{10.142}$$
$$\tag{10.143}$$

Identificando os coeficientes, obtemos $\nu = 2$, $c_1 = 0$, $c_2 = 1$, $a_{21} = 1$, $b_1 = \frac{1}{2}$ e $b_2 = \frac{1}{2}$. Escrevendo na forma tabular, temos:

$$\begin{array}{c|c} c & A \\ \hline & b \end{array} = \begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}$$

Observação 10.7.1. Nos métodos chamados explícitos, os elementos da diagonal principal (e acima dela) da matriz A devem ser nulos, pois a aproximação em cada estágio é calculada com base nos valores dos estágios anteriores. Nos métodos implícitos, essa restrição é removida, neste caso, o cálculo da aproximação em um estágio envolve a solução de uma equação algébrica.

Observação 10.7.2. Além da condição fixa $c_1 = 0$, para que um método seja de ordem pelo menos unitária, isto é, $p \geq 1$.

10.7.1 Métodos de Runge-Kutta com dois estágios

Os métodos de Runge-Kutta com dois estágios ($\nu = 2$) são da seguinte forma:

$$\begin{aligned} \tilde{u}_1 &= u^{(n)} && (10.144)\\ \tilde{u}_2 &= u^{(n)} + ha_{21}k_1 && (10.145)\\ u^{(n+1)} &= u^{(n)} + h[b_1k_1 + b_2k_2], && (10.146) \end{aligned}$$

onde $k_1 = f(t^{(n)},u^{(n)})$ e $k_2 = f(t^{(n)} + c_2h,\tilde{u}_2)$

Assumindo suavidade suficiente em f, usamos o polinônio de Taylor:

$$\begin{aligned} k_2 &= f(t^{(n)} + c_2h, \tilde{u}_2) && (10.147)\\ &= f(t^{(n)} + c_2h, u^{(n)} + a_{21}hk_1) && (10.148)\\ &= f(t^{(n)}, u^{(n)}) + h\left[c_2\frac{\partial f}{\partial t} + a_{21}k_1\frac{\partial f}{\partial u}\right] + O(h^2) && (10.149)\\ &= k_1 + h\left(c_2\frac{\partial f}{\partial t} + a_{21}k_1\frac{\partial f}{\partial u}\right) + O(h^2) && (10.150) \end{aligned}$$

fazendo com que (10.146) se torne

$$\begin{aligned} u^{(n+1)} &= u^{(n)} + hb_1k_1 + hb_2k_2 && (10.151)\\ &= u_n + hb_1k1 + hb_2\left[k_1 + h\left(c_2\frac{\partial f}{\partial t} + a_{21}k_1\frac{\partial f}{\partial u}\right) + O(h^2)\right] && (10.152)\\ &= u_n + h(b_1 + b_2)k_1 + h^2b_2\left(c_2\frac{\partial f}{\partial t} + a_{21}k_1\frac{\partial f}{\partial u}\right) + O(h^3) && (10.153) \end{aligned}$$

Usando a equação diferencial ordinária que desejamos resolver e derivando-a em t, obtemos:

$$\begin{aligned} u'(t) &= f(t,u(t)), && (10.154)\\ u''(t) &= \frac{\partial f}{\partial t} + \frac{\partial f}{\partial u}u'(t) = \frac{\partial f}{\partial t} + \frac{\partial f}{\partial u}f(t,u(t)). && (10.155) \end{aligned}$$

Agora, expandimos em série de Taylor a solução exata $u(t)$ em $t = t^{(n)}$,

$$\begin{aligned}
u(t^{(n+1)}) &= u(t^{(n)}+h) = u^{(n)} + hu'(t) + \frac{h^2}{2}u''(t) + O(h^3) && (10.156)\\
&= u^{(n)} + hf(t,u^{(n)}) + \frac{h^2}{2}\frac{d}{dt}f(t,u(t)) + O(h^3) && (10.157)\\
&= u^{(n)} + hf(t,u^{(n)}) + \frac{h^2}{2}\left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial u}u'(t^{(n)})\right) + O(h^3) && (10.158)\\
&= u^{(n)} + hf(t,u^{(n)}) + \frac{h^2}{2}\left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial u}f(t,u^{(n)})\right) + O(h^3) && (10.159)\\
&= u^{(n)} + hf(t,u^{(n)}) + \frac{h^2}{2}\left(\frac{\partial f}{\partial t} + \frac{\partial f}{\partial u}k_1\right) + O(h^3) && (10.160)
\end{aligned}$$

Finalmente comparamos os termos em (10.153) e (10.160) de forma a haver concordância na expansão de Taylor até segunda ordem, isto é, restanto apenas o erro de ordem 3 e produzindo um método de ordem de precisão $p = 2$:

$$b_1 + b_2 = 1, \quad b_2c_2 = \frac{1}{2} \quad \text{e} \quad a_{21} = c_2 \tag{10.161}$$

Este sistema é formada por três equações e quatro incógnitas, pelo que admite infinitas soluções. Para construir toda a família de soluções, escolha um parâmetro $\alpha \in (0,1]$ e defina a partir de (10.161):

$$b_1 = 1 - \frac{1}{2\alpha}, \;\; b_2 = \frac{1}{2\alpha}, \;\; c_2 = \alpha, \;\; a_{21} = \alpha \tag{10.162}$$

Portanto, obtemos o seguinte esquema genérico:

$$\begin{array}{c|c} c & A \\ \hline & b \end{array} \;=\; \begin{array}{c|cc} c_2 & a_{21} & \\ \hline & b_1 & b_2 \end{array} \;=\; \begin{array}{c|cc} \alpha & \alpha & \\ \hline & \left(1-\frac{1}{2\alpha}\right) & \frac{1}{2\alpha} \end{array}, \quad 0 < \alpha \leq 1.$$

Algumas escolhas comuns são $\alpha = \frac{1}{2}$, $\alpha = \frac{2}{3}$ e $\alpha = 1$:

$$\begin{array}{c|cc} 0 & & \\ \frac{1}{2} & \frac{1}{2} & \\ \hline & 0 & 1 \end{array}, \qquad \begin{array}{c|cc} 0 & & \\ \frac{2}{3} & \frac{2}{3} & \\ \hline & \frac{1}{4} & \frac{3}{4} \end{array} \quad \text{e} \quad \begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}.$$

Note que a tabela da direita fornece o método Euler modificado ($\alpha = 1$). O esquema iterativo assume a seguinte forma:

$$\begin{aligned}
\tilde{u}_1 &= u^{(n)} && (10.163)\\
\tilde{u}_2 &= u^{(n)} + h\alpha k_1 && (10.164)\\
u^{(n+1)} &= u^{(n)} + h\left[\left(1 - \frac{1}{2\alpha}\right)k_1 + \frac{1}{2\alpha}k_2\right], && (10.165)
\end{aligned}$$

onde $k_1 = f(t^{(n)},u^{(n)})$ e $k_2 = f(t^{(n)} + h\alpha,\tilde{u}_2)$. Ou, equivalentemente:

$$\begin{aligned} k_1 &= f(t^{(n)},u^{(n)}) && (10.166)\\ k_2 &= f(t^{(n)} + h\alpha,u^{(n)} + h\alpha k_1) && (10.167)\\ u^{(n+1)} &= u^{(n)} + h\left[\left(1 - \frac{1}{2\alpha}\right)k_1 + \frac{1}{2\alpha}k_2\right], && (10.168) \end{aligned}$$

10.7.2 Métodos de Runge-Kutta com três estágios

Os métodos de Runge-Kutta com 3 estágios podem ser descritos na forma tabular como:

$$\begin{array}{c|ccc} 0 & & & \\ c_2 & a_{21} & & \\ c_3 & a_{31} & a_{32} & \\ \hline & b_1 & b_2 & b_3 \end{array}$$

Seguindo um procedimento similar ao da Seção 10.7.1, podemos obter as condições equivalentes às condições (10.161) para um método com $\nu = 3$ e ordem $p = 3$, as quais são:

$$\begin{aligned} b_1 + b_2 + b_3 &= 1, && (10.169)\\ b_2c_2 + b_3c_3 &= \frac{1}{2}, && (10.170)\\ b_2c_2^2 + b_3c_3^2 &= \frac{1}{3}, && (10.171)\\ b_3a_{32}c_2 &= \frac{1}{6}, && (10.172)\\ a_{21} &= c_2, && (10.173)\\ a_{31} + a_{32} &= c_3. && (10.174) \end{aligned}$$

Assim, temos 6 condições para determinar 8 incógnitas, o que implica a existência de uma enorme família de métodos de Runge-Kutta com três estágios e ordem

$p = 3$. Se fixarmos os coeficientes c_2 e c_3, podemos os outros de forma única:

$$b_1 = 1 - \frac{1}{2c_2} - \frac{1}{2c_3} + \frac{1}{3c_2c_3} \tag{10.175}$$

$$b_2 = \frac{3c_3 - 2}{6c_2(c_3 - c_2)} \tag{10.176}$$

$$b_3 = \frac{2 - 3c_2}{6c_3(c_3 - c_2)} \tag{10.177}$$

$$a_{21} = c_2 \tag{10.178}$$

$$a_{31} = c_3 - \frac{1}{6b_3c_2} \tag{10.179}$$

$$a_{32} = \frac{1}{6b_3c_2} \tag{10.180}$$

Alguns exemplos de métodos de Runge-Kutta de 3 estágios são o método clássico de Runge-Kutta $\left(c_2 = \frac{1}{2} \text{ e } c_3 = 1\right)$ e o método de Nyström $\left(c_2 = c_3 = \frac{2}{3}\right)$:

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ 1 & -1 & 2 & \\ \hline & \frac{1}{6} & \frac{4}{6} & \frac{1}{6} \end{array} \quad \text{e} \quad \begin{array}{c|ccc} 0 & & & \\ \frac{2}{3} & \frac{2}{3} & & \\ \frac{2}{3} & 0 & \frac{2}{3} & \\ \hline & \frac{2}{8} & \frac{3}{8} & \frac{3}{8} \end{array} .$$

10.7.3 Métodos de Runge-Kutta com quatro estágios

As técnicas utilizadas nas seções 10.7.1 e 10.7.2 podem ser usadas para obter métodos de quarta ordem ($p = 4$) e quatro estágios ($\nu = 4$). As seguintes tabelas descrevem os dois esquemas mais conhecidos de Runge-Kutta quarta ordem com quatro estágios. O primeiro é denominado **método de Runge-Kutta 3/8** e o segundo é chamado de **método de Runge-Kutta clássico**.

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{3} & \frac{1}{3} & & & \\ \frac{2}{3} & -\frac{1}{3} & 1 & & \\ 1 & 1 & -1 & 1 & \\ \hline & \frac{1}{8} & \frac{3}{8} & \frac{3}{8} & \frac{1}{8} \end{array} \quad \text{e} \quad \begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{2}{6} & \frac{2}{6} & \frac{1}{6} \end{array}$$

O método de Runge-Kutta clássico é certamente o mais notório dos métodos

de Runge-Kutta e seu esquema iterativo pode ser escrito como a seguir:

$$k_1 = hf\left(t^{(n)},u^{(n)}\right) \tag{10.181}$$
$$k_2 = hf\left(t^{(n)}+h/2,u^{(n)}+hk_1/2\right) \tag{10.182}$$
$$k_3 = hf\left(t^{(n)}+h/2,u^{(n)}+hk_2/2\right) \tag{10.183}$$
$$k_4 = hf\left(t^{(n)}+h,u^{(n)}+hk_3\right) \tag{10.184}$$
$$u^{(n+1)} = u^{(n)}+\frac{k_1+2k_2+2k_3+k_4}{6} \tag{10.185}$$

A seguinte heurística, usando o método de Simpson para quadratura numérica, pode ajudar a compreender os estranhos coeficientes:

$$u(t^{(n+1)})-u(t^{(n)}) = \int_{t^{(n)}}^{t^{(n+1)}} f(t,u(s))ds \tag{10.186}$$
$$\approx \frac{h}{6}\left[f\left(t^{(n)},u(t^{(n)})\right)+4f\left(t^{(n)}+h/2,u(t^{(n)}+h/2)\right)\right. \tag{10.187}$$
$$+ \left. f\left(t^{(n)}+hu(t^{(n)}+h)\right)\right] \tag{10.188}$$
$$\approx h\frac{k_1+4(\frac{k_2+k_3}{2})+k_4}{6} \tag{10.189}$$

onde k_1 e k_4 representam os valores de $f(t,u)$ nos extremos; k_2 e k_3 são duas aproximações diferentes para a inclinação no meio do intervalo.

Exercícios resolvidos

ER 10.7.1. Construa o esquema iterativo o método clássico de Runge-Kutta três estágios cuja tabela é dada a seguir:

0			
$\frac{1}{2}$	$\frac{1}{2}$		
1	-1	2	
	$\frac{1}{6}$	$\frac{4}{6}$	$\frac{1}{6}$

Solução.

$$k_1 = f\left(t^{(n)},u^{(n)}\right) \tag{10.190}$$
$$k_2 = f\left(t^{(n)}+h/2,u^{(n)}+k_1/2\right) \tag{10.191}$$
$$k_3 = f\left(t^{(n)}+h,u^{(n)}-k_1+2k_2\right) \tag{10.192}$$
$$u^{(n+1)} = u^{(n)}+h\frac{k_1+4k_2+k_4}{6} \tag{10.193}$$

$\Diamond$

ER 10.7.2. Utilize o método clássico de Runge-Kutta três estágios para calcular o valor de $u(2)$ com passos $h = 10^{-1}$ e $h = 10^{-2}$ para o seguinte problema de valor inicial:

$$u'(t) = -u(t)^2 + t, \tag{10.194}$$
$$u(0) = 0. \tag{10.195}$$

Aplicando o processo iterativo obtido no Problema Resolvido 10.7.1, obtemos a seguinte rotina:

```
 def f(t,u):
return t-u**2


def RK3_classico(h,Tmax,u1):
   itmax = Tmax/h;
u=np.empty(itmax+1)
u[0]=u1

for i in np.arange(0,itmax):
t=i*h

k1 = f(t,     u[i])
k2 = f(t+h/2, u[i] + h*k1/2)
k3 = f(t+h,   u[i] + h*(2*k2-k1))

u[i+1] = u[i] + h*(k1+4*k2+k3)/6
return u



Tmax=2 #tempo maximo de simulacao
u1=0 #condicoes iniciais na forma vetorial
h=1e-2 #passo

sol=RK3_classico(h,Tmax,u1);
itmax=Tmax/h
print(sol[itmax])
```

Exercícios

E 10.7.1. Aplique o esquema de Runge-Kutta segunda ordem com dois estágios cujos coeficientes são dados na tabela a seguir $\begin{array}{c|cc} 0 & & \\ \frac{2}{3} & \frac{2}{3} & \\ \hline & \frac{1}{4} & \frac{3}{4} \end{array}$ para resolver o problema de valor inicial dado por:

$$x'(t) = \operatorname{sen}(x(t)), \tag{10.196}$$
$$x(0) = 2. \tag{10.197}$$

para $t = 2$ com $h = 1e-1$, $h = 1e-2$ e $h = 1e-3$. Expresse sua resposta com oito dígitos significativos corretos.

E 10.7.2. Resolva pelo método de Euler, Euler melhorado, Runge-Kutta clássico três estágios e Runge-Kutta clássico quatro estágios o problema de valor inicial tratados nos exercícios resolvidos 10.2.1 e 10.3.1 dado por:

$$u'(t) = -0{,}5u(t) + 2 + t \tag{10.198}$$
$$u(0) = 8 \tag{10.199}$$

Usando os seguinte passos: $h = 1$, $h = 10^{-1}$, $h = 10^{-2}$ e $h = 10^{-3}$ e compare a solução aproximada em $t = 1$ com as soluções obtidas com a solução exata dada por:

$$u(t) = 2t + 8e^{-t/2} \Longrightarrow u(1) = 2 + 8e^{-1/2} \approx 6{,}85224527770107 \tag{10.200}$$

E 10.7.3. Aplique o método de Euler, o método de Euler melhorado, o método clássico de Runge-Kutta três estágios e o método clássico de Runge-Kutta quatro estágios para resolver o problema de valor inicial dado por

$$u' = u + t \tag{10.201}$$
$$u(0) = 1 \tag{10.202}$$

com passo $h = 1$, $h = 10^{-1}$, $h = 10^{-2}$ e $h = 10^{-3}$ para obter aproximações para $u(1)$. Compare com a solução exata dada do problema dada por $u(t) = 2e^t - t - 1$ através do erro relativo e observe a ordem de precisão do método. Expresse a sua resposta com oito dígitos significativos para a solução e 2 dígitos significativos para o erro relativo.

10.8 Métodos de Runge-Kutta implícitos

Nas seções anteriores contruímos os métodos de Runge-Kutta implícito, nesta seção veremos uma nova família de métodos chamados implícitos. Nos métodos implícitos o processo recursivo produz uma equação implícita para $y^{(n+1)}$ em termos

de $y^{(n)}$, como por exemplo:

$$\begin{aligned} y^{(n+1)} &= y^{(n)} + hy^{(n+1)} \\ y^{(1)} &= 1 \end{aligned} \tag{10.203}$$

para resolver o problema de valor inicial dado por:

$$y'(t) = y(t) \tag{10.204}$$
$$y(0) = 1 \tag{10.205}$$

Note que este método é **implícito** pois a expressão que define a iteração depende de $u^{(n+1)}$ dos dois lados da Equação (10.203), exigindo que o termo seja isolado para a aplicação do método.

10.8.1 Método de Euler implícito

Contruiremos, agora, o mais simples dos métodos para resolver problemas de valor inicial: o método de Euler implícito, uma variante do método de Euler (explícito) que vimos na Seção 10.2. Seguinte o mesmo raciocício daquela seção, integramos o problema de valor inicial dado por

$$u'(t) = f(t,u(t)) \tag{10.206}$$
$$u(t^{(1)}) = a \tag{10.207}$$

de $t^{(1)}$ até $t^{(2)}$ obtemos (como feito anteriormente) para obter

$$u(t^{(2)}) = u(t^{(1)}) + \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\, dt \tag{10.208}$$

Diferentemente do método de Euler estudado, o método de Euler implícito aproxima a função $f(t,u)$ pela uma função constante $f(t,u(t)) \approx f(t^{(2)},u^{(2)})$ e, assim, obtemos o seguinte esquema:

$$u^{(2)} = u^{(1)} + hf(t^{(2)},u^{(2)}) \tag{10.209}$$

Generalizando este procedimento para t_n obtemos o **método de Euler implícito**

$$u^{(n+1)} = u^{(n)} + h\ f(t^{(n+1)},u^{(n+1)}). \tag{10.210}$$

Note que este método é **implícito** (a equação é implícita) pois depende de u_{n+1} dos dois lados da equação. Se a função f for simples o suficiente, podemos resolver a equação isolando o termo u_{n+1}. Se isso não for possível, devemos usar um dos métodos vistos anteriormente para calcular as raízes da equação (por exemplo, método da bissecção e método de Newton).

Exemplo 10.8.1. Considere o problema de valor inicial dado por

$$u'(t) = \lambda u(t) \tag{10.211}$$
$$u(0) = 1 \tag{10.212}$$

A relação de recorrência do método de Euler implícito é dado por:

$$y^{(n+1)} = y^{(n)} h\lambda y^{(n+1)} \tag{10.213}$$
$$y^{(1)} = 1 \tag{10.214}$$

Isolando a $y^{(n+1)}$ na primeira equação, obtemos o processo iterativo dado por:

$$y^{(n+1)} = \frac{y^{(n)}}{1-\lambda h} \tag{10.215}$$
$$y^{(1)} = 1 \tag{10.216}$$

10.8.2 O método trapezoidal

O método de Euler aproxima a função $f(t,u)$ como uma constante no intervalo $[t^{(1)},t^{(2)}]$. O método trapezoidal é muito semelhante ao método de Euler melhorado estudado na Seção 10.3, integramos de $t^{(1)}$ até $t^{(2)}$ a equação diferencial envolvida no problema de valor inicial

$$\begin{aligned} u'(t) &= f(t,u(t)), \quad t > t^{(1)} \\ u(t^{(1)}) &= a. \end{aligned} \tag{10.217}$$

para obter:

$$\int_{t^{(1)}}^{t^{(2)}} u'(t)\, dt = \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\, dt \tag{10.218}$$
$$u(t^{(2)}) - u(t^{(1)}) = \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\, dt \tag{10.219}$$
$$u(t^{(2)}) = u(t^{(1)}) + \int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\, dt \tag{10.220}$$

Exatamente como no método de Euler melhorado, aplicamos a regra do trapézio (ver 9.2.2) à integral envolvida no lado direito da expressão, isto é:

$$\int_{t^{(1)}}^{t^{(2)}} f(t,u(t))\, dt = \left[\frac{f\left(t^{(1)},u(t^{(1)})\right) + f\left(t^{(2)},u(t^{(2)})\right)}{2}\right] h + O(h^3) \tag{10.221}$$

onde $h = t^{(2)} - t^{(1)}$.

Repetindo este procedimento para cada n, obtemos o esquema iterativo do método trapezoidal:

$$u^{(n+1)} = u^{(n)} + \frac{h}{2}\left(f(t^{(n)},u^{(n)}) + f(t^{(n+1)},u^{(n+1)})\right) \quad (10.222)$$

$$u^{(1)} = a \quad (10.223)$$

Exemplo 10.8.2. Considere o problema de valor inicial dado por

$$u'(t) = \lambda u(t) \quad (10.224)$$

$$u(0) = 1 \quad (10.225)$$

onde λ é uma constante. A relação de recorrência do método de Euler trapezoidal é dado por:

$$y^{(n+1)} = y^{(n)} - \frac{\lambda h}{2}\left[y^{(n+1)} + y^{(n)}\right] \quad (10.226)$$

$$y^{(1)} = 1 \quad (10.227)$$

Isolando a $y^{(n+1)}$ na primeira equação, obtemos o processo iterativo dado por:

$$y^{(n+1)} = \frac{1+\lambda h/2}{1-\lambda h/2} y^{(n)} \quad (10.228)$$

$$y^{(1)} = 1 \quad (10.229)$$

10.8.3 O método theta

O método theta é uma generalização dos métodos de Euler e trapezoidal. A relação de recorrência do método theta é dada por:

$$u^{(n+1)} = u^{(n)} + h(\theta f(t^{(n)},u^{(n)}) + (1-\theta) f(t^{(n+1)},u^{(n+1)})) \quad (10.230)$$

Observe que quando $\theta = 1$, a relação recai no método de Euler, quando $\theta = \frac{1}{2}$, no método trapezoidal e quando $\theta = 0$, no método de Euler implícito.

Exercícios resolvidos

ER 10.8.1. Considere o problema de valor inicial dado por:

$$y'(t) = y(t)\left(1 - y(t)\right), \quad (10.231)$$

$$y(0) = \frac{1}{2}. \quad (10.232)$$

Construa a recursão via método de Euler implícito e explicite o termo $y^{(n+1)}$.

Solução. O método de Euler implícito produz a seguinte recursão:

$$y^{(n+1)} = y^{(n)} + hy^{(n+1)}\left(1 - y^{(n+1)}\right) \tag{10.233}$$

a qual pode ser escrita como:

$$h\left[y^{(n+1)}\right]^2 + (1-h)y^{(n+1)} - y^{(n)} = 0 \tag{10.234}$$

Usando a fórmula da equação quadrática temos:

$$y^{(n+1)} = \frac{-(1-h) \pm \sqrt{(1-h)^2 + 4hy^{(n)}}}{2h} \tag{10.235}$$

Como a condição inicial é positiva, é fácil ver que $y(t) > 0$ para todo t e, portanto:

$$\begin{aligned}
y^{(n+1)} &= \frac{-(1-h) + \sqrt{(1-h)^2 + 4hy^{(n)}}}{2h} && (10.236)\\
&= (1-h)\frac{-1 + \sqrt{1 + \frac{4hy^{(n)}}{(1-h)^2}}}{2h} && (10.237)\\
&= \frac{(1-h)}{2h}\left[\sqrt{1 + \frac{4hy^{(n)}}{(1-h)^2}} - 1\right] && (10.238)\\
&= \frac{(1-h)}{2h}\frac{4hy^{(n)}}{(1-h)^2}\frac{1}{\left[\sqrt{1 + \frac{4hy^{(n)}}{(1-h)^2}} + 1\right]} && (10.239)\\
&= \frac{2}{(1-h)}\frac{1}{\left[1 + \sqrt{1 + \frac{4hy^{(n)}}{(1-h)^2}}\right]}y^{(n)} && (10.240)\\
& && (10.241)
\end{aligned}$$

◊

Exercícios

Esta seção carece de exercícios. Participe da sua escrita.

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

10.9 O método de Taylor

Uma maneira alternativa de aumentar a ordem dos métodos de Euler anteriormente descritos consiste em truncar a série de Taylor de $u(t+h)$:

$$u(t+h) = u(t) + hu'(t) + \frac{h^2}{2!}u''(t) + \frac{h^3}{3!}u'''(t) + \ldots \tag{10.242}$$

Utilizando dois termos temos o método de Euler. Utilizando os três primeiros termos da série e substituindo $u'(t) = f(t,x)$ e $u''(t) = \frac{df}{dt}(t,x)$ temos o **método de Taylor de ordem** 2

$$\begin{aligned} u^{(n+1)} &= u^{(n)} + hf(t^{(n)},u^{(n)}) && (10.243)\\ &+ \frac{h^2}{2!}\left[\frac{\partial}{\partial t}f(t^{(n)},u^{(n)}) + \frac{\partial}{\partial u}f(t^{(n)},u^{(n)})f(t^{(n)},u^{(n)})\right] && (10.244)\end{aligned}$$

onde usamos a regra da cadeida para obter:

$$\frac{d}{dt}f(t,u) = \frac{\partial}{\partial t}f(t,u) + \frac{\partial}{\partial u}f(t,u)u'(t) = \frac{\partial}{\partial t}f(t,u) + \frac{\partial}{\partial u}f(t,u)f(t,u) \quad (10.245)$$

O método de Taylor de ordem 3 é

$$u^{(n+1)} = u^{(n)} + hf(t^{(n)},u^{(n)}) + \frac{h^2}{2!}\frac{df}{dt}(t^{(n)},u^{(n)}) + \frac{h^3}{3!}\frac{d^2f}{dt^2}(t^{(n)},u^{(n)}) \quad (10.246)$$

Exercícios resolvidos

Esta seção carece de exercícios resolvidos. Participe da sua escrita.
Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

Exercícios

Esta seção carece de exercícios. Participe da sua escrita.
Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

10.10 Método de Adams-Bashforth

Seja o problema de valor inicial

$$\begin{aligned} u'(t) &= f(t,u(t)) && (10.247)\\ u(t_0) &= a && (10.248)\end{aligned}$$

Nos métodos de passo simples, os valores calculados para $f\left(t^{(n)},u(t^{(n)})\right)$ nos passos anteriores são desprezados ao calcular o próximo passo. Nos métodos de passo múltiplo, os valores de $f\,(t,u))$, nos passos n, $n+1$, ..., $n+s-1$ são utilizados ao calcular f em $t^{(n+s)}$.

Integrando a equação diferencial no intervalo $[t^{(n+s-1)},t^{(n+s)}]$, obtemos:

$$u^{(n+s)} = u^{(n+s-1)} + \int_{t^{(n+s-1)}}^{t^{(n+s)}} f(t,u(t))dt \tag{10.249}$$

No método de Adams-Bashforth, o integrando em (10.249) é aproximado pelo polinômio que interpola $f(t^{(k)},u^{(k)})$ para $k = n, n+1, n+2, \ldots, n+s-1$, isto é:

$$u^{(n+s)} = u^{(n+s-1)} + \int_{t^{(n+s-1)}}^{t^{(n+s)}} p(t)dt \tag{10.250}$$

onde $p(t)$ é polinômio de grau $s-1$ dado na forma de Lagrange por:

$$p(t) = \sum_{j=0}^{s-1} \left[f(t^{(n)},u^{(n)}) \prod_{k=0,k\neq j}^{s-1} \frac{t - t^{(n+k)}}{t^{(n+j)} - t^{(n+k)}} \right] \tag{10.251}$$

Agora observamos que

$$\int_{t^{(n+s-1)}}^{t^{(n+s)}} p(t)dt = h \sum_{j=0}^{s-1} \beta_j f(t^{(n+j)},u^{(n+j)}) \tag{10.252}$$

onde

$$\beta_j = \frac{1}{h} \int_{t^{(n+s-1)}}^{t^{(n+s)}} \prod_{k=0,k\neq j}^{s-1} \frac{t - t^{(n+k)}}{t^{(n+j)} - t^{(n+k)}} dt \tag{10.253}$$

e obtemos a relação de recorrência:

$$u^{(n+s)} = u^{(n+s-1)} + h \sum_{j=0}^{s-1} \beta_j f(t^{(n+j)},u^{(n+j)}) \tag{10.254}$$

Observe que a integral envolvida no cálculo dos coeficientes β_j em (10.253) pode ser simplificada via a mudança de variáveis $t = t^{(n+s-1)} + h\tau$:

$$\beta_j = \int_0^1 \prod_{k=0,k\neq j}^{s-1} \frac{\tau + s - k - 1}{j - k} d\tau \tag{10.255}$$

$$= \frac{(-1)^{s-j-1}}{j!(s-j-1)!} \int_0^1 \prod_{k=0,k\neq j}^{s-1} (\tau + s - k - 1)d\tau \tag{10.256}$$

$$= \frac{(-1)^{s-j-1}}{j!(s-j-1)!} \int_0^1 \prod_{k=0,k\neq s-j-1}^{s-1} (\tau + k)d\tau \tag{10.257}$$

Observação 10.10.1 (Ordem do método de Adasm-Bashforth)**.** Da teoria de interpolação (ver capítulos 6 e 9), temos que o erro de aproximação da integral de uma função suficientemente suave por um polinômio interpolador em s pontos é de ordem $s+1$. Assim, o erro local de truncamento do método de Adams-Bashforth com s passos é $s+1$ e, portanto, o erro global de truncamente é de ordem s.

Exemplo 10.10.1. Calcule os coeficientes de Adams-Bashforth para $s = 2$ e, depois, construa seu processo iterativo.

$$\beta_0 = -\int_0^1 (\tau + 2 - 1 - 1)d\tau = -\frac{1}{2} \tag{10.258}$$

$$\beta_1 = \int_0^1 (\tau + 2 - 0 - 1)d\tau = \frac{3}{2} \tag{10.259}$$

O processo iterativo é dado por:

$$y^{(n+2)} = y^{(n)} + \frac{h}{2}\left[3f\left(t^{(n+1)},u(t^{(n+1)})\right) - f\left(t^{(n)},u(t^{(n)})\right)\right] \tag{10.260}$$

Exemplo 10.10.2. Calcule os coeficientes e de Adams-Bashforth para $s = 3$ e, depois, construa o processo iterativo.

$$\beta_0 = \frac{1}{2}\int_0^1 (\tau + 3 - 1 - 1)\cdot(\tau + 3 - 2 - 1)d\tau = \frac{5}{12} \tag{10.261}$$

$$\beta_1 = -\int_0^1 (\tau + 3 - 0 - 1)\cdot(\tau + 3 - 2 - 1)d\tau = -\frac{4}{3} \tag{10.262}$$

$$\beta_2 = \frac{1}{2}\int_0^1 (\tau + 3 - 0 - 1)\cdot(\tau + 3 - 1 - 1)d\tau = \frac{23}{12} \tag{10.263}$$

O processo iterativo é dado por:

$$y^{(n+3)} = y^{(n)} + \frac{h}{12}\left[23f\left(t^{(n+2)},u(t^{(n+2)})\right) - 16f\left(t^{(n+1)},u(t^{(n+1)})\right) + 5f\left(t^{(n)},u(t^{(n)})\right)\right] \tag{10.264}$$

Observação 10.10.2. Os coeficientes do método de Adams-Bashforth de ordem s podem, alternativamente, ser obtidos exigindo que o sistema seja exato para $f(t,u) = t^0$, $f(t,u) = t^1$, $f(t,u) = t^2$, ..., $f(t,u) = t^{s-1}$.

Exemplo 10.10.3. Obtenha o método de Adams-Bashforth para $s = 4$ como

$$u^{(n+4)} = u^{(n+3)} + \int_{t^{(n+3)}}^{t^{(n+4)}} f(t,u(t))dt \tag{10.265}$$

$$u^{(n+4)} = u^{(n+3)} + h\sum_{m=0}^{3} b_m f^{(n+m)} \tag{10.266}$$

$$u^{(n+4)} = u^{(n+3)} + h\left[b_3 f^{(n+3)} + b_2 f^{(n+2)} + b_1 f^{(n+1)} + b_0 f^{(n)}\right] \tag{10.267}$$

Para isso devemos obter $[b_3,b_2,b_1,b_0]$ tal que o método seja exato para polinômios até ordem 3. Podemos obter esses coeficientes de maneira análoga a obter os coeficientes de um método para integração.

Supondo que os nós $t^{(k)}$ estejam igualmente espaçados, e para facilidade dos cálculos, como o intervalo de integração é $[t^{(n+3)},t^{(n+4)}]$, translade $t^{(n+3)}$ para a origem tal que $[t^{(n)},t^{(n+1)},\ldots,t^{(n+4)}] = [-3h,-2h,-h,0,h]$.

Considere a base $[1,t,t^2,t^3]$ e substitua $f(t)$ por cada um dos elementos desta basa, obtendo:

$$\int_0^h 1\,dt = h = h\,[b_0(1)+b_1(1)+b_2(1)+b_3(1)] \tag{10.268}$$

$$\int_0^h t\,dt = \frac{h^2}{2} = h\,[b_0(0)+b_1(-h)+b_2(-2h)+b_3(-3h)] \tag{10.269}$$

$$\int_0^h t^2\,dt = \frac{h^3}{3} = h\left[b_0(0)^2+b_1(-h)^2+b_2(-2h)^2+b_3(-3h)^2\right] \tag{10.270}$$

$$\int_0^h t^3\,dt = \frac{h^4}{4} = h\left[b_0(0)^3+b_1(-h)^3+b_2(-2h)^3+b_3(-3h)^3\right] \tag{10.271}$$

que pode ser escrito na forma matricial

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & -1 & -2 & -3 \\ 0 & 1 & 4 & 9 \\ 0 & -1 & -8 & -27 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1/2 \\ 1/3 \\ 1/4 \end{pmatrix} \tag{10.272}$$

Resolvendo o sistema obtemos

$$[b_0,b_1,b_2,b_3] = \left[-\frac{9}{24},\frac{37}{24},-\frac{59}{24},\frac{55}{24}\right] \tag{10.273}$$

fornecendo o **método de Adams-Bashforth de** 4 **estágios**

$$u^{(n+4)} = u^{(n+3)} + \tfrac{h}{24}[55f^{(n+3)} - 59f^{(n+2)} + 37f^{(n+1)} - 9f^{(n)}] \tag{10.274}$$

A tabela abaixo mostra as coeficientes do método de Adams-Bashforth para até 8 passos.

1	1							
2	$-\frac{1}{2}$	$\frac{3}{2}$						
3	$\frac{5}{12}$	$-\frac{4}{3}$	$\frac{23}{12}$					
4	$-\frac{3}{8}$	$\frac{37}{24}$	$-\frac{59}{24}$	$\frac{55}{24}$				
5	$\frac{251}{720}$	$-\frac{637}{360}$	$\frac{109}{30}$	$-\frac{1387}{360}$	$\frac{1901}{720}$			
6	$-\frac{95}{288}$	$\frac{959}{480}$	$-\frac{3649}{720}$	$\frac{4991}{720}$	$-\frac{2641}{480}$	$\frac{4277}{1440}$		
7	$\frac{19087}{60480}$	$-\frac{5603}{2520}$	$\frac{135713}{20160}$	$-\frac{10754}{945}$	$\frac{235183}{20160}$	$-\frac{18637}{2520}$	$\frac{198721}{60480}$	
8	$-\frac{5257}{17280}$	$\frac{32863}{13440}$	$-\frac{115747}{13440}$	$\frac{2102243}{120960}$	$-\frac{296053}{13440}$	$\frac{242653}{13440}$	$-\frac{1152169}{120960}$	$\frac{16083}{4480}$

Observação 10.10.3. Note que os métodos de múltiplo passo requerem o conhecimento dos s valores previamente computados para calcular $y^{(n+s)}$. Assim, para inicializar um algoritmo com mais de um passo, não é suficiente conhecer a condição inicial. Usualmente, calcula-se os primeiros s passos usando um algoritmo de passo simples da mesma ordem do método múltiplo passo a ser aplicado.

Exercícios resolvidos

ER 10.10.1. Resolva numericamente o problema de valor inicial dado por:

$$y'(t) = \sqrt{1+y(t)} \tag{10.275}$$
$$y(0) = 0 \tag{10.276}$$

aplicando o método de Adams-Bashforth de dois passos e inicializando o método através do método de Euler modificado. Calcule o valor de $y(1)$ com passo de tamanho $h = 0{,}1$.

Solução. Primeiro observamos que o processo resursivo do método de Adams é dado por:

$$y^{(n+2)} = y^{(n+1)} + \frac{h}{2}\left[3f\left(t^{(n+1)},u(t^{(n+1)})\right) - f\left(t^{(n)},u(t^{(n)})\right)\right],\ n = 1,2,\ldots \tag{10.277}$$

O valor inicial é dado por $y^{(1)} = 0$. No entanto, para inicializar o método, precisamos calcular $y^{(2)}$, para tal, aplicamos o método de Euler modificado:

$$k_1 = \sqrt{1+0} = 1 \tag{10.278}$$
$$k_2 = \sqrt{1+0{,}1} = \sqrt{1{,}1} \approx 1{,}0488088 \tag{10.279}$$
$$y^{(2)} = \frac{0{,}1}{2}\left(1+1{,}0488088\right) = 0{,}10244044 \tag{10.280}$$

Aplicando o método de Adams-Bashforth, obtemos:

$$y^{(1)} = 0 \tag{10.281}$$
$$y^{(2)} = 0{,}10244044 \tag{10.282}$$
$$y^{(3)} = 0{,}20993619 \tag{10.283}$$
$$y^{(4)} = 0{,}32243326 \tag{10.284}$$
$$y^{(5)} = 0{,}43993035 \tag{10.285}$$
$$y^{(6)} = 0{,}56242745 \tag{10.286}$$
$$y^{(7)} = 0{,}68992455 \tag{10.287}$$
$$y^{(8)} = 0{,}82242165 \tag{10.288}$$
$$y^{(9)} = 0{,}95991874 \tag{10.289}$$
$$y^{(10)} = 1{,}10241584 \tag{10.290}$$
$$y^{(11)} = 1{,}24991294 \tag{10.291}$$

◊

A seguinte rotina implementa o método:

```
def f(t,u):
return np.sqrt(1+u)

def adams_bash_2(h,Tmax,u1):
dim=np.size(u1)
itmax=np.int(Tmax/h)
u=np.empty((itmax+1,dim))
u[0,:]=u1

#inicaliza com RK2
k1 = f(0,   u[0,:])
k2 = f(h, u[0,:] + k1* h)
u[1,:] = u[0,:] + (k1+k2)* h/2

fn_0=k1
for i in np.arange(0,itmax-1):
t=(i+1)*h
fn_1 = f(t,   u[i+1,:])
u[i+2,:] = u[i+1,:] + h*(-.5*fn_0 + 1.5*fn_1)
fn_0=fn_1
return u




u0=0
h=1e-1
Tmax=1
u=adams_bash_2(h,Tmax,u0)

print u
```

Exercícios

Em construção ... Gostaria de participar na escrita deste livro? Veja como em:
https://www.ufrgs.br/reamat/participe.html

10.11 Método de Adams-Moulton

O método de Adams-Moulton, assim como o método de Adams-Bashforth, é um método de passo múltiplo. A diferença entre estes dois métodos é que Adams-Bashforth é explícito, enquanto Adams-Moulton é implícito, isto é, os valores de $f\,(t,u))$, nos passos n, $n+1$, ..., $n+s-1$ e, inclusive, $n+s$ são utilizados ao calcular f em $t^{(n+s)}$.

Considere o problema de valor inicial

$$u'(t) = f(t,u(t)) \tag{10.292}$$
$$u(t_0) = a \tag{10.293}$$

Integrando a equação diferencial no intervalo $[t^{(n+s-1)},t^{(n+s)}]$, obtemos:

$$u^{(n+s)} = u^{(n+s-1)} + \int_{t^{(n+s-1)}}^{t^{(n+s)}} f(t,u(t))dt \tag{10.294}$$

Agora o integrando em (10.294) é aproximado pelo polinômio que interpola $f(t^{(k)},u^{(k)})$ para $k = n, n+1, n+2, \ldots, n+s$, isto é:

$$u^{(n+s)} = u^{(n+s-1)} + \int_{t^{(n+s-1)}}^{t^{(n+s)}} p(t)dt \tag{10.295}$$

onde $p(t)$ é polinômio de grau s dado na forma de Lagrange por:

$$p(t) = \sum_{j=0}^{s} \left[f(t^{(n)},u^{(n)}) \prod_{k=0,k\neq j}^{s} \frac{t-t^{(n+k)}}{t^{(n+j)}-t^{(n+k)}} \right] \tag{10.296}$$

Agora observamos que

$$\int_{t^{(n+s-1)}}^{t^{(n+s)}} p(t)dt = h\sum_{j=0}^{s} \beta_j f(t^{(n+j)},u^{(n+j)}) \tag{10.297}$$

onde

$$\beta_j = \frac{1}{h}\int_{t^{(n+s-1)}}^{t^{(n+s)}} \prod_{k=0,k\neq j}^{s} \frac{t-t^{(n+k)}}{t^{(n+j)}-t^{(n+k)}} dt \tag{10.298}$$

Aplicando a mudança de variáveis $t = t^{(n+s-1)} + h\tau$, temos:

$$\beta_j = \int_0^1 \prod_{k=0,k\neq j}^{s} \frac{\tau+s-k-1}{j-k} dt \tag{10.299}$$

$$= \frac{(-1)^{s-j}}{j!(s-j)!} \int_0^1 \prod_{k=0,k\neq j}^{s} (\tau+s-k-1) d\tau \tag{10.300}$$

$$= \frac{(-1)^{s-j}}{j!(s-j)!} \int_0^1 \prod_{k=0,k\neq s-j-1}^{s} (\tau+k-1) d\tau \tag{10.301}$$

Assim, obtemos a relação de recorrência:

$$u^{(n+s)} = u^{(n+s-1)} + h\sum_{j=0}^{s} \beta_j f(t^{(n+j)},u^{(n+j)}) \tag{10.302}$$

Observação 10.11.1. Os coeficientes do método de Adams-Moulton de s passos podem, alternativamente, ser obtidos exigindo que o sistema seja exato para $f(t,u) = t^0$, $f(t,u) = t^1$, $f(t,u) = t^2$, ..., $f(t,u) = t^s$.

Exemplo 10.11.1. Obtenha o método de Adams-Moulton para $s = 3$ como

$$u^{(n+3)} = u^{(n+2)} + \int_{t^{(n+3)}}^{t^{(n+4)}} f(t,u(t))\, dt \tag{10.303}$$

$$u^{(n+3)} = u^{(n+2)} + h\sum_{m=0}^{3} b_m f^{(n+m)} \tag{10.304}$$

$$u^{(n+3)} = u^{(n+2)} + h\left[b_3 f^{(n+3)} + b_2 f^{(n+2)} + b_1 f^{(n+1)} + b_0 f^{(n)}\right] \tag{10.305}$$

Para isso devemos obter $[b_3,b_2,b_1,b_0]$ tal que o método seja exato para polinômios até ordem 3. Podemos obter esses coeficientes de maneira análoga a obter os coeficientes de um método para integração.

Supondo que os nós t_k estejam igualmente espaçados, e para facilidade dos cálculos, como o intervalo de integração é $[t^{(n+2)},t^{(n+3)}]$, translade $t^{(n+2)}$ para a origem tal que $[t^{(n)},t^{(n+1)},\ldots,t^{(n+3)}] = [-2h,-h,0,h]$.

Considere a base $[1,t,t^2,t^3]$ e substitua $f(t)$ por cada um dos elementos da base,obtendo:

$$\int_0^h 1\, dt = h = h(b_0(1) + b_1(1) + b_2(1) + b_3(1)) \tag{10.306}$$

$$\int_0^h t\, dt = \frac{h^2}{2} = h(b_0(h) + b_1(0) + b_2(-h) + b_3(-2h)) \tag{10.307}$$

$$\int_0^h t^2\, dt = \frac{h^3}{3} = h(b_0(h)^2 + b_1(0)^2 + b_2(-h)^2 + b_3(-2h)^2) \tag{10.308}$$

$$\int_0^h t^3\, dt = \frac{h^4}{4} = h(b_0(h)^3 + b_1(0)^3 + b_2(-h)^3 + b_3(-2h)^3) \tag{10.309}$$

que pode ser escrito na forma matricial

$$\begin{pmatrix} 1 & 0 & 1 & 1 \\ 1 & 0 & -1 & -2 \\ 1 & 0 & 1 & 4 \\ 1 & 0 & -1 & -8 \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{pmatrix} = \begin{pmatrix} 1 \\ 1/2 \\ 1/3 \\ 1/4 \end{pmatrix} \tag{10.310}$$

Resolvendo o sistema obtemos

$$[b_0,b_1,b_2,b_3] = \left[\frac{1}{24}, -\frac{5}{24}, \frac{19}{24}, \frac{9}{24}\right] \tag{10.311}$$

fornecendo a regra

$$u_{n+3} \quad = u_{n+2} + \tfrac{h}{24}[9f_{n+3} + 19f_{n+2} - 5f_{n+1} + f^{(n)}] \tag{10.312}$$

A tabela abaixo mostra as coeficientes do método de Adams-Moulton para até oito passos.

1	1							
2	$\frac{1}{2}$	$\frac{1}{2}$						
3	$-\frac{1}{12}$	$\frac{2}{3}$	$\frac{5}{12}$					
4	$\frac{1}{24}$	$-\frac{5}{24}$	$\frac{19}{24}$	$\frac{3}{8}$				
5	$-\frac{19}{720}$	$\frac{53}{360}$	$-\frac{11}{30}$	$\frac{323}{360}$	$\frac{251}{720}$			
6	$\frac{3}{160}$	$-\frac{173}{1440}$	$\frac{241}{720}$	$-\frac{133}{240}$	$\frac{1427}{1440}$	$\frac{95}{288}$		
7	$-\frac{863}{60480}$	$\frac{263}{2520}$	$-\frac{6737}{20160}$	$\frac{586}{945}$	$-\frac{15487}{20160}$	$\frac{2713}{2520}$	$\frac{19087}{60480}$	
8	$\frac{275}{24192}$	$-\frac{11351}{120960}$	$\frac{1537}{4480}$	$-\frac{88547}{120960}$	$\frac{123133}{120960}$	$-\frac{4511}{4480}$	$\frac{139849}{120960}$	$\frac{5257}{17280}$

Exemplo 10.11.2. O esquema iterativo de Adams-Moulton com três passos, isto é, $s = 2$ é dado na forma:

$$u^{(n+2)} = u^{(n+1)} + \frac{h}{12}\left[5f\left(t^{(n+2)},u(t^{(n+2)})\right) + 8f\left(t^{(n+1)},u(t^{(n+1)})\right) - f\left(t^{(n)},u(t^{(n)})\right)\right] \tag{10.313}$$

Exercícios resolvidos

ER 10.11.1. Resolva o problema de valor inicial dado por:

$$u'(t) = -2u(t) + te^{-t} \tag{10.314}$$

$$u(0) = -1 \tag{10.315}$$

via Adams-Moulton com $s = 2$ (três passos) com $h = 0{,}1$ e $h = 0{,}01$ e compare com a solução exata dada por $u(t) = (t-1)e^{-t}$ nos instantes $t = 1$ e $t = 2$. Inicialize com Euler modificado.

Solução. Primeiro observamos que $f(u,t) = -2u + te^{-t}$ e que o esquema de Adams-Moulton pode ser escrito como:

$$u^{(n+2)} = u^{(n+1)} + \frac{h}{12}\left[5f\left(t^{(n+2)},u(t^{(n+2)})\right) + 8f\left(t^{(n+1)},u(t^{(n+1)})\right) - f\left(t^{(n)},u(t^{(n)})\right)\right] \tag{10.316}$$

de forma que:

$$u^{(n+2)} = u^{(n+1)} + \frac{h}{12}\left[8f\left(t^{(n+1)},u(t^{(n+1)})\right) - f\left(t^{(n)},u(t^{(n)})\right)\right] + \frac{5h}{12}f\left(t^{(n+2)},u(t^{(n+2)})\right) \tag{10.317}$$

$$= u^{(n)} + \frac{h}{12}\left[8f^{(n+1)} - f^{(n)}\right] + \frac{5h}{12}\left(t^{(n+2)}e^{t^{(n+2)}} - 2u(t^{(n+2)})\right) \tag{10.318}$$

Assim:

$$\left(1 + \frac{5h}{6}\right)u^{(n+2)} = u^{(n+1)} + \frac{h}{12}\left[8f^{(n+1)} - f^{(n)}\right] + \frac{5h}{12}t^{(n+2)}e^{-t^{(n+2)}} \tag{10.319}$$

Os valores obtidos são:

	t=1	t=2
h=0,1	-0,000223212480142	0,135292280956
h=0,01	-2,02891229566e-07	0,135335243537
Exato	0	0,135335283237

A seguinte rotina implementa a recursão:

```
## resolve u'(t)=l*u(t) + g

def g(t):
return t*np.exp(-t)



u0=-1
h=1e-2
Tmax=2
itmax=np.int(Tmax/h)

u=np.empty(itmax+1)
fn=np.empty(itmax+1)

u[0]=u0
l=-2
```

```
#Iniciliza com Euler modificado
k1= l*u[0] + g(0)
k2= l*(u[0]+h*k1) + g(h)
u[1]= u[0]+ h *(k1+k2)/2

fn[0]= k1
fn[1]= l*u[1] + g(h)



for n in np.arange(0,itmax-1):
gn2=g((n+2)*h)

u[n+2]= (u[n+1] + h/12*(8*fn[n+1]-fn[n]) + 5*h/12*gn2 ) / (1+5*h/6)
fn[n+2]=l*u[n+2]+gn2

for n in np.arange(0,itmax+1):
print h*n,u[n], (h*n-1)*np.exp(-h*n)
```

◊

ER 10.11.2. Repita o Problema 10.10.1 pelo método de Adams-Moulton, isto, é resolva numericamente o problema de valor inicial dado por:

$$y'(t) = \sqrt{1+y(t)}, \tag{10.320}$$
$$y(0) = 0, \tag{10.321}$$

aplicando o método de Adams-Moulton de dois passos. Calcule o valor de $y(1)$ com passo de tamanho $h = 0{,}1$.

Solução. Primeiro observamos que o processo resursivo do método de Adams é dado por:

$$y^{(n+1)} = y^{(n)} + \frac{h}{2}\left[f\left(t^{(n+1)},u(t^{(n+1)})\right) + f\left(t^{(n)},u(t^{(n)})\right)\right],\ n = 1,2,\ldots \tag{10.322}$$

O valor inicial é dado por $y^{(1)} = 0$. Primeiramente, precisamos isolar $y^{(x+1)}$ na Equação 10.322:

$$y^{(n+1)} = y^{(n)} + \frac{h}{8}\sqrt{h^2+16+16y^{(n)}+8h\sqrt{1+y^{(n)}}} + \frac{h}{2}\sqrt{1+y^{(n)}} + \frac{h^2}{8},\ n = 1,2,\ldots \tag{10.323}$$

$$
\begin{aligned}
y^{(1)} &= 0 && (10.324)\\
y^{(2)} &= 0{,}1025 && (10.325)\\
y^{(3)} &= 0{,}21 && (10.326)\\
y^{(4)} &= 0{,}3225 && (10.327)\\
y^{(5)} &= 0{,}44 && (10.328)\\
y^{(6)} &= 0{,}5625 && (10.329)\\
y^{(7)} &= 0{,}69 && (10.330)\\
y^{(8)} &= 0{,}8225 && (10.331)\\
y^{(9)} &= 0{,}96 && (10.332)\\
y^{(10)} &= 1{,}1025 && (10.333)\\
y^{(11)} &= 1{,}25 && (10.334)
\end{aligned}
$$

$\Diamond$

ER 10.11.3. Resolva o problema de valor inicial dado por

$$
\begin{aligned}
y'(t) &= y^3 - y + t, && (10.335)\\
y(0) &= 0, && (10.336)
\end{aligned}
$$

aplicando o método de Adams-Moulton de dois passos. Calcule o valor de $y(1)$ com passo de tamanho $h = 0{,}1$ e $h = 0{,}01$. Primeiro observamos que o processo resursivo do método de Adams é dado por:

$$
\begin{aligned}
y^{(n+1)} &= y^{(n)} + \frac{h}{2}\left[f\left(t^{(n+1)},u(t^{(n+1)})\right) + f\left(t^{(n)},u(t^{(n)})\right)\right],\ n = 1,2,\ldots && (10.337)\\
y^{(1)} &= 0 && (10.338)
\end{aligned}
$$

Observamos que o problema de isolar $y^{(n+1)}$ pode ser escrito como

$$
y^{(n+1)} - \frac{h}{2} f\left(t^{(n+1)},u(t^{(n+1)})\right) = y^{(n)} + \frac{h}{2} f\left(t^{(n)},u(t^{(n)})\right) \qquad (10.339)
$$

O termo da esquerda é uma expressão não linear em $y^{(n+1)}$ e o termo da direita é conhecido, isto é, pode ser calculado com base nos valores anteriormente calculados. Devemos, então, escolher um método numérico de solução de equações algébricas não lineares (veja capítulo 3) como o método de Newton visto na Seção 3.4 para resolver uma equação do tipo:

$$
u - \frac{h}{2} f(t^{(n+1)},u) = a, \qquad (10.340)
$$

isto é:

$$u - \frac{h}{2}\left(u^3 - u + t^{(n+1)}\right) = a, \tag{10.341}$$

com $a = y^{(n)} + \frac{h}{2} f\left(t^{(n)}, u(t^{(n)})\right)$ Os valores obtidos são: 0,37496894 e 0,37512382 quando o método é inicializadom com Euler melhorado. A seguinte rotina implementa a recursão:

```
def f(t,u):
return u**3-u+t

def resolve(u,t,a,h,beta): #resolve equacao nao-linear por metodo de Newton
ctrl=2
cont=0
while (ctrl>0):
cont=cont+1
residuo=u-beta*h*(u**3-u+t)-a
derivada=1-beta*h*(3*u**2-1)
u1=u-residuo/derivada
if np.abs(u1-u)<derivada*1e-10:
ctrl=ctrl-1
u=u1
# print cont
return u


def adams_moulton_2(h,Tmax,u1):
itmax=np.int(Tmax/h)
u=np.empty((itmax+1,1))
u[0]=u1

for i in np.arange(0,itmax):
t=i*h
fn=f(t,u[i])
a=u[i] + h*fn/2
u[i+1]=resolve(u[i]+ h*fn,t+h,a,h,1/2)
return u



u0=0
h=1e-2
```

```
Tmax=1
itmax=np.int(Tmax/h)

u=adams_moulton_2(h,Tmax,u0)
print u[itmax]
```

Exercícios

E 10.11.1. Encontre o método de Adams-Moulton para $s = 0$.

E 10.11.2. Encontre o método de Adams-Moulton para $s = 1$.

E 10.11.3. Repita o Problema 10.11.3 usando Adams-Moulton com 3 passos e inicilizando com Runge-Kutta quarta ordem clássico.

10.12 Método de Adams-Moulton para sistemas lineares

Esquemas implícitos como o de Adams-Moulton apresentam a dificuldade adicional de necessitar do valor de $f(t^{(n+1)},u^{(n+1)})$ para calcular o valor de $u^{(n+1)}$. Pelo menos para sistemas lineares, o método pode ser explicitado. Seja o seguinte problema de valor inicial linear:

$$\begin{aligned} u'(t) &= Au(t) + g(t), && (10.344)\\ u(t^{(1)}) &= a. && (10.345) \end{aligned}$$

Onde $u(t)$ é um vetor de n entradas e A é uma matriz $n \times n$.

Considere agora o esquema de Adams-Moulton dado na Equação (10.302) com $f(t,u) = Au + g(t)$:

$$u^{(n+s)} = u^{(n+s-1)} + h\sum_{j=0}^{s} \beta_j \left[Au^{(n+j)} + g(t^{(n+j)})\right] \quad (10.346)$$

o que pode ser escrito como:

$$\begin{aligned} (I_d - h\beta_s A)\, u^{(n+s)} &= u^{(n+s-1)} + h\sum_{j=0}^{s-1} \beta_j \left[Au^{(n+j)} + g(t^{(n+j)})\right] \\ &+ h\beta_s g(t^{(n+s)}) \quad (10.347) \end{aligned}$$

onde I_d é matriz identidade $n \times n$. O sistema linear envolvido em (10.347) pode ser resolvido sempre que $I_d - h\beta_s A$ for inversível, o que sempre acontece quando h é suficientemente pequeno.

Exercícios resolvidos

Esta seção carece de exercícios resolvidos. Participe da sua escrita.
Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

Exercícios

Esta seção carece de exercícios. Participe da sua escrita.
Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

10.13 Estratégia preditor-corretor

Esquemas implícitos como o de Adams-Moulton (Seção 10.11) e o de Runge-Kutta (Seção 10.8), embora úteis para resolver problemas rígidos (ver Seção 10.14), apresentam a dificuldade de necessitar do valor de $f(t^{(n+1)},u^{(n+1)})$ para calcular o valor de $u^{(n+1)}$, exigindo a solução de uma equação algébrica a cada passo. Uma forma de aproximar o comportamento de um método ímplicito através de um esquema implícito consiste em aplicar a, assim chamada, **estratégia preditor-corretor**.

Os métodos do tipo preditor-corretor empregam um esquema explícito para **predizer** o valor de $u^{(n+1)}$ e, depois, um método implícito para recalcular, isto é, **corrigir** $u^{(n+1)}$.

Exemplo 10.13.1. Considere o método de Euler implícito (ver 10.8.1) aplicado para resolver o problema de valor inicial

$$u'(t) = f(t,u(t)) \tag{10.348}$$
$$u(t^{(1)}) = a \tag{10.349}$$

cujo processo iterativo é dado por

$$u^{(n+1)} = u^{(n)} + hf(t^{(n+1)},u^{(n+1)}). \tag{10.350}$$

Agora aplicamos o método de Euler (ver 10.2) para predizer $u^{(n+1)}$:

$$u^{(n+1)} = u^{(n)} + hf(t^{(n)},u^{(n)}). \tag{10.351}$$

E agora, retornamos ao método de Euler implícito:

$$u^{(n+1)} = u^{(n)} + hf(t^{(n+1)},\tilde{u}^{(n+1)}). \tag{10.352}$$

Desta forma, a estratégia preditor-corretor aplicada ao método de Euler implícito com predição via método de Euler produz o método de Euler melhorado, ver 10.3, isto é:

$$\begin{aligned} \tilde{u}^{(n+1)} &= u^{(n)} + hf(t^{(n)},u^{(n)}), & (10.353)\\ u^{(n+1)} &= u^{(n)} + hf(t^{(n+1)},\tilde{u}^{(n+1)}). & (10.354) \end{aligned}$$

Exemplo 10.13.2. Considere o método de trapezoidal (ver 10.8.2) aplicado para resolver o problema de valor inicial

$$\begin{aligned} u'(t) &= f(t,u(t)) & (10.355)\\ u(t^{(1)}) &= a & (10.356) \end{aligned}$$

cujo processo iterativo é dado por

$$u^{(n+1)} = u^{(n)} + \frac{h}{2}\left[f(t^{(n)},u^{(n)}) + f(t^{(n+1)},u^{(n+1)})\right]. \tag{10.357}$$

Agora aplicamos o método de Euler (ver 10.2) para predizer $u^{(n+1)}$:

$$\tilde{u}^{(n+1)} = u^{(n)} + hf(t^{(n)},u^{(n)}). \tag{10.358}$$

E agora, retornamos ao método trapezoidal para obter:

$$\begin{aligned} \tilde{u}^{(n+1)} &= u^{(n)} + hf(t^{(n)},u^{(n)}), & (10.359)\\ u^{(n+1)} &= u^{(n)} + \frac{h}{2}\left[f(t^{(n)},u^{(n)}) + f(t^{(n+1)},\tilde{u}^{(n+1)})\right]. & (10.360) \end{aligned}$$

Exemplo 10.13.3. Considere o método de Adams-Moulton de segunda ordem (ver 10.11) aplicado para resolver o problema de valor inicial

$$\begin{aligned} u'(t) &= f(t,u(t)) & (10.361)\\ u(t^{(1)}) &= a & (10.362) \end{aligned}$$

cujo processo iterativo é dado por

$$u^{(n+1)} = u^{(n)} + \frac{h}{2}\left[f(t^{(n)},u^{(n)}) + f(t^{(n+1)},u^{(n+1)})\right]. \tag{10.363}$$

Agora aplicamos o método de Adams-Bashforth de segunda ordem (ver 10.10) para predizer $u^{(n+1)}$:

$$\tilde{u}^{(n+1)} = u^{(n)} + \frac{h}{2}\left[-f(t^{(n-1)},u^{(n-1)}) + 3f(t^{(n)},u^{(n)})\right]. \tag{10.364}$$

Assim, obtemos o seguinte método:

$$\tilde{u}^{(n+1)} = u^{(n)} + \frac{h}{2}\left[-f(t^{(n-1)},u^{(n-1)}) + 3f(t^{(n)},u^{(n)})\right], \quad (10.365)$$

$$u^{(n+1)} = u^{(n)} + \frac{h}{2}\left[f(t^{(n)},u^{(n)}) + f(t^{(n+1)},\tilde{u}^{(n+1)})\right]. \quad (10.366)$$

10.13.1 Exercícios resolvidos

Esta seção carece de exercícios resolvidos. Participe da sua escrita.
Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

Exercícios

E 10.13.1. Construa o esquema preditor corretor combinando Adams-Moulton de quarta ordem e Adams-Bashforth de quarta ordem.

E 10.13.2. Seja o problema de valor inicial dado por:

$$u'(t) = \sqrt{u(t)+1} \quad (10.369)$$

$$u(0) = 0 \quad (10.370)$$

Resolva numericamente esse problema pelo método de Adams-Bashforth de segunda ordem e pelo método preditor corretor combinando Adams-Bashforth de segunda order com Adams-Moulton de segunda ordem. Compare a solução obtida para $t = 10$ com a solução exata dada por:

$$u(t) = \frac{t^2}{4} + t. \quad (10.371)$$

Inicialize os métodos empregando Runge-Kuta de segunda ordem.

10.14 Problemas rígidos

Esta seção (ou subseção) está sugerida. Participe da sua escrita. Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

10.15 Validação e "Benchmarking"

Toda metodologia numérica deve ser validada ao ser aplicada para resolver um problema. A validação aumenta a confiabilidade na qualidade dos resultados obtidos. A validação procura detectar erros de implementação, características numéricas espúrias não prevista em projeto, como propagação catastrófica erros de arrendodamento, inadequação do método para o problema proposto etc. A principal técnica de validação consiste em comparar a solução produzida com soluções de alta qualidade e confiabilidade, os chamados *benchmarks*. Quando um *benchmark* não estiver disponível, ainda se dispõe de algumas técnicas para avaliar a qualidade do método. Existe uma hierarquia das técnicas de validação conforme listados a seguir:

1. Expressão analítica: Testar o código com problemas que admitem soluções analíticas constitui a forma mais confiável para validar o esquema usado. Expressões analíticas são expressões matemáticas das seguintes formas:

   Expressão aritmética: Expressões envolvendo apenas um número finito de operações aritméticas elementares (soma, subtração, multiplicação e divisão) e números inteiros. Ex: $u(t) = \frac{t^2+1}{3t-4}$ ou $u(t) = t^3 + \frac{3}{4}$.

   Expressão algébrica: Expressões envolvendo apenas um número finito de operações aritméticas elementares e expoentes fracionários. Ex: $u(t) = t^2 + \sqrt{t}$ ou $u(t) = t^{3/2} + \sqrt{2}$.

   Expressão forma-fechada: Expressões envolvendo apenas um número finito de operações aritméticas elementares, expoentes reais, logaritmos, exponencias, funções trigonométricas e funções trigonométricas inversas. Ex: $u(t) = \ln(1 + t^\pi)$, $u(t) = e^{-t}\,\mathrm{sen}\,(t)$ ou $u(t) = \tan^{-1}(t + 1)$.

   Expressão envolvendo funções especiais: Além das operações e funções acima, são permitidas funções especiais, como a função gama, funções de Bessel, séries de taylor, series de Fourier e outras séries envolvendo funções elementares e especiais.

2. Expressão matemática semi-analítica: Expressão matemática envolvendo, além das operações e funções acima, outros processos de limite, como derivação e integração. Ex: $u(t) = \int_0^1 \log|t - x|x^t dx$.

3. Solução numérica com reformulação analítica prévia: Neste caso, não se dispõe de uma expressão matemática para a solução, mas pode-se comparar resultado produzido pelo método numérico com outro problema numérico cuja solução é a mesma e pode ser obtida por outra metodologia numérica mais confiável.

4. Benchmark puramente numérico: Um *benchmark* puramente numérico é uma aproximação numérica para a solução de um problema muito bem estabelecida e de alta confiabilidade. Os benchmarks numéricos normalmente são produzidos comparando diversos métodos numéricos diferentes e independentes e com grande refinamento.

5. Validação por comparação: Quando não se dispõe de *benchmarks*, ainda se pode comparar o resultado obtido com outros métodos numéricos. Em caso de divergência, pode ser bastante difícil dissernir qual método produz melhores resultados.

6. Convergência numérica: Este é o teste mais rudimentar que se aplica a métodos e numéricos e consiste em comparar o resultados produzidos com diferentes malhas de cálculo diferentes. Espera-se que o refino da malha produza soluções que convergem para a soluçao exata. Resultados muito próximos entre refinos sugere qualidade nos resultados.

Exemplo 10.15.1 (Expressão analítica)**.** A solução do problema de valor inicial estudado no Exercício Resolvido 10.2.1 dado por:

$$\begin{aligned} u'(t) &= -0{,}5u(t) + 2 + t, && (10.372)\\ u(0) &= 8, && (10.373) \end{aligned}$$

admite uma solução em forma de expressão analítica dada por:

$$u(t) = 2t + 8e^{-t/2}. \tag{10.374}$$

Exemplo 10.15.2 (Expressão envolvendo funções especiais)**.** A solução do problema de valor inicial dado por:

$$\begin{aligned} u'(t) &= -u^3(t) + u^2(t) && (10.375)\\ u(0) &= \frac{1}{2} && (10.376) \end{aligned}$$

é dada na forma:

$$u(t) = \frac{1}{1 + W(e^{1-t})} \tag{10.377}$$

onde W é a função de Lambert é a função inversa de $f(y) = ye^y$, onde $y = W(x)$.

Exemplo 10.15.3 (Expressão matemática semi-analítica)**.** A solução do problema de valor inicial dado por:

$$\begin{aligned} u^{(5)} &+ au^{(4)} + bu'''(t) + cu''(t) + du'(t) + u(t) = 1 && (10.378)\\ u^{(4)}(0) &= u'''(0) = u''(0) = u'(0) = u(0) && (10.379) \end{aligned}$$

é dada na forma:

$$u(t) = 1 + Ae^{r_1 t} + Be^{r_2 t} + Ce^{r_3 t} + De^{r_4 t} + Ee^{r_5 t} \tag{10.380}$$

onde r_1, r_2, r_3, r_4 e r_5 são as raízes do polinômio característico

$$p(x) = x^5 + ax^4 + bx^3 + cx^2 + dx + 1, \tag{10.381}$$

cujas raízes, salvo casos particulares, só pode ser obtida por aproximações numéricas.

Exemplo 10.15.4 (Solução numérica com reformulação analítica prévia)**.** A solução do problema de valor inicial dado por:

$$u'(t) = u^3(t) + u^2(t) + u(t) + 1 \tag{10.382}$$
$$u(0) = 0 \tag{10.383}$$

é dada na forma:

$$\ln\left(\frac{(u(t)+1)^2}{u(t)^2+1}\right) + 2\arctan\left(u\left(t\right)\right) = 4t \tag{10.384}$$

Esta analítica estabelece uma relação funcional implícita entre t e $u(t)$, no entanto, é necessário resolver uma equação algébrica não-linear para cada t. Ainda assim, pode ser um excelente benchmark, pois o valor de $u(t)$ é dado explicitamente em função de t, isto é, podemos ver t como uma função de u. Por exemplo, é fácil descobrir que $u = 1$ quando $t = \frac{\ln(2)}{4} + \frac{\pi}{8} \approx 0.5659858768387104$. Além disso tomando o limite $u \to +\infty$, descobrimos que $u(t)$ tende a infinito quando $t \to \frac{\pi}{4}-$.

Exemplo 10.15.5 (Solução numérica com reformulação analítica prévia)**.** A solução do problema de valor inicial dado por:

$$u'(t) = \left[\cos\left(u(t)\right) + u(t)\right]\left(1 + \cos(t)\right) \tag{10.385}$$
$$u(0) = 0 \tag{10.386}$$

é dada na forma:

$$\int_0^{y(t)} \frac{d\tau}{\cos(\tau) + \tau} = t + \operatorname{sen}\left(t\right) \tag{10.387}$$

Esta expressão reformula o problema como uma equação integral em $y(t)$. Esta nova reformulação pode ser bastante útil para produzir resultados de benchmark se fixamos o forma de $y(t)$, usamos uma técnica de quadratura númerica de boa

qualidade para aproximar a integral do lado esquerdo da equação. Por exemplo, escolhendo $t(y) = 100$, temos:

$$\int_0^{100} \frac{d\tau}{\cos(\tau) + \tau} = 5{,}574304717298400 \tag{10.388}$$

Resolvendo a equação algébrica

$$t + \operatorname{sen}(t) = 5{,}574304717298400, \tag{10.389}$$

obtemos:

$$t = 5{,}924938036503083. \tag{10.390}$$

A tabela a seguir mostra os valores de $y(t)$ para $t = 5{,}924938036503083$ obtidos por quatro métodos de quarta-ordem: Runge-Kutta clássico, Adams-Bashforth, preditor-corretor com Adams-Bashforth quarta ordem e Adams-Moulton quarta ordem e Adams-Moulton. Os últimos três métodos foram inicializados com Runge-Kutta clássico de quarta ordem.

	$h = 10^{-1}$	$h = 10^{-2}$	$h = 10^{-3}$	$h = 10^{-4}$
Runge-Kutta 4	95,02737096	99,04376734	99,81705606	99,9925711
Adams-Bashforth 4	94,68537569	99,04349066	99,81705572	99,9925711
Pred. Corr. 4	94,68537569	99,04349066	99,81705572	99,9925711
Adams-Moulton 4	94,71724913	99,04324261	99,81705570	99,9925711

Para $h = 10^{-6}$, todos os quatro métodos produzem o resultado 99,99999287.

10.16 Convergência, consistência e estabilidade

Esta seção (ou subseção) está sugerida. Participe da sua escrita. Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

10.17 Exercícios finais

E 10.17.1. Considere o problema de valor inicial dado por

$$\frac{du(t)}{dt} = -u(t) + e^{-t} \tag{10.391}$$

$$u(0) = 0 \tag{10.392}$$

Resolva analiticamente este problema usando as técnicas elementares de equações diferenciais ordinárias. A seguir encontre aproximações numéricas usando os métodos de Euler, Euler modificado, Runge-Kutta clássico e Adams-Bashforth de ordem 4 conforme pedido nos itens.

a) Construa uma tabela apresentando valores com 7 algarismos significativos para comparar a solução analítica com as aproximações numéricas produzidas pelos métodos sugeridos. Construa também uma tabela para o erro absoluto obtido por cada método numérico em relação à solução analítica. Nesta última tabela, expresse o erro com 2 algarismos significativos em formato científico. Dica: $format('e',8)$ para a segunda tabela.

	0,5	1,0	1,5	2,0	2,5
Analítico					
Euler					
Euler modificado					
Runge-Kutta clássico					
Adams-Bashforth ordem 4					

	0,5	1,0	1,5	2,0	2,5
Euler					
Euler modificado					
Runge-Kutta clássico					
Adams-Bashforth ordem 4					

b) Calcule o valor produzido por cada um desses método para $u(1)$ com passo $h = 0{,}1$, $h = 0{,}05$, $h = 0{,}01$, $h = 0{,}005$ e $h = 0{,}001$. Complete a tabela com os valores para o erro absoluto encontrado.

	0,1	0,05	0,01	0,005	0,001
Euler					
Euler modificado					
Runge-Kutta clássico					
Adams-Bashforth ordem 4					

E 10.17.2. Considere o seguinte modelo para o crescimento de uma colônia de bactérias, baseado na equação logística (ver (10.29))

$$u'(t) = \alpha u(t)\,(A - u(t)) \tag{10.393}$$

onde $u(t)$ indica a densidade de bactérias em unidades arbitrárias na colônia e α e A são constantes positivas. Pergunta-se:

a) Se $A = 10$ e $\alpha = 1$ e $u(0) = 1$, use métodos numéricos para obter aproximação para $u(t)$ em $t = 5 \cdot 10^{-2}$, $t = \cdot 10^{-1}$, $t = 5 \cdot 10^{-1}$ e $t = 1$.

b) Se $A = 10$ e $\alpha = 1$ e $u(0) = 1$, use métodos numéricos para obter tempo necessário para que a população dobre?

c) Se $A = 10$ e $\alpha = 1$ e $u(0) = 4$, use métodos numéricos para obter tempo necessário para que a população dobre?

E 10.17.3. Considere o seguinte modelo para a evolução da velocidade de um objeto em queda:

$$v' = g - \alpha v^2 \tag{10.395}$$

Sabendo que $g = 9{,}8$ e $\alpha = 10^{-2}$ e $v(0) = 0$. Pede-se a velocidade ao tocar o solo e o instante quando isto acontece, dado que a altura inicial era 100.

E 10.17.4. Considere o seguinte modelo para o oscilador não linear de Van der Pol:

$$u''(t) - \alpha(A - u(t)^2)u'(t) + w_0^2 u(t) = 0 \tag{10.396}$$

onde A, α e w_0 são constantes positivas.

a) Encontre a frequência e a amplitude de oscilações quando $w_0 = 1$, $\alpha = .1$ e $A = 10$. (Teste diversas condições iniciais)

b) Estude a dependência da frequência e da amplitude com os parâmetros A, α e w_0. (Teste diversas condições iniciais)

c) Que diferenças existem entre esse oscilador não linear e o oscilador linear?

E 10.17.5. Considere o seguinte modelo para um oscilador não linear:

$$u''(t) - \alpha(A - z(t))u'(t) + w_0^2 u(t) = 0 \tag{10.397}$$

$$Cz'(t) + z(t) = u(t)^2 \tag{10.398}$$

onde A, α, w_0 e C são constantes positivas.

a) Encontre a frequência e a amplitude de oscilações quando $w_0 = 1$, $\alpha = .1$, $A = 10$ e $C = 10$. (Teste diversas condições iniciais)

b) Estude a dependência da frequência e da amplitude com os parâmetros A, α, w_0 e C. (Teste diversas condições iniciais)

E 10.17.6. Considere o seguinte modelo para o controle de temperatura em um processo químico:

$$CT'(t) + T(t) = \kappa P(t) + T_{ext} \quad (10.399)$$
$$P'(t) = \alpha(T_{set} - T(t)) \quad (10.400)$$

onde C, α e κ são constantes positivas e $P(t)$ indica o potência do aquecedor. Sabendo que T_{set} é a temperatura desejada, interprete o funcionamento esse sistema de controle. Faça o que se pede:

a) Calcule a solução quando a temperatura externa $T_{ext} = 0$, $T_{set} = 1000$, $C = 10$, $\kappa = .1$ e $\alpha = .1$. Considere condições iniciais nulas.

b) Quanto tempo demora o sistema para atingir a temperatura 900K?

c) Refaça os dois primeiros itens com $\alpha = 0{,}2$ e $\alpha = 1$

b) Faça testes para verificar a influência de T_{ext}, α e κ na temperatura final.

E 10.17.7. Considere a equação do pêndulo dada por:

$$\frac{d^2\theta(t)}{dt^2} + \frac{g}{l}\operatorname{sen}(\theta(t)) = 0 \quad (10.401)$$

onde g é o módulo da aceleração da gravidade e l é o comprimento da haste.

a) Mostre analiticamente que a energia total do sistema dada por

$$\frac{1}{2}\left(\frac{d\theta(t)}{dt}\right)^2 - \frac{g}{l}\cos(\theta(t)) \quad (10.402)$$

é mantida constante.

b) Resolva numericamente esta equação para $g = 9{,}8m/s^2$ e $l = 1m$ e as seguintes condições iniciais:

i. $\theta(0) = 0{,}5$ e $\theta'(0) = 0$.

ii. $\theta(0) = 1{,}0$ e $\theta'(0) = 0$.

iii. $\theta(0) = 1{,}5$ e $\theta'(0) = 0$.

iv. $\theta(0) = 2{,}0$ e $\theta'(0) = 0$.

v. $\theta(0) = 2{,}5$ e $\theta'(0) = 0$.

vi. $\theta(0) = 3{,}0$ e $\theta'(0) = 0$.

Em todos os casos, verifique se o método numérico reproduz a lei de conservação de energia e calcule período e amplitude.

E 10.17.8. Considere o modelo simplificado de FitzHugh-Nagumo para o potencial elétrico sobre a membrana de um neurônio:

$$\frac{dV}{dt} = V - V^3/3 - W + I \tag{10.403}$$

$$\frac{dW}{dt} = 0{,}08(V + 0{,}7 - 0{,}8W) \tag{10.404}$$

onde I é a corrente de excitação.

- Encontre o único estado estacionário (V_0,W_0) com $I = 0$.
- Resolva numericamente o sistema com condições iniciais dadas por (V_0,W_0) e

  $I = 0$

  $I = 0{,}2$

  $I = 0{,}4$

  $I = 0{,}8$

  $I = e^{-t/200}$

Capítulo 11

Problemas de valores de contorno

Neste capítulo, tratamos dos métodos numéricos para resolver equações diferenciais ordinárias com condições de contorno.

Nos códigos `Python` apresentados, assumimos que as seguintes bibliotecas e módulos estão carregados:

```
>>> from __future__ import division
>>> import numpy as np
>>> from numpy import linalg
>>> import matplotlib.pyplot as plt
```

11.1 Método de diferenças finitas

Nesta seção, discutimos os fundamentos do **método de diferenças finitas** (MDF) para **problemas de valores de contorno** (PVC). Este método consiste na reformulação do problema contínuo em um problema discreto usando fórmulas de diferenças finitas tomadas sobre uma malha apropriada.

Para introduzir os conceitos principais, consideramos o seguinte problema de valor de contorno (PVC)

$$\begin{aligned} -u_{xx} &= f(x,u), \quad a < x < b, && (11.1)\\ u(a) &= u_a, && (11.2)\\ u(b) &= u_b, && (11.3) \end{aligned}$$

onde u_a e u_b são dados. Por ter fixados os valores da variável u nos contornos, este é chamado de PVC com condições de Dirichlet[1].

A resolução de um tal problema pelo método de diferenças finitas consiste em quatro etapas fundamentais: 1. construção da malha, 2. construção do problema

[1] Johann Peter Gustav Lejeune Dirichlet, 1805 - 1859, matemático alemão.

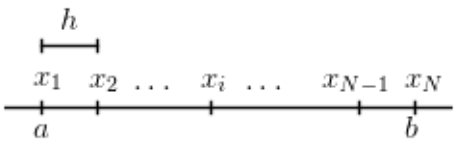


Figura 11.1: Malha uniforme de N pontos em um intervalo $[a, b]$.

discreto, 3. resolução do problema discreto e 4. visualização e interpretação dos resultados.

1. Construção da malha. A malha consiste em uma representação discreta do domínio $[a, b]$. Como veremos, sua construção tem impacto direto nas próximas etapas do método. Aqui, vamos construir a malha mais simples possível, aquela que consiste de N pontos igualmente espaçados, isto é, a chamada **malha uniforme**.

Para tanto, seja $N \in \mathbb{N}$ dado e, então, tomamos o seguinte conjunto discreto $\mathcal{P}_N = \{x_1, x_2, \ldots, x_N\}$ (a malha), onde

$$x_i = a + (i-1)h, \quad i = 1, 2, \ldots, N, \tag{11.4}$$

com

$$h := \frac{b-a}{N-1}, \tag{11.5}$$

o qual é chamado de **tamanho (ou passo) da malha** (veja a Figura 11.1).

2. Construção do problema discreto. A segunda etapa consiste na discretização das equações, no nosso caso, das equações (11.1)-(11.3).

Vamos começar pela Equação (11.1). Em um ponto da malha x_i, $i = 2, 3, \ldots, N-1$, temos

$$-u_{xx}(x_i) = f(x_i, u(x_i)). \tag{11.6}$$

Usando a fórmula de diferenças finitas central de ordem 2 para a segunda derivada, temos

$$-\left(\frac{u(x_i - h) - 2u(x_i) + u(x_i + h)}{h^2} + O(h^2)\right) = f(x_i, u(x_i)). \tag{11.7}$$

Rearranjando os termos, obtemos

$$-\frac{u(x_i - h) - 2u(x_i) + u(x_i + h)}{h^2} = f(x_i, u(x_i)) + O(h^2). \tag{11.8}$$

Agora, denotando por u_i a aproximação numérica de $u(x_i)$, a equação acima nos fornece

$$\frac{1}{h^2}u_{i-1} - \frac{2}{h^2}u_i + \frac{1}{h^2}u_{i+1} = -f(x_i, u_i), \tag{11.9}$$

para $i = 2, 3, \ldots, N-1$. Observamos que trata-se de um sistema de N incógnitas, a saber u_i, e de $N-2$ equações, isto é, um sistema subdeterminado.

Para obtermos um sistema determinado, aplicamos as condições de contorno. Da condição de contorno dada na Equação (11.2), temos

$$u(a) = u_a \Rightarrow u_1 = u_a. \tag{11.10}$$

Analogamente, da condição de contorno dada na Equação (11.2), temos

$$u(b) = u_b \Rightarrow u_N = u_b. \tag{11.11}$$

Por fim, as equações (11.11), (11.9) e (11.10) determinam o problema discreto associado

$$\begin{aligned} u_1 &= u_a, && (11.12)\\ \frac{1}{h^2}u_{i-1} - \frac{2}{h^2}u_i + \frac{1}{h^2}u_{i+1} &= -f(x_i, u_i), \quad i = 2, \ldots, N-1, && (11.13)\\ u_N &= u_b. && (11.14) \end{aligned}$$

Este é um sistema de equações de N incógnitas e N equações.

3. Resolução do sistema discreto. Esta etapa consiste em resolver o sistema discreto construído na etapa anterior.

Para o PVC (11.1)-(11.3), construímos o problema discreto (11.12)-(11.14). Este é um problema de N equações e N incógnitas. Observamos que se $f(x, u)$ é uma função linear, o sistema será linear e podemos resolver o sistema usando de técnicas numéricas para sistema lineares. Agora, se $f(x, u)$ é uma função não linear, podemos usar, por exemplo, do método de Newton para sistemas.

4. Visualização e interpretação dos resultados. A solução do problema discreto consiste dos valores u_i, isto é, de aproximações dos valores de u nos pontos da malha. Para visualizarmos a solução podemos, por exemplo, construir o gráfico do conjunto de pontos $\{(x_i, u_i)\}$. Ainda, para obtermos aproximações da solução em outros pontos que não fazem parte da malha, podemos usar de técnicas de interpolação e/ou ajuste.

Exemplo 11.1.1. Use o método de diferenças finitas para resolver o seguinte problema de valor de contorno com condições de Dirichlet homogêneas:

$$\begin{aligned} -u_{xx} &= 100(x-1)^2, \quad 0 < x < 1, && (11.15)\\ u(0) &= 0, && (11.16)\\ u(1) &= 0. && (11.17) \end{aligned}$$

Use a fórmula de diferenças finitas central de ordem 2 para discretizar a derivada em uma malha uniforme de 11 pontos. Calcule, também, a solução analítica deste

problema, faça um esboço das soluções numérica e analítica e compute o erro absoluto médio definido por

$$E := \frac{1}{N}\sum_{i=1}^{N} |u(x_i) - u_i|\,, \tag{11.18}$$

onde x_i é o i-ésimo ponto da malha, $i = 1, 2, \ldots, N$ e N é o número de pontos na mesma. Por fim, repita seus cálculos para uma malha com 101 pontos. O que ocorre com o erro absoluto médio?

Solução. Vamos seguir as etapas conforme acima.

1. Construção da malha. Tomando $N = 11$, definimos os pontos da malha no domínio [0, 1] por:

$$x_i = (i-1)h, \quad i = 1, 2, \ldots, N, \tag{11.19}$$

com $h = 1/(N-1)$.

Em `Python`, podemos construir a malha da seguinte forma:

```
a = 0
b = 1
N = 11
h = (b-a)/(N-1)
x = np.linspace(a,b,N)
```

2. Construção do problema discreto. Usando a fórmula de diferenças finitas central de ordem 2 para aproximar a derivada na Equação (11.15), obtemos o seguinte sistema de equações:

$$-\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} = 100(x_i - 1)^2, \quad i = 2, \ldots, N-1. \tag{11.20}$$

Completamos este sistema com as condições de contorno dadas nas equações (11.16) e (11.17), donde

$$u_1 = u_N = 0. \tag{11.21}$$

Ou seja, obtemos o seguinte problema discreto:

$$\begin{aligned} u_1 &= 0, && (11.22)\\ -\frac{1}{h^2}\left(u_{i+1} - 2u_i + u_{i+1}\right) &= 100(x_i - 1)^2, \quad i = 2, \ldots, N-1, && (11.23)\\ u_N &= 0. && (11.24) \end{aligned}$$

Observamos que este é um sistema linear $N \times N$, o qual pode ser escrito na forma matricial $A\underline{u} = b$, cujos matriz de coeficientes é

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \cdots & 0 \\ 1 & -2 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & -2 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}, \tag{11.25}$$

o vetor das incógnitas e o vetor dos termos constantes são

$$\underline{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_N \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} 0 \\ -100h^2(x_2-1)^2 \\ -100h^2(x_3-1)^2 \\ \vdots \\ 0 \end{bmatrix}. \tag{11.26}$$

Em `Python`, podemos construir o problema discreto a seguinte forma:

```
A = np.zeros((N,N))
b = np.zeros(N)

A[0,0] = 1
b[0] = 0
for i in np.arange(1,N-1):
    A[i,i-1] = 1
    A[i,i] = -2
    A[i,i+1] = 1
    b[i] = -100 * h**2 * (x[i]-1)**2
A[N-1,N-1] = 1
b[N-1] = 0
```

3. Resolução do problema discreto. Neste caso, o problema discreto consiste no sistema linear $A\underline{u} = b$ e, portanto, a solução é

$$\underline{u} = A^{-1}b. \tag{11.27}$$

Em `Python`, podemos computar a solução do sistema $A\underline{u} = b$ com:

```
u = np.linalg.solve(A,b)
```

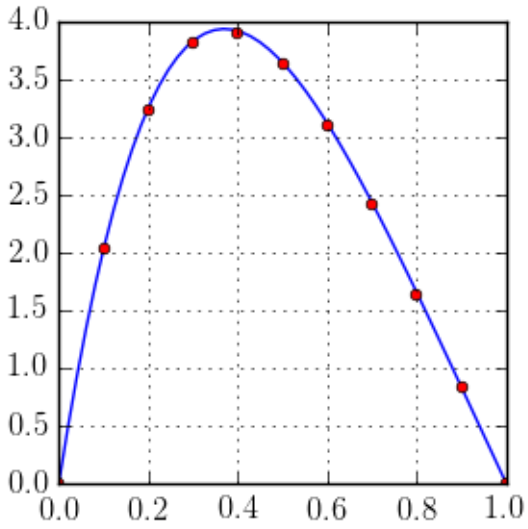


Figura 11.2: Esboço dos gráficos das soluções analítica (linha) e numérica (pontos) do PVC dado no Exemplo 11.1.1.

4. Visualização e interpretação dos resultados. Tendo resolvido o problema discreto $A\underline{u} = b$, obtemos os valores da solução numérico de u nos pontos da malha, isto é, obtivemos o conjunto de pontos $\{(x_i, u_i)\}_{i=1}^{N}$. Neste exemplo, queremos comparar a solução numérica com a solução analítica.

A solução analítica pode ser obtida por integração. Temos:

$$\begin{aligned} -u_{xx} = 100(x-1)^2 \Rightarrow -u_x + c_1 = 100\frac{(x-1)^3}{3} \\ \Rightarrow -u + c_2 x + c_1 = 100\frac{(x-1)^4}{12}, \end{aligned} \tag{11.28}$$

ou seja, $u(x) = -\dfrac{(x-1)^4}{12} + c_2 x + c_1$. As constantes são determinadas pelas condições de contorno dadas pelas equações (11.16) e (11.17), isto é:

$$\begin{aligned} u(0) = 0 \Rightarrow c_1 = \frac{100}{12}, \\ u(1) = 0 \Rightarrow c_2 = -\frac{100}{12}. \end{aligned} \tag{11.29}$$

Portanto, a solução analítica é:

$$u(x) = -100\frac{(x-1)^4}{12} - 100\frac{x}{12} + \frac{100}{12} \tag{11.30}$$

Tabela 11.1: Erro absoluto médio das soluções numéricas com $N = 11$ e $N = 101$ do PVC dado no Exemplo 11.1.1.

N	h	E
11	0,1	$1{,}3 \times 10^{-2}$
101	0,01	$1{,}4 \times 10^{-4}$

A Figura 11.2 mostra o esboço dos gráficos das soluções analítica (11.30) e a da solução numérica (11.27).

Em `Python`, podemos fazer o esboço das soluções analítica e numérica da seguinte forma:

```
#def. sol. analitica
def ue(x):
    return -100.0*(x-1)**4/12 - 100*x/12 + 100.0/12

#grafico
xx = np.linspace(0,1)
yy = np.zeros(50)
for i,xxi in enumerate(xx):
    yy[i] = ue(xxi)

plt.plot(x',u,'ro',xx,yy,'b-')
plt.show()
```

Por fim, computamos o erro absoluto médio das soluções numéricas com $N = 11$ e $N = 101$. A Tabela 11.1 mostra os resultados obtidos. Observamos, que ao diminuirmos 10 vezes o tamanho do passo h, o erro absoluto médio diminui aproximadamente 100 vezes. Este resultado é esperado, pois o problema discreto (11.22)-(11.24) aproxima o problema contínuo (11.15)-(11.17) com erro de truncamento de ordem h^2. Verifique!

Em `Python`, podemos computar o erro absoluto médio da seguinte forma:

```
E = 0
for i,xi in enumerate(x):
    E += np.abs(ue(xi) - u[i])
E /= N
```

$\Diamond$

x	u	x	u
0.50	1.000000	1.00	1.643900
0.60	1.143722	1.10	1.745332
0.70	1.280661	1.20	1.834176
0.80	1.410269	1.30	1.908160
0.90	1.531724	1.40	1.964534
1.00	1.643900	1.50	2.000000

Tabela 11.2: Solução numérica do Exercício 11.1.1.

Exercícios resolvidos

ER 11.1.1. Use o método de diferenças finitas para resolver o seguinte problema de valor de contorno:

$$\begin{aligned} -u_{xx} + u &= e^{-x}, \quad 0 < x < 1, && (11.31)\\ u(0{,}5) &= 1, && (11.32)\\ u(1{,}5) &= 2. && (11.33) \end{aligned}$$

Para tanto, use a fórmula de diferenças finitas central de ordem 2 para discretizar a derivada em uma malha uniforme com passo $h = 0{,}1$. Faça, então, um esboço do gráfico da solução computada.

Solução. O passo h é uma malha uniforme com N pontos no domínio $[0{,}5, 1{,}5]$ satisfaz:

$$h = \frac{(b-a)}{N-1} \Rightarrow N = \frac{(b-a)}{h} + 1. \tag{11.34}$$

Ou seja, a malha deve conter $N = 11$ pontos igualmente espaçados. Denotamos os pontos na malha por x_i, onde $x_i = 0{,}5 + (i-1)h$.

Agora, a equação diferencial dada no i-ésimo ponto da malha é:

$$-u_{xx}(x_i) + u(x_i) = e^{x_i}, \quad i = 2, 3, \ldots, N-1. \tag{11.35}$$

Denotando $u_i \approx u(x_i)$ e usando a fórmula de diferenças finitas central de ordem dois para a derivada u_{xx}, obtemos:

$$-\left(\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2}\right) + u_i = e^{x_i}, \tag{11.36}$$

para $i = 2, 3, \ldots, N-1$. Rearranjando os termos e aplicando as condições de contorno, temos o problema discretizado como segue:

$$\begin{gathered} u_1 = 1 \\ -u_{i-1} + (2 + h^2)u_i - u_{i+1} = h^2 e^{x_i}, \quad i = 2, \ldots, N-1, \\ u_N = 2. \end{gathered} \tag{11.37}$$

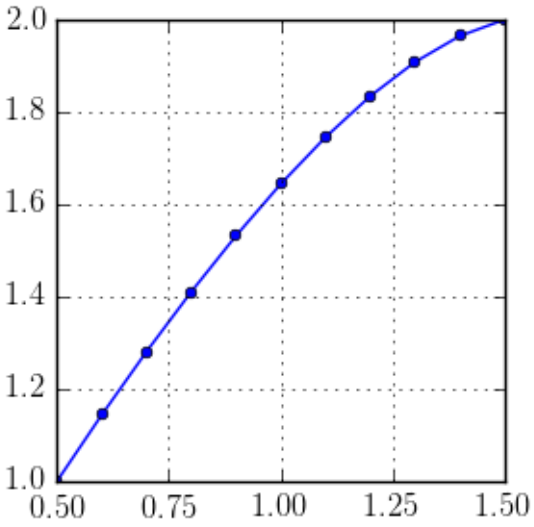


Figura 11.3: Esboço do gráfico da solução numérica do Exercício 11.1.1.

O problema discreto obtido é um sistema linear $N \times N$. Resolvendo este sistema, obtemos a solução discreta apresentada na Tabela 11.2. A Figura 11.3 mostra um esboço do gráfico da solução computada.

Em `Python`, podemos computar a solução numérica e graficá-la com o seguinte código:

```
#malha
a = 0.5
b = 1.5
N = 11
h = (b-a)/(N-1)
x = np.linspace(a,b,N)

#sistema
A = np.zeros((N,N))
b = np.zeros(N)

A[0,0] = 1
b[0] = 1
for i in np.arange(1,N-1):
    A[i,i-1] = -1
    A[i,i] = 2 + h**2
    A[i,i+1] = -1
```

```
    b[i] = h**2 * np.exp(x[i])
A[N-1,N-1] = 1
b[N-1] = 2

#solucao
u = np.linalg.solve(A,b)

#grafico
plt.plot(x,u,'b-o')
plt.show()
```

◊

Exercícios

E 11.1.1. Considere o seguinte problema de valor de contorno para a equação de calor no estado estacionário:

$$\begin{cases} -u_{xx} = 32, \ \ 0 < x < 1. \\ u(0) = 5 \\ u(1) = 10 \end{cases} \tag{11.38}$$

Defina $u_j = u(x_j)$ onde $x_j = (j-1)h$ e $j = 1, \ldots, 5$. Aproxime a derivada segunda por um esquema de segunda ordem e transforme a equação diferencial em um sistema de equações lineares. Escreva este sistema linear na forma matricial e resolva-o. Faça o mesmo com o dobro de subintervalos, isto é, com malha de 9 pontos.

E 11.1.2. Considere o seguinte problema de valor de contorno para a equação de calor no estado estacionário:

$$\begin{cases} -u_{xx} = 200e^{-(x-1)^2}, \ \ 0 < x < 2. \\ u(0) = 120 \\ u(2) = 100 \end{cases} \tag{11.41}$$

Defina $u_j = u(x_j)$ onde $x_j = (j-1)h$ e $j = 1, \ldots, 21$. Aproxime a derivada segunda por um esquema de segunda ordem e transforme a equação diferencial em um sistema de equações lineares. Resolva o sistema linear obtido.

E 11.1.3. Considere o seguinte problema de valor de contorno para a equação

de calor no estado estacionário:

$$\begin{cases} -u_{xx} = 200e^{-(x-1)^2}, \ \ 0 < x < 2. \\ u'(0) = 0 \\ u(2) = 100 \end{cases} \tag{11.42}$$

Defina $u_j = u(x_j)$ onde $x_j = (j-1)h$ e $j = 1, \ldots, 21$. Aproxime a derivada segunda por um esquema de segunda ordem, a derivada primeira na fronteira por um esquema de primeira ordem e transforme a equação diferencial em um sistema de equações lineares. Resolva o sistema linear obtido.

E 11.1.4. Considere o seguinte problema de valor de contorno para a equação de calor no estado estacionário com um termo não linear de radiação:

$$\begin{cases} -u_{xx} = 100 - \frac{u^4}{10000}, \ \ 0 < x < 2. \\ u(0) = 0 \\ u(2) = 10 \end{cases} \tag{11.43}$$

Defina $u_j = u(x_j)$ onde $x_j = (j-1)h$ e $j = 1, \ldots, 21$. Aproxime a derivada segunda por um esquema de segunda ordem e transforme a equação diferencial em um sistema de equações não lineares. Resolva o sistema obtido. Expresse a solução com dois algarismos depois do separador decimal. Dica: Veja problema 38 da lista 2, seção de sistemas não lineares.

E 11.1.5. Considere o seguinte problema de valor de contorno para a equação de calor no estado estacionário com um termo não linear de radiação e um termo de convecção:

$$\begin{cases} -u_{xx} + 3u_x = 100 - \frac{u^4}{10000}, \ \ 0 < x < 2. \\ u'(0) = 0 \\ u(2) = 10 \end{cases} \tag{11.44}$$

Defina $u_j = u(x_j)$ onde $x_j = (j-1)h$ e $j = 1, \ldots, 21$. Aproxime a derivada segunda por um esquema de segunda ordem, a derivada primeira na fronteira por um esquema de primeira ordem, a derivada primeira no interior por um esquema de segunda ordem e transforme a equação diferencial em um sistema de equações não lineares. Resolva o sistema obtido.

E 11.1.6. Considere o seguinte problema de valor de contorno:

$$\begin{cases} -u'' + 2u' = e^{-x} - \frac{u^2}{100}, \ \ 1 < x < 4. \\ u'(1) + u(1) = 2 \\ u'(4) = -1 \end{cases} \tag{11.45}$$

Defina $u_j = u(x_j)$ onde $x_j = 1 + (j-1)h$ e $j = 1, \ldots, 101$. Aproxime a derivada segunda por um esquema de segunda ordem, a derivada primeira na fronteira por um esquema de primeira ordem, a derivada primeira no interior por um esquema de segunda ordem e transforme a equação diferencial em um sistema de equações não lineares. Resolva o sistema obtido.

Apêndice A

Rápida introdução ao Python

Neste apêndice, discutiremos os principais aspectos da linguagem computacional `Python` que são essenciais para uma boa leitura desta versão do livro. O material aqui apresentado, é uma adaptação livre do Apêndice A de [12].

A.1 Sobre a linguagem Python

Python é uma linguagem de programação de alto nível, interpretada e multiparadigma. Lançada por Guido van Rossum[1] em 1991 é, atualmente, mantida de forma colaborativa e aberta.

Para mais informações, consulte:

- Página oficial da linguagem Python: `https://www.python.org/`
- Comunidade Python Brasil: `http://wiki.python.org.br/`

Para iniciantes, recomendamos o curso EAD gratuíto no site Codecademy:

`https://www.codecademy.com/learn/python`

A.1.1 Instalação e execução

Para executar um código `Python` é necessário ter instalado um interpretador para a linguagem. No site oficial do Python estão disponíveis para *download* os interpretadores `Python 2.7` e `Python 3` para vários sistemas operacionais, como `Linux`, `Mac OS` e `Windows`. Muitas distribuições de `Linux` (`Linux Mint`, `Ubuntu`, etc.) têm o `Python` no seu sistema de pacotes (incluindo documentação em várias línguas).

Ao longo do texto, assumiremos que o leitor estaja usando um computar rodando `Linux`. Para outros sistemas, pode ser necessário fazer algumas adaptações.

[1] Guido van Rossum, nascido em 1956, programador de computadores dos Países Baixos.

A.1.2 Usando Python

O uso do Python pode ser feito de três formas básicas:

- usando um **console Python** de modo iterativo;
- executando um código `codigo.py` no console `Python`;
- executando um código `Python codigo.py` diretamente em terminal;

Exemplo A.1.1. Considere o seguinte pseudocódigo:

```
s = "Olá, mundo!". (Sem imprimir na tela o resultado.)
saída(s). (Imprime na tela.)
```

Implemente este pseudocódigo em `Python`: a) usando diretamente um console; b) digitando seu código em um arquivo separado e executando-o no console Python com a função `execfile`.b) digitando seu código em um arquivo separado e executando-o em terminal com o comando `python`.

Solução. Seguem as soluções de cada item:

a) No console temos:

```
>>> s = "Olá, mundo!"
>>> print(s)
Olá, mundo!
```

Para sair do console, digite:

```
>>> quit()
```

b) Abra o editor de texto de sua preferência e digite o código:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

s = 'Olá'
print(s)
```

Salve o arquivo como, por exemplo, `ola.py`. No console `Python`, digite:

```
>>> execfile("ola.py")
```

c) Abra o editor de texto de sua preferência e digite o código:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

s = 'Olá'
print(s)
```

Salve o arquivo como, por exemplo, `ola.py`. Em um terminal, digite:

```
$ python ola.py
```

◊

A.2 Elementos da linguagem

`Python` é uma linguagem de alto nível, interpretada e dinâmica. Uma variável é criada quando um valor é atribuído a ela. Por exemplo:

```
>>> x=1
>>> y = x * 2.0
```

a variável `x` recebe o valor `int` 1 e, logo após, na segunda linha de comando, a variável `y` recebe o valor `double` 2. Observamos que o símbolo `=` significa o operador de atribuição não o de igualdade. O operador lógico de igualdade no `Python` é `==`. Veja os seguintes comandos:

```
>>> print(x,y)
(1, 2.0)
>>> type(x), type(y)
(<type 'int'>, <type 'float'>)
```

Comentários e continuação de linha de comando são usados como no seguinte exemplo:

```
>>> #isto é um comentário
...
>>> x = 1 \
... + 2
>>> print(x)
3
```

A.2.1 Operações matemáticas elementares

Em `Python`, os operadores matemáticos elementares são os seguintes:

```
+ adição
- subtração
* multiplicação
/ divisão
** potenciação
```

Atenção, a operação de divisão se comporta diferente nas versões `Python 2.7` e `Python 3`. Em `Python 3`, temos:

```
>>> 1/2
0.5
```

Já, em `Python 2.7`:

```
>>> 1/2
0
>>> from __future__ import division
>>> 1/2
0.5
```

A.2.2 Funções e constantes elementares

Várias funções e constantes elementares estão disponíveis no pacote módulo Python math. Por exemplo:

```
>>> import math as math
>>> math.cos(math.pi)
-1.0
>>> math.exp(1)
2.718281828459045
>>> math.log(math.exp(1))
1.0
```

Observamos que `math.log` é a função logaritmo natural, isto é, $f(x) = \ln(x)$, enquanto que a implementação Python de $f(x) = \log(x)$ é:

```
>>> math.log10(10)
1.0
```

Veja mais na documentação do módulo math:

```
>>> help(math)
```

A.2.3 Operadores lógicos

Em `Python`, o valor lógico verdadeiro é escrito como `True` e o valor lógico falso como `False`. Temos os seguintes operadores lógicos disponíveis:

```
and  e lógico
or  ou lógico
not  negação
== igualdade
!= diferente
<  menor que
>  maior que
<= menor ou igual que
>= maior ou igual que
```

Exemplo A.2.1. Se $x = 2$, então x é maior ou igual a 1 e menor que 3?

Solução. Em `Python`, temos:

```
>>> x=2
>>> (x >= 1) and (x < 3)
True
```

$\Diamond$

A.3 Matrizes

Em `Python`, temos um ótimo suporte para computação científica com o pacote numpy. Uma matriz $A = [a_{i,j}]_{i,j=1}^{m,n}$ em Python é definida usando-se a seguinte sintaxe:

```
>>> import numpy as np
>>> A = np.array([[ a11 , a12 , ... , a1n], [...]. [am1 , am2 , ... , amn]])
```

Exemplo A.3.1. Defina a matriz:

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

Solução. Em `Python`, digitamos:

```
>>> import numpy as np
>>> A = np.array([[1,2,3],
...               [4,5,6]])
```

```
>>> print(A)
[[1 2 3]
 [4 5 6]]
```

◊

A seguinte lista contém uma série de funções que geram matrizes particulares:

```
numpy.eye      matriz identidade
numpy.linspace vetor de elementos linearmente espaçados
numpy.ones     matriz cheia de uns
numpy.zeros    matriz nula
```

A.3.1 Obtendo dados de uma matriz

A função `numpy.shape` retorna o tamanho de uma matriz, por exemplo:

```
>>> A = np.ones((3,2))
>>> print(A)
[[ 1.  1.]
 [ 1.  1.]
 [ 1.  1.]]
>>> nl, nc = np.shape(A)
>>> print(nl,nc)
(3, 2)
```

informando que a matriz `A` tem três linhas e duas colunas.

Existem vários métodos para acessar os elementos de uma matriz dada `A`:

- a matriz inteira acessa-se com a sintaxe:

  `A`

- o elemento da i-ésima linha e j-ésima coluna acessa-se usando a sintaxe:

  `A[i,j]`

- o bloco formado pelas linhas i_1, i_2 e pelas colunas j_1, j_2 obtém-se usando a sintaxe:

  `A[i1:i2, j1:j2]`

Exemplo A.3.2. Veja as seguintes linhas de comando:

```
>>> from numpy import random
>>> A = np.random.random((3,4))
>>> A
array([[ 0.39235668,  0.30287204,  0.24379253,  0.98866709],
       [ 0.72049734,  0.99300252,  0.14232844,  0.25604346],
       [ 0.61553036,  0.80615392,  0.22418474,  0.13685148]])
>>> A[2,3]
0.13685147547025989
>>> A[1:3,1:4]
array([[ 0.99300252,  0.14232844,  0.25604346],
       [ 0.80615392,  0.22418474,  0.13685148]])
```

Definida uma matriz A em Python, as seguintes sintaxes são bastante úteis:

```
A[:,:]   toda a matriz
A[i:j,k] os elementos das linhas i até j (exclusive) da k-ésima coluna
A[i,j:k] os elementos da i-ésima linha das colunas j até k (exclusive)
A[i,:]   a i-ésima linha da matriz
A[:,j]   a j-ésima coluna da matriz
```

Atenção, os índices em `Python` iniciam-se em 0. Assim, o comando `A[1:3,1:4]` retorna o bloco da matriz A compreendido da segunda à terceira linha e da segunda a quarta coluna desta matriz.

Exemplo A.3.3. Veja as seguintes linhas de comando:

```
>>> B = np.random.random((4,4))
>>> B
array([[ 0.94313432,  0.72650883,  0.55487089,  0.18753526],
       [ 0.02094937,  0.45726099,  0.51925464,  0.8535878 ],
       [ 0.75948469,  0.95362926,  0.77942318,  0.06464183],
       [ 0.91243198,  0.22775889,  0.04061536,  0.14908227]])
>>> aux = np.copy(B[:,2])
>>> B[:,2] = np.copy(B[:,3])
>>> B[:,3] = np.copy(aux)
>>> B
array([[ 0.94313432,  0.72650883,  0.18753526,  0.55487089],
       [ 0.02094937,  0.45726099,  0.8535878 ,  0.51925464],
       [ 0.75948469,  0.95362926,  0.06464183,  0.77942318],
       [ 0.91243198,  0.22775889,  0.14908227,  0.04061536]])
```

A.3.2 Operações matriciais e elemento-a-elemento

Em `Python` com `numpy`, o operador `*` opera elemento a elemento. Por exemplo:

```
>>> A = np.array([[1,2],[2,1]]); print(A)
[[1 2]
 [2 1]]
>>> B = np.array([[2,1],[2,1]]); print(B)
[[2 1]
 [2 1]]
>>> print(A*B)
[[2 2]
 [4 1]]
```

A multiplicação matricial obtemos com:

```
>>> C = A.dot(B)
>>> print(C)
[[6 3]
 [6 3]]
```

Aqui, temos as sintaxes análogas entre operações elemento-a-elemento:

```
+ adição
- subtração
* multiplicação
/ divisão
** potenciação
```

Exemplo A.3.4. Veja as seguintes linhas de comando:

```
>>> A = np.ones((2,2))
>>> A
array([[ 1.,  1.],
       [ 1.,  1.]])
>>> B = 2 * np.ones((2,2))
>>> B
array([[ 2.,  2.],
       [ 2.,  2.]])
>>> A*B
array([[ 2.,  2.],
       [ 2.,  2.]])
>>> A.dot(B)
array([[ 4.,  4.],
```

```
       [ 4.,  4.]])
>>> A/B
array([[ 0.5,  0.5],
       [ 0.5,  0.5]])
```

A.4 Estruturas de ramificação e repetição

A linguagem `Python` contém estruturas de repetição e ramificação padrões de linguagens estruturadas.

A.4.1 A instrução de ramificação "if"

A instrução "if" permite executar um pedaço do código somente se uma dada condição for satisfeita.

Exemplo A.4.1. Veja o seguinte código Python:

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

i = 2
if (i == 1):
    print("Olá!")
elif (i == 2):
    print("Hallo!")
elif (i == 3):
    print("Hello!")
else:
    print("Ça Va!")
```

Qual é a saída apresentada pelo código? Por quê?

Observamos que, em `Python`, a identação é obrigatória, pois é ela que defini o escopo da instrução.

A.4.2 A instrução de repetição "for"

A instrução `for` permite que um pedaço de código seja executado repetidamente.

Exemplo A.4.2. Veja o seguinte código:

```
for i in range(6):
    print(i)
```

Qual é a saída deste código? Por quê?

Exemplo A.4.3. Veja o seguinte código:

```
import numpy as np
for i in np.arange(1,8,2):
    print(i)
```

Qual é a saída deste código? Por quê?

Exemplo A.4.4. Veja o seguinte código:

```
for k = 10:-3:1
    disp(k)
end
```

O que é mostrado no console do Python?

Exemplo A.4.5. Veja o seguinte código:

```
import numpy as np
for i in np.arange(10,1,-3):
    print(i)
```

O que é mostrado no console do Python?

A.4.3 A instrução de repetição "while"

A instrução `while` permite que um pedaço de código seja executado repetidamente até que uma dada condição seja satisfeita.

Exemplo A.4.6. Veja o seguinte código Python:

```
s = 0
i = 1
while (i <= 10):
    s = s + i
    i = i + 1
```

Qual é o valor de `s` ao final da execução? Por quê?

A.5 Funções

Além das muitas funções disponíveis em `Python` (e os tantos muitos pacotes livres disponíveis), podemos definir nossas próprias funções. Para tanto, existe a instrução `def`. Veja os seguintes exemplos:

Exemplo A.5.1. O seguinte código:

```
def f(x):
    return x + np.sin(x)
```

define a função $f(x) = x + \text{sen } x$.

Observe que $f(\pi) = \pi$. Confirme isso computando:

```
>>> f(np.pi)
```

Exemplo A.5.2. O seguinte código em `Python`:

```
def h(x,y):
    if (x < y):
        return y - x
    else:
        return x - y
```

define a função:

$$h(x,y) = \begin{cases} y - x & , x < y \\ x - y & , x \geq y \end{cases} \tag{A.2}$$

Exemplo A.5.3. O seguinte código:

```
def J(x):
    y = np.zeros((2,2))
    y[0,0] = 2*x[0]
    y[0,1] = 2*x[1]

    y[1,0] = -x[1]*np.sin(x[0]*x[1])
    y[1,1] = -x[0]*np.sin(x[0]*x[1])

    return y
```

define a matriz jacobiana $J(x_1,x_2) := \frac{\partial(f_1,f_2)}{\partial(x_1,x_2)}$ da função:

$$\boldsymbol{f}(x_1,x_2) = (x_1^2 + x_2^2,\ \cos(x_1 x_2)). \tag{A.3}$$

A.6 Gráficos

Para criar um esboço do gráfico de uma função de uma variável real $y = f(x)$, podemos usar a biblioteca `Python` mathplotlib.A função `matplotlib.pyplot.plot` faz uma representação gráfica de um conjunto de pontos $\{(x_i, y_i)\}$ fornecidos. Existe uma série de opções para esta função de forma que o usuário pode ajustar várias questões de visualização. Veja a documentação.

Exemplo A.6.1. Veja as seguintes linhas de código:

```
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> def f(x): return x**3 + 1
...
>>> x = np.linspace(-2,2)
>>> plt.plot(x, f(x))
[<matplotlib.lines.Line2D object at 0x7f4f6d153510>]
>>> plt.grid()
>>> plt.show()
```

Resposta dos Exercícios

Recomendamos ao leitor o uso criterioso das respostas aqui apresentadas. Devido a ainda muito constante atualização do livro, as respostas podem conter imprecisões e erros.

E 2.1.1. a) 4; b) 9; c) b^2; d) 7; e) 170; f) 7,125; g) 3,28

E 2.1.2. a) 21,172; b) 5,5; c) 303,25; d) $4,\bar{6}$.

E 2.1.3. $(101,1)_2$.

E 2.1.4. $(11,1C)_{16}$.

E 2.1.5. a) $(12,\bar{3}\bar{1})_5$; b) $(45,1)_6$.

E 2.1.6. 10,5; $(1010,1)_2$.

E 2.1.7. a) $(100101,001)_2$; b) $(11,4)_{16}$; c) $(11,5)_8$; d) $(9,A)_{16}$.

E 2.1.8. 50; 18.

E 2.2.1.

$$\begin{array}{ll} a)\ 2{,}99792458 \times 10^5 & b)\ 6{,}62607 \times 10^{-34} \\ c)\ 6{,}674 \times 10^{-8} & d)\ 9{,}80665 \times 10^4 \end{array} \tag{2.32}$$

E 2.2.2. Em `Python`, temos:

```
>>> print("%1.7e" % 299792.458)
2.9979458e+04
>>> print("%1.5e" % 66.2607)
6.62607e+01
>>> print("%1.3e" % 0.6674)
6.674e-01
>>> print("%1.5e" % 9806.65e1)
9.80665e+04
```

E 2.3.1. (a) 1,1; (b) 7,3; (c) $-5,9$.

E 2.3.2. (a) 1,2; (b) 1,2; (c) 2,4; (d) $-2,4$.

E 2.3.3.

Este exercício está sem resposta sugerida. Proponha uma resposta. Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

E 2.4.1. a) $2^6 + 2^5 + 2^1 = 98$; b) $2^4 + 2^3 + 2^2 + 2^0 = 29$; c) -2^7; d) $-2^7 + 2^6 + 2^5 + 2^1 + 2^0 = -29$; e)$-2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 2^0 = -1$. Observe que o dígito mais significativo (mais à esquera) tem peso negativo.

E 2.4.2. a) 25186; b) 7453; c) -7453; d) -1.

E 2.4.3. a) 3,75; b) $-5{,}75$.

E 2.4.5. Devido à precisão finita do sistema de numeração, o laço para quando x for suficientemente grande em comparação a 1 a ponto de x+1 ser aproximado para 1. Isso acontece quando 1 é da ordem do épsilon de máquina em relação a x, isto é, quando $x \approx 2/\%eps$. O tempo de execução fica em torno de 28 anos.

E 2.5.1. a) $\varepsilon_{abs} = 5{,}9 \times 10^{-4}$, $\varepsilon_{rel} = 1{,}9 \times 10^{-2}\%$; b) $\varepsilon_{abs} = \times 10^{-5}$, $\varepsilon_{rel} = \times 10^{-3}\%$; c) $\varepsilon_{abs} = 1$, $\varepsilon_{rel} = 10^{-5}\%$.

E 2.5.2. a) 1,7889; b) 1788,9; c) 0,0017889; d) 0,0045966; e) $2{,}1755 \times 10^{-10}$; f) $2{,}1755 \times 10^{10}$.

E 2.5.3. a) 3270, 3280; b) 42,5, 42,6; c) 0,0000333, 0,0000333.

E 2.5.4. a) 2; b) 2.

E 2.5.5.

$$0{,}1x - 0{,}01 = 12 \tag{2.83}$$
$$0{,}1x = 12 + 0{,}01 = 12{,}01 \tag{2.84}$$
$$x = 120{,}1 \tag{2.85}$$

A resposta exata é 120,1.

E 2.5.6. a) $\delta_{\text{abs}} = 3{,}46 \times 10^{-7}$, $\delta_{\text{rel}} = 1{,}10 \times 10^{-7}$; b) $\delta_{\text{abs}} = 1{,}43 \times 10^{-4}$, $\delta_{\text{rel}} = 1{,}00 \times 10^{-3}$.

E 2.8.1. 2%, deve-se melhorar a medida na variável x, pois, por mais que o erro relativo seja maior para esta variável, a propagação de erros através desta variáveis é muito menos importante do que para a outra variável.

E 2.8.2. 3,2% pela aproximação ou 3,4% pelo segundo método, isto é, $(0{,}96758 \leq I \leq 1{,}0342)$.

E 2.9.1. Quando μ é pequeno, $e^{1/\mu}$ é um número grande. A primeira expressão produz um "overflow" (número maior que o máximo representável) quando μ é pequeno. A segunda expressão, no entanto, reproduz o limite 1 quando $\mu \to 0+$.

E 2.9.2. a) $\frac{1}{2} + \frac{x^2}{4!} + O(x^4)$; b) $x/2 + O(x^2)$; c) $5 \cdot 10^{-4}x + O(x^2)$; d) $\frac{\sqrt{2}}{4}y + O(y^2) = \frac{\sqrt{2}}{4}x + O(x^2)$

E 2.9.3. A expressão da direita se comporta melhor devido à retirada do cancelamento catastrófico em x em torno de 0.

E 2.9.4. Possíveis soluções são:

$$\sqrt{e^{2x}+1} - e^x = \sqrt{e^{2x}+1} - e^x \cdot \frac{\sqrt{e^{2x}+1} + e^x}{\sqrt{e^{2x}+1} + e^x} \tag{2.202}$$
$$= \frac{e^{2x} + 1 - e^{2x}}{\sqrt{e^{2x}+1} + e^x} = \frac{1}{\sqrt{e^{2x}+1} + e^x} \tag{2.203}$$

e, de forma análoga:

$$\sqrt{e^{2x}+x^2}-e^x=\frac{x^2}{\sqrt{e^{2x}+x^2}+e^x}. \tag{2.204}$$

E 2.9.5. $4{,}12451228\times10^{-16}$ J; 0,002%; $0{,}26654956\times10^{-14}$ J; 0,002%; $4{,}98497440\times10^{-13}$ J; 0,057%; $1{,}74927914\times10^{-12}$ J; 0,522%.

E 3.1.1.

Observamos que a equação é equivalente a $\cos(x)-x=0$. Tomando, então, $f(x)=\cos(x)-x$, temos que $f(x)$ é contínua em $[0,\pi/2]$, $f(0)=1$ e $f(\pi/2)=-\pi/2<0$. Logo, do teorema de Bolzano 3.1.1, concluímos que a equação dada tem pelo menos uma solução no intervalo $(0,\pi/2)$.

E 3.1.2.

No Exercício 3.1.1, mostramos que a função $f(x)=\cos(x)-x$ tem um zero no intervalo $[0,\pi/2]$. Agora, observamos que $f'(x)=-\operatorname{sen}(x)-1$. Como $0<\operatorname{sen}x<1$ para todo $x\in(0,\pi/2)$, temos que $f'(x)<0$ em $(0,\pi/2)$, isto é, $f(x)$ é monotonicamente decrescente neste intervalo. Logo, da Proposição 3.1.1, temos que existe um único zero da função neste intervalo.

E 3.1.3.

$k\approx0{,}161228$

E 3.1.5.

Escolhendo o intervalo $[a,b]=[-1{,}841-10^{-3},-1{,}841+10^{-3}]$, temos $f(a)\approx5\times10^{-4}>0$ e $f(b)\approx-1{,}2\times10^{-3}<0$, isto é, $f(a)\cdot f(b)<0$. Então, o teorema de Bolzano nos garante que o zero exato x^* de $f(x)$ está no intervalo (a,b). Logo, da escolha feita, $|-1{,}841-x^*|<10^{-3}$.

E 3.1.6. Basta aplicar as ideias da solução do Exercício 3.1.5.

E 3.2.1. 0,6875

E 3.2.2. Intervalo $(0{,}4,0{,}5)$, zero 0,45931. Intervalo $(1{,}7,1{,}8)$, zero 1,7036. Intervalo $(2{,}5,2{,}6)$, zero 2,5582.

E 3.2.3. a) $x_1=1$. b) Dica: como $x_2=2$ é raiz dupla, tem-se que $p'(x_2)=0$.

E 3.2.5. 1,390054; 1,8913954; 2,4895673; 3,1641544; 3,8965468

E 3.2.6. $k\theta=\frac{lP}{2}\cos(\theta)$ com $\theta\in(0,\pi/2)$; 1,030.

E 3.2.7. 19; 23; 26; 0,567143; 1,745528; 3,385630

E 3.2.8. a) 0,623; b) 0,559; c) 0,500; d) 0,300; e) $-0{,}3$; f) -30; g) -30

E 3.2.9. a) 0,0294; b) $2.44e-3$; c) $2.50e-4$; d) $1.09\cdot10^{-7}$; e) -10^{-12}; f) -10^{-12}; g) -10^{-12}

E 3.3.1. $-1{,}8414057$

E 3.3.2.

0,7391

E 3.3.3.
Tomemos $x^{(1)} = 1$ como aproximação inicial para a solução deste problema, iterando a primeira sequência a), obtemos:

$$x^{(1)} = 1 \tag{3.79}$$
$$x^{(2)} = \ln\left(\frac{10}{1}\right) = 2{,}3025851 \tag{3.80}$$
$$x^{(3)} = \ln\left(\frac{10}{2{,}3025851}\right) = 1{,}4685526 \tag{3.81}$$
$$\vdots \tag{3.82}$$
$$x^{(21)} = 1{,}7455151 \tag{3.83}$$
$$x^{(31)} = 1{,}745528 \tag{3.84}$$
$$x^{(32)} = 1{,}745528 \tag{3.85}$$

Iterando a segunda sequência b), obtemos:

$$x^{(1)} = 1 \tag{3.86}$$
$$x^{(2)} = 10e^{-1} = 3{,}6787944 \tag{3.87}$$
$$x^{(3)} = 10e^{-3{,}6787944} = 0{,}2525340 \tag{3.88}$$
$$x^{(4)} = 10e^{-0{,}2525340} = 7{,}7682979 \tag{3.89}$$
$$x^{(5)} = 10e^{-7{,}7682979} = 0{,}0042293 \tag{3.90}$$
$$x^{(6)} = 10e^{-0{,}0042293} = 9{,}9577961 \tag{3.91}$$

Este experimento numérico sugere que a iteração a) converge para 1,745528 e a iteração b) não é convergente.

E 3.3.7. $x_1 \approx 1{,}4506619$, $x_2 \approx 4{,}8574864$, $x_3 = 7{,}7430681$.

E 3.3.10.
0.0431266

E 3.4.1. raiz:0,82413, processo iterativo: $x^{(n+1)} = x^{(n)} + \frac{\cos(x)-x^2}{\text{sen }(x)+2x}$

E 3.4.3. 0,65291864

E 3.4.4. 0,0198679; 0,533890; 0,735412; 1,13237 e 1,38851.

E 3.4.6. -99.99970, -0.3376513; -1.314006.

E 3.4.9.
$x_0 > 1$.

E 3.4.10.

$$x^{(0)} = \text{C.I.} \tag{3.147}$$
$$x^{(n+1)} = x^{(n)}\left(2 - Ax^{(n)}\right) \tag{3.148}$$
$$\tag{3.149}$$

E 3.4.11.

$$x_0 = \text{C.I.} \tag{3.150}$$
$$x^{(n+1)} = x^{(n)}\left(1 - \frac{1}{n}\right) + \frac{A}{nx^{(n)}} \tag{3.151}$$

E 3.4.12.

$$x_0 = \text{C.I.} \tag{3.152}$$

$$x^{(n+1)} = x^{(n)} + \frac{x^{(n)} - Ax^{(n)}}{2} = \frac{(3-A)x^{(n)}}{2} \tag{3.153}$$

$$\tag{3.154}$$

E 3.6.5. Seja $f(x) \in C^2$ um função tal que $f(x^*) = 0$ e $f'(x^*) \neq 0$. Considere o processo iterativo do método das secantes:

$$x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)})}{f(x^{(n)}) - f(x^{(n-1)})}(x^{(n)} - x^{(n-1)}) \tag{3.201}$$

Esta expressão pode ser escrita como:

$$x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)})(x^{(n)} - x^{(n-1)})}{f(x^{(n)}) - f(x^{(n-1)})} \tag{3.202}$$

$$\tag{3.203}$$

$$= \frac{x^{(n)}\left(f(x^{(n)}) - f(x^{(n-1)})\right) - f(x^{(n)})(x^{(n)} - x^{(n-1)})}{f(x^{(n)}) - f(x^{(n-1)})} \tag{3.204}$$

$$= \frac{x^{(n)}f(x^{(n-1)}) - x^{(n-1)}f(x^{(n)})}{f(x^{(n)}) - f(x^{(n-1)})} \tag{3.205}$$

Subtraindo x^* de ambos os lados temos:

$$x^{(n+1)} - x^* = \frac{x^{(n)}f(x^{(n-1)}) - x^{(n-1)}f(x^{(n)})}{f(x^{(n)}) - f(x^{(n-1)})} - x^* \tag{3.206}$$

$$= \frac{x^{(n)}f(x^{(n-1)}) - x^{(n-1)}f(x^{(n)}) - x^*\left(f(x^{(n)}) - f(x^{(n-1)})\right)}{f(x^{(n)}) - f(x^{(n-1)})} \tag{3.207}$$

$$= \frac{(x^{(n)} - x^*)f(x^{(n-1)}) - (x^{(n-1)} - x^*)f(x^{(n)})}{f(x^{(n)}) - f(x^{(n-1)})} \tag{3.208}$$

Definimos $\epsilon_n = x_n - x^*$, equivalente a $x_n = x^* + \epsilon_n$

$$\epsilon_{n+1} = \frac{\epsilon_n f(x^* + \epsilon_{n-1}) - \epsilon_{n-1}f(x^* + \epsilon_n)}{f(x^* + \epsilon_n) - f(x^* + \epsilon_{n-1})} \tag{3.209}$$

Aproximamos a função $f(x)$ no numerador por

$$f(x^* + \epsilon) \approx f(x^*) + \epsilon f'(x^*) + \epsilon^2 \frac{f''(x^*)}{2} \tag{3.210}$$

$$f(x^* + \epsilon) \approx \epsilon f'(x^*) + \epsilon^2 \frac{f''(x^*)}{2} \tag{3.211}$$

$$\epsilon_{n+1} \approx \frac{\epsilon_n\left[\epsilon_{n-1}f'(x^*) + \epsilon_{n-1}^2\frac{f''(x^*)}{2}\right] - \epsilon_{n-1}\left[\epsilon_n f'(x^*) + \epsilon_n^2\frac{f''(x^*)}{2}\right]}{f(x^* + \epsilon_n) - f(x^* + \epsilon_{n-1})} \tag{3.212}$$

$$= \frac{\frac{f''(x^*)}{2}\left(\epsilon_n\epsilon_{n-1}^2 - \epsilon_{n-1}\epsilon_n^2\right)}{f(x^* + \epsilon_n) - f(x^* + \epsilon_{n-1})} \tag{3.213}$$

$$= \frac{1}{2}f''(x^*)\frac{\epsilon_n\epsilon_{n-1}\left(\epsilon_{n-1} - \epsilon_n\right)}{f(x^* + \epsilon_n) - f(x^* + \epsilon_{n-1})} \tag{3.214}$$

Observamos, agora, que

$$\begin{aligned} f(x^* + \epsilon_n) - f(x^* + \epsilon_{n-1}) &\approx \left[f(x^*) + f'(x^*)\epsilon_n\right] - \left[f(x^*) + f'(x^*)\epsilon_{n-1}\right] \\ &= f'(x^*)(\epsilon_n - \epsilon_{n-1}) \end{aligned} \tag{3.215}$$

Portanto:

$$\epsilon_{n+1} \approx \frac{1}{2}\frac{f''(x^*)}{f'(x^*)}\epsilon_n\epsilon_{n-1} \tag{3.216}$$

ou, equivalentemente:

$$x^{(n+1)} - x^* \approx \frac{1}{2}\frac{f''(x^*)}{f'(x^*)}\left(x^{(n)} - x^*\right)\left(x^{(n-1)} - x^*\right) \tag{3.217}$$

E 3.7.2.

$x > a$ com $a \approx 0{,}4193648$.

E 3.7.3.

$z_1 \approx 0.3252768$, $z_2 \approx 1.5153738$, $z_3 \approx 2.497846$, $z_4 \approx 3.5002901$, $z_j \approx j - 1/2 - (-1)^j \frac{e^{-2j+1}}{\pi}$, $\quad j > 4$

E 3.7.4.

150 W, 133 W, 87 W, 55 W, 6,5 W

E 3.7.5.

a) 42 s e 8 min2 s, b) 14 min56 s.

E 3.7.6.

118940992

E 3.7.7.

7,7 cm

E 3.7.8.

4,32 cm

E 3.7.9.

$(0{,}652919, 0{,}426303)$

E 3.7.10.

7,19% ao mês

E 3.7.11.

4,54% ao mês.

E 3.7.12.

500 K, 700 K em $t = 3\ln(2)$, 26 min, 4 h27 min.

E 3.7.13.

$(\pm 1{,}1101388, -0{,}7675919)$, $(\pm 1{,}5602111, 0{,}342585)$

E 3.7.14.

1,5318075

E 3.7.15.

Aproximadamente 2500 reais por hora.

E 3.7.16.

a) 332,74 K b) 359,33 K

E 3.7.17.

1,2285751, 4,76770758, 7,88704085

E 4.1.1. Escrevemos o sistema na forma matricial e resolvemos:

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 0 \\ 1 & 0 & 10 & -48 \\ 0 & 10 & 1 & 25 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 1 & 1 & 0 \\ 0 & -1 & 9 & -48 \\ 0 & 10 & 1 & 25 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 1 & 1 & 0 \\ 0 & 10 & 1 & 25 \\ 0 & -1 & 9 & -48 \end{array}\right] \sim \tag{4.34}$$

$$\sim \left[\begin{array}{ccc|c} 1 & 1 & 1 & 0 \\ 0 & 10 & 1 & 25 \\ 0 & 0 & 9.1 & -45.5 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 1 & 1 & 0 \\ 0 & 10 & 1 & 25 \\ 0 & 0 & 1 & -5 \end{array}\right] \sim \tag{4.35}$$

$$\sim \left[\begin{array}{ccc|c} 1 & 1 & 0 & 5 \\ 0 & 10 & 0 & 30 \\ 0 & 0 & 1 & -5 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 1 & 0 & 5 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -5 \end{array}\right] \sim \tag{4.36}$$

$$\sim \left[\begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -5 \end{array}\right] \tag{4.37}$$

Portanto $x = 2$, $y = 3$, $z = -5$

E 4.5.1.

$$\begin{aligned} a &= (0, 1, 2, 1, 2, 1) && (4.123)\\ b &= (5, 3, 4, 2, 3, 2) && (4.124)\\ c &= (4, 1, 1, 1, 2, 0) && (4.125)\\ d &= (13, 10, 20, 16, 35, 17) && (4.126)\\ x &= (1, 2, 3, 4, 5, 6) && (4.127) \end{aligned}$$

E 4.5.2.

$$\begin{aligned} a &= (0, -1, -1, -1, -1, -1, -1, -1, -1, -1/2) && (4.129)\\ b &= (1, 5, 5, 5, 5, 5, 5, 5, 5, 1) && (4.130)\\ c &= (-1, -1, -1, -1, -1, -1, -1, -1, -1, 0) && (4.131)\\ d &= (0, \cos(2/10), \cos(3/10), \cos(4/10), \cos(5/10), && (4.132)\\ &\quad \cos(6/10), \cos(7/10), \cos(8/10), \cos(9/10), \cos(1),0) && (4.133)\\ x &= (0{,}324295, 0{,}324295, 0{,}317115, 0{,}305943, 0{,}291539, && (4.134)\\ &\quad 0{,}274169, 0{,}253971, 0{,}230846, 0{,}20355, 0{,}165301, 0{,}082650) && (4.135) \end{aligned}$$

E 4.6.1.
$\lambda = \frac{71\times 30}{41} \approx 51.95122$, para $\lambda = 51$: $k_1 = k_\infty = 350.4$, $k_2 = 262.1$. Para $\lambda = 52$: $k_1 = k_\infty = 6888$, $k_2 = 5163$.

E 4.6.2.
$k_1(A) = 36$, $k_2(A) = 18{,}26$, $K_\infty(A) = 20{,}8$.

E 4.6.3.
$k_1 = k_\infty = 6888$, $k_2 = \sqrt{26656567}$ e $k_1 = 180$, $k_2 = 128{,}40972$ e $k_\infty = 210$

E 4.6.4.
$\frac{18}{\varepsilon} + 3$. Quando $\varepsilon \to 0+$, a matriz converge para uma matriz singular e o número de condicionamento diverge para $+\infty$.

E 4.6.5.
As soluções são $[-0.0000990 \quad 0.0000098]^T$ e $[0.0098029 \quad 0.0990294]^T$. A grande variação na solução em função de pequena variação nos dados é devido ao mau condicionamento da matriz ($k_1 \approx 1186274.3$).
Exemplo de implementação:

```
A=[1e5 -1e4+1e-2; -1e4+1e-2 1000.1]
b1=[-10 1]'
b2=[-9.999 1.01]'
A\b1
A\b2
```

E 4.6.6. 0,695; 0,292; 0,188; 0,0237; 0,0123; 0,00967

E 4.7.4.
0,324295, 0,324295, 0,317115, 0,305943, 0,291539, 0,274169, 0,253971, 0,230846, 0,203551, 0,165301, 0,082650

E 4.7.5.
Permute as linhas 1 e 2.

E 4.8.1. $\lambda = 86.1785$ associado ao autovetor dado por $v_1 = [0.65968 \quad 0.66834 \quad 0.34372]^T$.

E 4.8.2.

Este exercício está sem resposta sugerida. Proponha uma resposta. Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

E 4.8.3. 158,726

E 4.8.4.

Este exercício está sem resposta sugerida. Proponha uma resposta. Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

E 4.9.3. Dica: $P(-1) = -3$, $P(1) = -1$ e $P(2) = 9$ produzem três equações lineares para os coeficientes a, b e c. Resp: a) $P(x) = 3x^2 + x - 5$, b) $A \approx 2.49$ e $B \approx -1.29$ c)$A_1 \approx 1.2872058$, $A_2 \approx -4.3033034$, $B_1 \approx 2.051533$ e $B_2 \approx -0.9046921$.

E 5.1.1. $\nabla f = [2xy - y\,\mathrm{sen}\,(xy), x^2 - x\,\mathrm{sen}\,(xy)]^T$

$$J_F = \begin{bmatrix} \cos(x) - x\,\mathrm{sen}\,(x) & 1 \\ -2e^{-2x+y} & e^{-2x+y} \end{bmatrix} \tag{5.37}$$

$$(J_L)_{ij} = a_{ij} \tag{5.38}$$

E 5.1.2.

Este exercício está sem resposta sugerida. Proponha uma resposta. Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

E 5.1.3. As curvas possuem dois pontos de intersecção. A posição exata destes pontos de intersecção é dada por $\left(\sqrt{2\sqrt{3}-3}, 2\sqrt{3}-2\right)$ e $\left(-\sqrt{2\sqrt{3}-3}, 2\sqrt{3}-2\right)$. Use a solução exata para comparar com a solução aproximada obtida.

E 5.1.4. $(\pm 0.8241323, 0.6791941)$

E 5.1.5. $x \approx 0{,}259751$, $y \approx 0{,}302736$, $z \approx 0{,}045896$

E 5.1.6.

Este exercício está sem resposta sugerida. Proponha uma resposta. Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

E 5.1.7. $y = mx + b$ com $m \approx -0.0459710$ e $b \approx 0.479237$ Uma metodologia possível para resolver este problema é dada a seguir:

Sejam x_1 e x_2 as abscissas dos dois pontos em que a reta tangencia a curva. A equação da reta bitangente assume a seguinte forma:

$$y = f(x_1) + m(x - x_1) \tag{5.45}$$

onde o coeficiente angular m é dado por

$$m = \frac{f(x_2) - f(x_1)}{x_2 - x_1} \tag{5.46}$$

Da condição de tangência, temos que o coeficiente angular da reta, m, deve igual à derivada da função $f(x)$ nos dois pontos de tangência.

$$m = f'(x_1) = f'(x_2) \tag{5.47}$$

E sabemos que:

$$f'(x) = \frac{\cos(x)}{1+x} - \frac{\text{sen}\,(x)}{(1+x)^2}. \tag{5.48}$$

Assim, podemos reescrever o problema como

$$\frac{\cos(x_1)}{1+x_1} - \frac{\text{sen}\,(x_1)}{(1+x_1)^2} - \frac{\cos(x_2)}{1+x_2} + \frac{\text{sen}\,(x_2)}{(1+x_2)^2} = 0 \tag{5.49}$$

$$\frac{\cos(x_1)}{1+x_1} - \frac{\text{sen}\,(x_1)}{(1+x_1)^2} - \frac{f(x_2) - f(x_1)}{x_2 - x_1} = 0 \tag{5.50}$$

Este é um sistema não linear de duas incógnitas.

Os valores iniciais para o método podem ser obtidos do gráfico buscando valores próximos aos dois primeiros pontos de máximos. Por exemplo: $x_1^{(0)} = 1$ e $x_2^{(0)} = 8$. Obtemos $x_1 \approx 1{,}2464783$ e $x_2 \approx 8{,}1782997$ e m pode ser obtido através desses valores.

E 5.1.8. $(0.1956550; 0.2441719)$, $(0.3694093; 0.4590564)$, $(0.9990712; 1.1865168)$ e $(1.4773606; 1.5552232)$

E 5.1.9. $(0.0449310; 0.0648872; 0.0698750)$, $(0.3981385; 0.5658310; 0.6069019)$,

$(1.1862966; 1.4348545; 1.480127)$

E 5.1.10. $(-1{,}2085435, -1{,}0216674)$ e $(2{,}7871115, 1{,}3807962)$

E 5.1.11. A primeira curva trata-se de uma elipse de centro (3,1) e semi-eixos 4 e 6, portanto seus pontos estão contidos no retângulo $-1 \leq x \leq 7$ e $-5 \leq y \leq 7$.

As soluções são $(-0{,}5384844, -1{,}7978634)$ e $(2{,}8441544, 6{,}9954443)$.

E 5.1.12. $(x_1, x_2, x_3) \approx (453{,}62,\ 901{,}94,\ 144{,}43)$

E 5.1.13. Inicialização do método: $A^{(0)} = 3{,}1$ e $b^{(0)} = \sqrt{\frac{6{,}7}{3{,}1}}$ $A \approx 3.0297384$ e $b \approx 1.4835346$.

E 5.1.14. $f(-1{,}1579702, -1{,}2020694) \approx 2.376985$

E 5.1.15.

Este exercício está sem resposta sugerida. Proponha uma resposta. Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

E 5.1.16. $x \approx 0{,}2982646$, $y \approx -0{,}2990796$, $z \approx -1{,}6620333$ e $x \approx -0{,}0691328$, $y \approx 0{,}2923039$, $z \approx -0{,}8235705$.

E 5.1.17.

$$F(x) = \begin{bmatrix} x_1 - x_2 \\ -x_1 + 5(x_2 + x_2^3) - x_3 - 10\exp(-2/3) \\ -x_2 + 5(x_3 + x_3^3) - x_4 - 10\exp(-3/3) \\ -x_3 + 5(x_4 + x_4^3) - x_5 - 10\exp(-4/3) \\ \vdots \\ -x_9 + 5(x_{10} + x_{10}^3) - x_{11} - 10\exp(-10/3) \\ x_{11} - 1 \end{bmatrix} \tag{5.74}$$

$$J_F(x) = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & \dots & 0 \\ -1 & 5(1+3x_2^2) & -1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 5(1+3x_3^2) & -1 & 0 & \dots & 0 \\ 0 & 0 & -1 & 5(1+3x_4^2) & -1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{5.75}$$

Resposta final: 0,80447, 0,80447, 0,68686, 0,57124, 0,46535, 0,37061, 0,28883, 0,22433, 0,19443, 0,28667, 1

E 5.1.18. $f(0{,}8108792, 1{,}6217584) \approx 0{,}1950369$ e $f(0{,}5527864, 1{,}1055728) \approx 0{,}1455298$

E 6.1.1. $p(x) = -3 + 2x + 5x^3$.

E 6.1.2. $p(x) = 0{,}25 + x^2$.

E 6.4.1.
$\int_0^1 P(x)dx = \frac{f(0)+f(1)}{2}$, $\frac{1}{12}\max_{x\in[0,1]}|f''(x)|$

E 7.1.1. $f(x) = -0{,}55 - 0{,}01x$.

E 7.1.2. $f(x) = 0{,}19 - 0{,}47x$.

E 7.1.3. a) $-0{,}6025387$; b) $-0{,}5651848$; c) $0{,}2851848$; d) $0{,}1488041$.

E 7.2.1. $a_1 = -0{,}67112$, $a_2 = -0{,}12123$, $a_3 = 0{,}73907$.

E 7.2.2. $y = -0{,}0407898x^2 + 2{,}6613293x + 1{,}9364598$.

E 7.2.3. a) $a = 25{,}638625$, $b = 9{,}8591874$, $c = 4{,}9751219$; b)$a = 31{,}475524$, $b = 65{,}691531$, $c = -272{,}84382$, $d = 208{,}23621$.

E 8.1.1.

a) $f'(x)$ onde $f(x) = \text{sen}\,(x)$ e $x = 2$ para $h = 10^{-2}$ e $h = 10^{-3}$, respectivamente.
Progressiva ordem 1: $-0{,}42069$ e $-0{,}41660$.
Regressiva ordem 1: $-0{,}41159$ e $-0{,}41569$.
Central ordem 2: $-0{,}41614$ e $-0{,}41615$.
Exata: $\cos(2) = -0{,}41615$

b) $f'(x)$ onde $f(x) = e^{-x}$ e $x = 1$ para $h = 10^{-2}$ e $h = 10^{-3}$, respectivamente.

Progressiva ordem 1: $-0{,}36605$ e $-0{,}36788$.

Regressiva ordem 1: $-0{,}36972$ e $-0{,}36806$.

Central ordem 2: $-0{,}36789$ e $-0{,}36788$.

Exata: $-e^{-1} = -0{,}36788$

E 8.1.3.

a) $f'(0) = \frac{-3f(0)+4f(h)-f(2h)}{2h} + \mathcal{O}(h^2)$

b) $f'(0) = \frac{3f(0)-4f(-h)+f(-2h)}{2h} + \mathcal{O}(h^2)$

c) $f'(0) = \frac{1}{h_1+h_2} l \left[-\frac{h_2}{h_1} f(-h_1) + \left(\frac{h_2}{h_1} - \frac{h_1}{h_2} \right) f(0) + \frac{h_1}{h_2} f(h_2) \right]$

E 8.1.4.

$Caso$	a	b	c	d
$v_i = 1$	1.72	1.56	1.64	1.86
$v_i = 4.5$	2.46	1.90	2.18	1.14

(8.41)

E 8.1.5.

Segue a tabela com os valores da derivada para vários valores de h.

h	10^{-2}	10^{-4}	10^{-6}	10^{-7}	10^{-8}	10^{-9}
$D_{+,h}f(1{,}5)$	$-0{,}3125246$	$-0{,}3161608$	$-0{,}3161973$	$-0{,}3161976$	$-0{,}3161977$	$-0{,}3161977$

(8.42)

h	10^{-10}	10^{-11}	10^{-12}	10^{-13}	10^{-14}	10^{-15}
$D_{+,h}f(1{,}5)$	$-0{,}3161976$	$-0{,}3161971$	$-0{,}3162332$	$-0{,}3158585$	$-0{,}3178013$	$-0{,}3747003$

(8.43)

Observe que o valor exato é $-0{,}3161977$ e o h ótimo é algo entre 10^{-8} e 10^{-9}.

E 8.2.1.

a) $f''(0) = \frac{f(0)-2f(h)+f(2h)}{h^2} + \mathcal{O}(h)$

b) $f''(0) = \frac{f(0)-2f(-h)+f(-2h)}{h^2} + \mathcal{O}(h)$

E 8.4.2. $f''(x^*) = \frac{f(x_0)-2f(x_1)+f(x_2)}{h^2}$

E 9.2.2.

$$I_{Simpson} = \frac{1}{3} I_{Trap} + \frac{2}{3} I_{PM} \tag{9.90}$$

E 9.2.3.

n	Ponto médio	Trapézios	Simpson
3	0.1056606	0.7503919	0.5005225
5	0.1726140	0.3964724	0.2784992
7	0.1973663	0.3062023	0.2393551
9	0.2084204	0.2721145	0.2306618

(9.91)

E 9.4.2.
-0.2310491, -0.2452073, - 0.2478649.

E 9.4.4.
a)-0.2472261, -0.2416451, -0.2404596, -0.2400968, -0.2399563, -0.2398928. b)-0.2393727, -0.2397994, -0.2398104, -0.2398115, -0.2398117, -0.2398117.

E 9.5.1.

$$a) I(h) = 4.41041 \cdot 10^{-1} - 8.49372 \cdot 10^{-12} h - 1.22104 \cdot 10^{-2} h^2 - 1.22376 \cdot 10^{-7} h^3 + 8.14294 \cdot 10^{-3} h^4 \quad (9.152)$$

$$b) I(h) = 7.85398 \cdot 10^{-1} - 1.46294 \cdot 10^{-11} h - 4.16667 \cdot 10^{-2} h^2 - 2.16110 \cdot 10^{-7} h^3 + 4.65117 \cdot 10^{-6} h^4 \quad (9.153)$$

$$c) I(h) = 1.58730 \cdot 10^{-3} - 9.68958 \cdot 10^{-10} h + 2.03315 \cdot 10^{-7} h^2 - 1.38695 \cdot 10^{-5} h^3 + 2.97262 \cdot 10^{-4} h^4 \quad (9.154)$$

$$d) I(h) = 4.61917 \cdot 10^{-1} + 3.83229 \cdot 10^{-12} h + 2.52721 \cdot 10^{-2} h^2 + 5.48935 \cdot 10^{-8} h^3 + 5.25326 \cdot 10^{-4} h^4 \quad (9.155)$$

E 9.5.2.

1.5707963	2.0943951		
1.8961189	2.0045598	1.9985707	
1.9742316	2.0002692	1.9999831	2.0000055

E 9.5.3. a) 0.7468337; b) 2.4606311; c) 1.6595275.

E 9.5.5. $R(6,6) = -10.772065$, $R(7,7) = 5.2677002$, $R(8,8) = 6.1884951$, $R(9,9) = 6.0554327$, $R(10,10) = 6.0574643$. O valor desta integral com oito dígitos corretos é aproximado por 6.0574613.

E 9.6.1.
$w_1 = 1/6$, $w_2 = 2/3$, $w_3 = 1/6$. O esquema construído é o de Simpson e a ordem de exatidão é 3.

E 9.6.2. 3

E 9.6.3. 5

E 9.6.4. $\int_0^1 f(x)\,dx \approx \frac{3}{2} f(1/3) - 2f(1/2) + \frac{3}{2} f(2/3)$ com ordem 3.

E 9.6.5. 5, 4, 3

E 9.7.1.

n	G-L	Exato	Erro Absoluto
hline 2	0,2227		2,47E−01
3	0,4157	0,4701	5,44E−02
4	0,4437		2,64E−02
5	0,4616		8,47E−03

E 9.9.1.

n	b	c	d	e	f
2	2.205508	3.5733599	3.6191866	3.6185185	3.618146
4	2.5973554	3.6107456	3.6181465	3.6180970	3.6180970
6	2.7732372	3.6153069	3.6181044	3.6180970	3.6180970
8	2.880694	3.6166953	3.6180989	3.6180970	3.6180970

Solução do item e: Como

$$\cos(x) = 1 + \sum_{n=1}^{\infty} (-1)^n \frac{x^{2n}}{(2n)!} \tag{9.230}$$

temos

$$\frac{1-\cos(x)}{\sqrt{x}} = -\sum_{n=1}^{\infty} (-1)^n \frac{x^{2n-1/2}}{(2n)!}, \quad x \geq 0 \tag{9.231}$$

Logo, podemos integrar

$$I = 4 + 2\int_0^1 \frac{\cos(x)-1}{\sqrt{|x|}}dx = 4 - 2\sum_{n=1}^{\infty}(-1)^n \int_0^1 \frac{x^{2n-1/2}}{(2n)!}dx \tag{9.232}$$

$$= 4 - 2\sum_{n=1}^{\infty}(-1)^n \frac{1}{(2n)!(2n+1/2)} \tag{9.233}$$

Solução do item f)

$$2\int_0^1 \left(x^{-1/2} - \frac{x^{3/2}}{2} + \frac{x^{7/2}}{24}\right)dx = 2\left(2 - \frac{1}{5} + \frac{1}{54}\right) = \frac{977}{270} \tag{9.234}$$

$$2\int_0^1 \frac{\cos(x) - P_4(x)}{\sqrt{x}}dx = \sqrt{2}\int_{-1}^1 \frac{\cos\left(\frac{1+u}{2}\right) - P_4\left(\frac{1+u}{2}\right)}{\sqrt{1+u}}du \tag{9.235}$$

E 9.9.5. 4,1138

E 9.9.6. a) 19,2; 22,1; 23,3; b) 513,67K

E 9.9.8. $\displaystyle\int_{-1}^1 f(x)dx = f\left(-\frac{\sqrt{3}}{3}\right) + f\left(\frac{\sqrt{3}}{3}\right)$

E 9.9.9. $w_1 = w_3 = 1$ e $w_2 = 0$ com ordem 3.

E 10.2.1.

t	Exato	Euler $h = 0{,}1$	Euler $h = 0{,}01$
0	1	1	1
1	$2e^{-1} \approx 0{,}7357589$	0,6973569	0,7320647
2	$2e^{-2} + 1 \approx 1{,}2706706$	1,2431533	1,2679593
3	$2e^{-3} + 2 \approx 2{,}0995741$	2,0847823	2,0980818

E 10.2.2.

t	Exato	Euler $h = 0{,}1$	Euler $h = 0{,}01$
0	0	0	0
1	0,8657695	0.8799602	0.8671764
2	1,3017603	1.3196842	1.3035243
3	1,4713043	1.4827638	1.4724512

E 10.2.3. Aproximação via Euler: 2,4826529, exata: $e^{\text{sen }(2)} \approx 2{,}4825777$. Erro relativo aproximado: 3×10^{-5}.

E 10.5.1.

h		$2 \cdot 10^{-2}$	$2 \cdot 10^{-2}$	$2 \cdot 10^{-3}$	$2 \cdot 10^{-4}$
Euler	x	0,4302019	0,4355057	0,4358046	0,4358324
	y	0,6172935	0,6457760	0,6486383	0,6489245
Euler mod,	x	0,4343130	0,4358269	0,4358354	0,4358355
	y	0,6515479	0,6489764	0,6489566	0,6489564

E 10.5.2.

h		$t = 0{,}5$	$t = 1{,}0$	$t = 1{,}5$	$t = 2{,}0$
10^{-3}	x	1,9023516	1,6564208	1,3124281	0,9168299
	y'	-0,3635613	-0,6044859	-0,7564252	-0,8072298
10^{-4}	x	1,9023552	1,6564243	1,3124309	0,9168319
	y'	-0,3635670	-0,6044930	-0,7564334	-0,8072397

E 10.6.1.

h	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
Euler	0,4495791	0,4660297	0,4675999	0,4677562	0,4677718
ε_{rel}	9,1e-03	8,9e-04	8,9e-05	8,9e-06	8,9e-07
Euler mod,	0,4686037	0,4677811	0,4677736	0,4677735	0,4677735
ε_{rel}	1,8e-03	1,6e-05	1,6e-07	1,6e-09	1,6e-11

A solução exata vale $u(1) = \frac{1+2e^{-1}+e^{-2}}{4} = \left(\frac{1+e^{-1}}{2}\right)^2 \approx 0.467773541395$.

E 10.6.2.

h	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
Euler	1,1617930	1,1395726	1,1374484	1,1372369	1,1372157
Euler mod	1,1365230	1,1372075	1,1372133	1,1372134	1,1372134

E 10.7.1. 2,9677921, 2,9682284 e 2,9682325.

E 10.7.2.

Euler	6,0000000	6,7898955	6,8461635	6,8516386
ε_{rel}	1,2e-01	9,1e-03	8,9e-04	8,9e-05
Euler mod,	7,0000000	6,8532949	6,8522554	6,8522454
ε_{rel}	2,2e-02	1,5e-04	1,5e-06	1,5e-08
RK_3	6,8333333	6,8522321	6,8522453	6,8522453
ε_{rel}	2,8e-03	1,9e-06	1,9e-09	1,8e-12
RK_4	6,8541667	6,8522454	6,8522453	6,8522453
ε_{rel}	2,8e-04	1,9e-08	1,9e-12	1,3e-15

E 10.7.3.

Euler	2,0000000	3,1874849	3,4096277	3,4338479
ε_{rel}	4,2e-01	7,2e-02	7,8e-03	7,9e-04
Euler mod	3,0000000	3,4281617	3,4364737	3,4365628
ε_{rel}	1,3e-01	2,4e-03	2,6e-05	2,6e-07
RK_3	3,3333333	3,4363545	3,4365634	3,4365637
ε_{rel}	3,0e-02	6,1e-05	6,5e-08	6,6e-11
RK_4	3,4166667	3,4365595	3,4365637	3,4365637
ε_{rel}	5,8e-03	1,2e-06	1,3e-10	1,2e-14

E 10.11.1.

$$y^{(n)} = y^{(n)} + hf\left(t^{(n)}, u(t^{(n)})\right) \tag{10.342}$$

Este esquema é equivalente ao método de Euler Implícito.

E 10.11.2.

$$y^{(n+1)} = y^{(n)} + \frac{h}{2}\left[f\left(t^{(n+1)}, u(t^{(n+1)})\right) + f\left(t^{(n)}, u(t^{(n)})\right)\right] \tag{10.343}$$

Este esquema é equivalente ao método trapezoidal.

E 10.11.3. 0,37517345 e 0,37512543.

E 10.13.1.

$$\tilde{u}^{(n+1)} = u^{(n)} + \frac{h}{24}\left[-9f(t^{(n-3)}, u^{(n-3)}) + 37f(t^{(n-2)}, u^{(n-2)}) - 59f(t^{(n-1)}, u^{(n-1)}) + 55f(t^{(n)}, u^{(n)})\right], \tag{10.367}$$

$$u^{(n+1)} = u^{(n)} + \frac{h}{24}\left[f(t^{(n-2)}, u^{(n-2)}) - 5f(t^{(n-1)}, u^{(n-1)}) + 19f(t^{(n)}, u^{(n)}) + 9f(t^{(n+1)}, \tilde{u}^{(n+1)})\right]. \tag{10.368}$$

E 10.13.2. Adams-Bashforth: 34,99965176, Preditor-corretor: 34,99965949, Exato: 35

E 10.17.1.

	0,5	1,0	1,5	2,0	2,5
Analítico	0,3032653	0,3678794	0,3346952	0,2706706	0,2052125
Euler	0,3315955	0,3969266	0,3563684	0,2844209	0,2128243
Euler modificado	0,3025634	0,3671929	0,3342207	0,2704083	0,2051058
Runge-Kutta clássico	0,3032649	0,3678790	0,3346949	0,2706703	0,2052124
Adams-Bashforth ordem 4	0,3032421	0,3678319	0,3346486	0,2706329	0,2051848

	0,5	1,0	1,5	2,0	2,5
Euler	2,8e-2	2,9e-2	2,2e-2	1,4e-2	7,6e-3
Euler modificado	7,0e-4	6,9e-4	4,7e-4	2,6e-4	1,1e-4
Runge-Kutta clássico	4,6e-7	4,7e-7	3,5e-7	2,2e-7	1,2e-7
Adams-Bashforth ordem 4	2,3e-5	4,8e-5	4,7e-5	3,8e-5	2,8e-5

	0,1	0,05	0,01	0,005	0,001
Euler	2,9e-2	5,6e-3	2,8e-3	5,5e-4	2,8e-4
Euler modificado	6,9e-4	2,5e-5	6,2e-6	2,5e-7	6,1e-8
Runge-Kutta clássico	4,7e-7	6,9e-10	4,3e-11	6,8e-14	4,4e-15
Adams-Bashforth ordem 4	4,8e-5	9,0e-8	5,7e-9	9,2e-12	5,8e-13

E 10.17.2.

a) 1,548280989603, 2,319693166841, 9,42825618574 e 9,995915675174.

b) 0,081093021622.

c) 0,179175946923.

Obs: A solução analitica do problema de valor inicial é dada por:

$$u(t) = \frac{Au_0}{(A - u_0)e^{-A\alpha at} + u_0} \tag{10.394}$$

Os valores exatos para os itens b e c são: $\frac{1}{10}\ln\left(\frac{9}{4}\right)$ e $\frac{1}{10}\ln(6)$.

E 10.17.3. O valor exato é $\sqrt{\frac{g}{\alpha}\left[1-e^{-200\alpha}\right]} \approx 29{,}109644835142$ em $t = \frac{1}{\sqrt{g\alpha}} \tanh^{-1}\left(\sqrt{1-e^{-200\alpha}}\right) \approx 2{,}3928380185497$

E 11.1.1.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \\ 2 \\ 2 \\ 10 \end{bmatrix} \tag{11.39}$$

Solução: [5, 9.25, 11.5, 11.75, 10]

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{bmatrix} = \begin{bmatrix} 5 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \\ 0.5 \\ 10 \end{bmatrix} \tag{11.40}$$

Solução: $[5, 7.375, 9.25, 10.625, 11.5, 11.875, 11.75, 1.125, 10]$

E 11.1.2. 120. 133.56 146.22 157.83 168.22 177.21 184.65 190.38 194.28 196.26 196.26 194.26 190.28 184.38 176.65 167.21 156.22 143.83 130.22 115.56 100.

E 11.1.3. 391.13 391.13 390.24 388.29 385.12 380.56 374.44 366.61 356.95 345.38 331.82 316.27 298.73 279.27 257.99 234.99 210.45 184.5 157.34 129.11 100.

E 11.1.4. 0., 6.57, 12.14, 16.73, 20.4, 23.24, 25.38, 26.93 , 28, 28.7, 29.06, 29.15, 28.95, 28.46, 27.62 , 26.36, 24.59, 22.18, 19.02, 14.98, 10.

E 11.1.5. $u(0) = 31.62$, $u(1) = 31{,}50$, $u(1{,}9) = 18{,}17$.

E 11.1.6. $u(1) = 1{,}900362$, $u(2{,}5) = 1.943681$, $u(4) = 1{,}456517$.

Referências Bibliográficas

[1] Cecill and free sofware. http://www.cecill.info. Acessado em 30 de julho de 2015.

[2] M. Baudin. Introduction to scilab. http://forge.scilab.org/index.php/p/docintrotoscilab/. Acessado em 30 de julho de 2015.

[3] R.L. Burden and J.D. Faires. *Análise Numérica*. Cengage Learning, 8 edition, 2013.

[4] J. P. Demailly. *Analyse Numérique et Équations Differentielles*. EDP Sciences, Grenoble, nouvelle Édition edition, 2006.

[5] W Gautschi. Numerical analysis: An introduction birkhauser. *Barton, Mass, USA*, 1997.

[6] Walter Gautschi and Gabriele Inglese. Lower bounds for the condition number of vandermonde matrices. *Numerische Mathematik*, 52(3):241–250, 1987/1988.

[7] L.F. Guidi. Notas da disciplina cálculo numérico. http://www.mat.ufrgs.br/~guidi/grad/MAT01169/calculo_numerico.pdf. Acessado em julho de 2016.

[8] E. Isaacson and H.B. Keller. *Analysis of numerical methods*. Dover, Ontário, 1994.

[9] Arieh Iserles. *A first course in the numerical analysis of differential equations*. Cambridge university press, 2009.

[10] W.H. Press. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007.

[11] R. Rannacher. Einführung in die numerische mathematik (numerik 0). http://numerik.uni-hd.de/~lehre/notes/num0/numerik0.pdf. Acessado em 10.08.2014.

[12] Todos os Colaboradores. Cálculo numérico - um livro colaborativo - versão com scilab. disponível em https://www.ufrgs.br/reamat/CalculoNumerico/livro-sci/main.html, Novembro 2016.

Índice Remissivo