Notas de aula: Cálculo Numérico

utilizando a linguagem Julia

Daniel Cassimiro

./rosto/capa-eps-converted-to.pdf

September 12, 2024

Licença

Contents

Chapter 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:

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

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.

Chapter 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.

2.1 Sistema de numeração e mudança de base

Usualmente, utilizamos o sistema de numeração decimal, isto é, base 10, para representar números. Esse é um sistema de numeração em que a posição do algarismo indica a potência de 10 pela qual seu valor é multiplicado.

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*

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

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

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 $-$ à 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 `Julia` podemos converter o número $(1001{,}101)_2$ para a base decimal computando

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

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!

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

Exemplo 2.1.5 (Sistema hexadecimal)**.** O sistema de numeração cuja 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. A linguagem `Julia` tem prefixos para representar números nas bases 2, 8 e 10. Por exemplo, temos:

```
>>> print(0b1001) # bin -> dec
9
>>> print(0o157) # oct -> dec
111
>>> print(0xbeba) # hex -> dec
48826
>>> string(9,base=2) # dec -> bin
1001
```

Também é possível usar a função int() para interpretar a representação de um inteiro em uma base com b entre 2 e 36. Por exemplo, temos:

```
>>> print(int('1001', 2)) # Base 2
9
>>> print(int('1001', 5)) # Base 5
126
>>> print(int('ABCD', 20)) # Base 20
84653
>>> print(int('zz', 36)) # Base 36
1295
```

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 o quociente 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 `Julia`, 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(f"d0 = {d0}, x = {x}")
d0 = 1, x = 4
```

```
>>> d1 = x%2; x = int(x/2); print(f"d1 = {d1}, x = {x}")
d1 = 0, x = 2
>>> d2 = x%2; x = int(x/2); print(f"d2 = {d2}, x = {x}")
d2 = 0, x = 1
>>> d3 = x%2; x = int(x/2); print(f"d3 = {d3}, x = {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 $0{,}625 = (0{,}101)_2$.

No `GNU Octave`, podemos computar esta conversão da parte fracionária da seguinte forma

```
>> x = 0.625
x =  0.62500
>> d = fix(2*x), x = 2*x - d
d =  1
x =  0.25000
>> d = fix(2*x), x = 2*x - d
d = 0
x =  0.50000
>> d = fix(2*x), x = 2*x - d
d =  1
x = 0
```

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. O `GNU Octave` oferece algumas funções para a conversão de números inteiros em base decimal para uma base dada. Por exemplo, temos:

```
>> dec2base(9,2)
ans = 1001
>> dec2base(111,8)
ans = 157
>> dec2base(48826,16)
ans = BEBA
```

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_1 para base decimal e depois converter para a base b_2.

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 no `GNU Octave` com o auxílio das funções `mod` e `fix`. A primeira calcula o resto da divisão entre dois números, enquanto que a segunda retorna a parte inteira de um número dado. No nosso exemplo, temos:

```
>> x = 125
x =  125
>> d = mod(x,6), x = fix(x/6)
d =  5
x =  20
>> d = mod(x,6), x = fix(x/6)
d =  2
x =  3
>> d = mod(x,6), x = fix(x/6)
d =  3
x = 0
```

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 $0{,}58\overline{3} = (0{,}33)_6$.

No `GNU Octave`, podemos computar esta conversão da parte fracionária da seguinte forma

```
>> x = 0.58 + 1/3/100
x =  0.58333
>> d = fix(6*x), x = 6*x - d
d =  3
x =  0.50000
>> x = 0.5 #isso é realmente necessário?
x =  0.50000
>> d = fix(6*x), x = 6*x - d
d =  3
x = 0
```

$\Diamond$

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:

$$(101{,}01)_2 = 1 \cdot 2^2 + 1 \cdot 2^0 + 1 \cdot 2^{-2} = 5{,}25. \tag{2.23}$$

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! $\Diamond$

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 $d_0 \neq 0$[3][4], 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. No `GNU Octave`, podemos controlar a impressão de números usando o comando `printf`. Por exemplo:

[3]Em algumas referências é usado $M_b = (0{,}d_{-1}d_{-2}d_{-3}\cdots)_b$.
[4]No caso de $x = 0$, $M_b = (0{,}00\cdots)_b$.

```
>> printf('%1.5f\n',-pi)
-3.14159
>> printf('%1.5e\n',-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

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:

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

(2.31)

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

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

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

[5] caso $x \neq 0$.

[6] com desempate infinito.

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

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

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

```
>> round(1.49)
ans =  1
>> round(1.50)
ans =  2
>> round(2.50)
ans =  3
```

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:

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

No `GNU Octave`, podemos obter a representação de $x_3 = -0{,}675$ fazendo:

```
>> printf("%1.2f\n", ceil(-0.675*1e2)/1e2)
-0.67
```

e, em notação normalizada, temos:

```
>> printf("%1.1e\n", ceil(-0.675*1e2)/1e2)
-6.7e-01
```

Em `GNU Octave`, a representação de números por arredondamento por proximidade com desempate par é o padrão. Assim, para obtermos a representação desejada de $x_3 = 0{,}675$ fazemos:

```
>> printf("%1.2f\n", -0.675)
-0.68
```

e, em notação normalizada, temos:

```
>> printf("%1.1e\n", -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 `GNU Octave` usa arredondamento por proximidade com desempate par como padrão. Assim sendo, por exemplo

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

Agora:

```
>> printf('%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 no `GNU Octave`:

```
>> 2^2 == 4
ans =  1
>> sqrt(3)^2 == 3
ans = 0
```

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

$$\begin{aligned}
[111\ldots111] &= 2^{n-1} + 2^{n-2} + \cdots + 2^1 + 2^0 = 2^n - 1,\\
&\vdots\\
[000\ldots011] &= 3,\\
[000\ldots010] &= 2,\\
[000\ldots001] &= 1,\\
[000\ldots000] &= 0.
\end{aligned} \tag{2.43}$$

Exemplo 2.4.1. No `Scilab`,

```
-->uint8( bin2dec('00000011') )
   ans = 3
-->uint8( bin2dec('11111110') )
   ans = 254
```

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 $(-1)^s(d_{n-2}\ldots d_1d_0)_2$. Assim, é possível representar números inteiros entre $1-2^{n-1}$ e $2^{n-1}-1$, com duas representações para o zero: $(1000\ldots000)_2$ e $(00000\ldots000)_2$.

Exemplo 2.4.2. Em um registro com 8 bits, teremos os números

$$\begin{aligned}
&[11111111] = -(2^6 + \cdots + 2 + 1) = -127,\\
&\qquad\vdots\\
&[10000001] = -1,\\
&[10000000] = -0,\\
&[01111111] = 2^6 + \cdots + 2 + 1 = 127,\\
&\qquad\vdots\\
&[00000010] = 2,\\
&[00000001] = 1,\\
&[00000000] = 0.
\end{aligned} \tag{2.44}$$

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 $-d_{n-1}2^{n-1} + (d_{n-2} \ldots d_1 d_0)_2$.

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

Exemplo 2.4.3. O registro com 8 bits [01000011] representa o número:

$$-0(2^7) + (1000011)_2 = 2^6 + 2 + 1 = 67. \tag{2.45}$$

Agora, o registro [10111101] representa:

$$-1(2^7) + (0111101)_2 = -128 + 2^5 + 2^4 + 2^3 + 2^2 + 1 = -67. \tag{2.46}$$

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

Exemplo 2.4.4. Em um registro com 8 bits, teremos os números

$$\begin{aligned}
[11111111] &= -2^7 + 2^6 + \cdots + 2 + 1 = -1 \\
&\vdots \\
[10000001] &= -2^7 + 1 = -127 \\
[10000000] &= -2^7 = -128 \\
[01111111] &= 2^6 + \cdots + 2 + 1 = 127 \\
&\vdots \\
[00000010] &= 2 \\
[00000001] &= 1 \\
[00000000] &= 0
\end{aligned} \tag{2.47}$$

O `GNU Octave` trabalha com representação complemento de 2 de números inteiros. O comando `bitunpack` mostra o barramento de um número dado, por exemplo:

```
>> bitunpack(int8(3))
ans =
   1   1   0   0   0   0   0   0
```

mostra o barramento com 8 **bits** do número inteiro 3. Note que a ordem dos **bits** é inversa daquela apresentada no texto acima. Aqui, o **bit** mais à esquerda fornece o coeficiente de 2^0, enquanto o **bit** mais à direita fornece o coeficiente de -2^7.

O comando `bitpack` converte um barramento para o número em decimal, por exemplo:

```
>> bitpack(logical([1 1 0 0 0 0 0 0]),'int8')
ans = 3
```

2.4.2 Sistema de ponto fixo

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

Exemplo 2.4.5. 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:

$$[10000000000000000000000000000000] \tag{2.48}$$

e

$$[00000000000000000000000000000000] \tag{2.49}$$

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

$$[11111111111111111111111111111111] \tag{2.50}$$

e

$$[00000000000000000000000000000000] \tag{2.51}$$

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

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

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:

$$F(\beta,|M|,|E|,BIAS) \text{ ou } F(\beta,|M|,E_{MIN},E_{MAX}) \tag{2.53}$$

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)

$$x = (-1)^s M \times 2^{c-BIAS}, \tag{2.54}$$

onde a **característica** é representada por

$$c = (c_{10}c_9 \cdots c_1c_0)_2 = c_{10}2^{10} + \cdots + c_1 2^1 + c_0 2^0 \tag{2.55}$$

e o significando por

$$M = (1.m_1m_2 \cdots m_{51}m_{52})_2. \tag{2.56}$$

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

Exemplo 2.4.6. O registro

$$[0|100\ 0000\ 0000|1010\ 0000\ 0000 \ldots 0000\ 0000] \tag{2.57}$$

representa o número

$$(-1)^0(1 + 2^{-1} + 2^{-3}) \times 2^{1024-1023} = (1 + 0.5 + 0.125)2 = 3.25. \tag{2.58}$$

Observação 2.4.3. No `GNU Octave`, podemos usar os comandos `bitpack` e `bitunpack` transformar um registro de ponto flutuante de 64 **bits** em decimal e vice-versa. Entretanto, um tal registro no `GNU Octave` tem a seguinte estrutura

$$[m_{52}m_{51}m_{50} \ldots m_1|c_0c_1c_2 \cdots c_{10}|s]. \tag{2.59}$$

Desta forma, o decimal 3.25 tem, aqui, o registro

$$[000 \ldots 0101|000 \ldots 01|0]. \tag{2.60}$$

O que podemos verificar com o comando

```
>> bitpack(logical([zeros(1,49) 1 0 1 zeros(1,10) 1 0]),'double')
ans =  3.2500
```

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 $[100\ldots000]$.

Exemplo 2.4.7. 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

$$1024 - BIAS = 1. \tag{2.61}$$

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

Com $|E| = 11$ temos

$$\begin{aligned}
&[111\ 1111\ 1111] = \text{reservado}\\
&[111\ 1111\ 1110] = 2046 - BIAS = 1023_{10} = E_{MAX}\\
&\qquad\qquad\vdots =\\
&[100\ 0000\ 0001] = 2^{10} + 1 - BIAS = 2_{10}\\
&[100\ 0000\ 0000] = 2^{10} \quad - BIAS = 1_{10}\\
&[011\ 1111\ 1111] = 1023 \quad - BIAS = 0_{10}\\
&[011\ 1111\ 1110] = 1022 \quad - BIAS = -1_{10}\\
&\qquad\qquad\vdots =\\
&[000\ 0000\ 0001] = 1 - BIAS = -1022 = E_{MIN}\\
&[000\ 0000\ 0000] = \text{reservado}
\end{aligned} \tag{2.62}$$

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

$$[0|000\ 0000\ 0001|0000\ 0000\ 0000\ldots0000\ 0000] \tag{2.63}$$

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

$$MINR = (1+0)_2 \times 2^{1-1023} \approx 0.2225 \times 10^{-307}. \tag{2.64}$$

O maior número representável é dado por

$$[0|111\ 1111\ 1110|1111\ 1111\ \cdots\ 1111\ 1111] \tag{2.65}$$

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

$$MAXR = (2 - 2^{-52}) \times 2^{2046-1023} \approx 2^{1024} \approx 0.17977 \times 10^{309}. \tag{2.66}$$

Observação 2.4.4. No `GNU Octave`, podemos obter o maior e o menor `double` positivo não nulo com os comandos:

```
>> realmax
ans =   1.7977e+308
>> realmin
ans =   2.2251e-308
```

Casos especiais

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

$$[0|000\ 0000\ 0000|0000\ 0000\ 0000...0000\ 0000] \tag{2.67}$$

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

$$x = (-1)^s (0.m_1 m_2 \cdots m_{51} m_{52})_2 \times 2^{1-BIAS}. \tag{2.68}$$

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 $(1, 1 + \epsilon)$.

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

Exemplo 2.4.8. Com 64 bits, temos que o épsilon será dado por

$$\begin{aligned} 1 \rightarrow \quad & (1.0000\ 0000....0000)_2 \times 2^0 \\ \epsilon \rightarrow + & (0.0000\ 0000....0001)_2 \times 2^0 = 2^{-52} \\ & (1.0000\ 0000....0001)_2 \times 2^0 \neq 1 \end{aligned} \tag{2.69}$$

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

Observação 2.4.5. No `GNU Octave`, o épsilon de máquina é representado pela constante `eps`. Observe os seguintes resultados:

```
>> 1 + 1e-16 == 1
ans =  1
>> 1 + eps == 1
ans = 0
```

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.

Observação 2.4.6. Veja como o `GNU Octave` se comporta nos seguintes casos de exceção:

```
>> 0/0
warning: division by zero
ans = NaN
>> 1/0
warning: division by zero
ans = Inf
>> 1/-0
warning: division by zero
ans = -Inf
>> 2*realmax
ans = Inf
>> 1/2^9999
ans = 0
```

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 complemento de dois de números inteiros com 8 **bits** no `GNU Octave`, escreva o número decimal que corresponde aos seguintes barramentos:

a) [01100010].

b) [00011101].

c) [00010010].

E 2.4.4. Usando a representação complemento de dois de números inteiros com 16 **bits** no `GNU Octave`, escreva o número decimal que corresponde aos seguintes barramentos:

a) [0110001001100010].

b) [0001110100011101].

c) [0001001011100010].

E 2.4.5. 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 2.4.6. Explique a diferença entre o sistema de ponto fixo e ponto flutuante.

E 2.4.7. Usando a representação de `double` no `GNU Octave`, escreva o número decimal que corresponde aos seguintes barramentos:

a) [000 . . . 0111|00000000001|0].

b) [000 . . . 01110|10000000001|1].

E 2.4.8. Considere a seguinte rotina escrita para ser usada no `GNU Octave`:

```
x=1
while x+1>x
    x=x+1
end
```

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*

$$|x - \overline{x}|. \tag{2.70}$$

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

$$\frac{|x - \overline{x}|}{|x|}, \quad x \neq 0. \tag{2.71}$$

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

$$\frac{|x - \bar{x}|}{|x|} \times 100\%. \tag{2.72}$$

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

$$|x - \bar{x}| = |123456{,}789 - 123000| = 456{,}789 \tag{2.73}$$

e o erro relativo é

$$\frac{|x - \bar{x}|}{|x|} = \frac{456{,}789}{123456{,}789} \approx 0{,}00369999 \text{ ou } 0{,}36\% \tag{2.74}$$

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

$$|y - \bar{y}| = |1{,}23456789 - 1{,}13| = 0{,}10456789 \tag{2.75}$$

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

$$\frac{|y-\bar{y}|}{|y|} = \frac{0{,}10456789}{1{,}23456789} \approx 0{,}08469999 \text{ ou } 8{,}4\% \tag{2.76}$$

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]

$$\frac{|x-\overline{x}|}{|x|} < 5 \times 10^{-s}. \tag{2.78}$$

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

$$\frac{|x-\overline{x}|}{|x|} = \frac{0{,}000333}{0{,}333333} \approx 0{,}000999 \leq 5 \times 10^{-\mathbf{3}}. \tag{2.79}$$

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

$$\frac{|x-\bar{x}_1|}{|x|} = \frac{|0{,}666888 - 0{,}666|}{0{,}666888} \approx 0{,}00133... < 5 \times 10^{-3}. \tag{2.80}$$

[9]Esta definição é apresentada em [**?**]. 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

$$DIGSE(x,\bar{x}) = s \approx int\left|\log_{10}\frac{|x-\overline{x}|}{|x|}\right|. \tag{2.77}$$

$$\frac{|x-\bar{x}_2|}{|x|} = \frac{|0{,}666888 - 0{,}667|}{0{,}666888} \approx 0{,}000167... < 5 \times 10^{-4}. \tag{2.81}$$

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 à 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

$$\frac{|x-\overline{x}|}{|x|} = \frac{|10 - 9{,}999|}{10} \approx 0{,}0000999... < 5 \times 10^{-4}. \tag{2.82}$$

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:

$$\frac{|1 - 1{,}5|}{|1|} = 5 \times 10^{-1} < 5 \times 10^{0}. \tag{2.83}$$

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 **??**, 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 o sistema de numeração *binary64* ou *double*, temos $\epsilon = 2^{-52} \approx 2{,}22 \cdot 10^{-16}$.

Este erro é bem pequeno para a maioria das aplicações! Assumindo que x e y são representados com todos dígitos corretos, esperamos ter 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, existem situações em que o erro se propaga de forma muito catastrófica, em especial, quando subtraímos números positivos muito próximos.

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

$$0{,}987624687925 - 0{,}987624 = 0{,}687925 \times 10^{-6} \tag{2.87}$$

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

$$0{,}987625 - 0{,}987624 = 0{,}100000 \times 10^{-5} \tag{2.88}$$

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

$$\frac{|0{,}987624687925 - 0{,}987625|}{|0{,}987624687925|} = 0{,}00003159 \tag{2.89}$$

e

$$\frac{|0{,}987624 - 0{,}987624|}{|0{,}987624|} = 0\%, \tag{2.90}$$

enquanto no lado direito o erro relativo é enorme:

$$\frac{|0{,}100000 \times 10^{-5} - 0{,}687925 \times 10^{-6}|}{0{,}687925 \times 10^{-6}} = 45{,}36\%. \tag{2.91}$$

$\Diamond$

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

$$x^2 + 300x - 0{,}014 = 0, \tag{2.92}$$

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:

$$\begin{aligned}
\Delta &= b^2 - 4 \cdot a \cdot c && (2.93)\\
&= 0{,}300000 \times 10^{3} \times 0{,}300000 \times 10^{3} && (2.94)\\
&+ 0{,}400000 \times 10^{1} \times 0{,}100000 \times 10^{1} \times 0{,}140000 \times 10^{-1} && (2.95)\\
&= 0{,}900000 \times 10^{5} + 0{,}560000 \times 10^{-1} && (2.96)\\
&= 0{,}900001 \times 10^{5} && (2.97)
\end{aligned}$$

e as raízes:

$$x_1,x_2 = \frac{-0{,}300000 \times 10^3 \pm \sqrt{\Delta}}{0{,}200000 \times 10^1} \tag{2.98}$$

$$= \frac{-0{,}300000 \times 10^3 \pm \sqrt{0{,}900001 \times 10^5}}{0{,}200000 \times 10^1} \tag{2.99}$$

$$= \frac{-0{,}300000 \times 10^3 \pm 0{,}300000 \times 10^3}{0{,}200000 \times 10^1} \tag{2.100}$$

$$\tag{2.101}$$

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

$$\begin{aligned} \tilde{x}_1 &= \frac{-0{,}300000 \times 10^3 - 0{,}300000 \times 10^3}{0{,}200000 \times 10^1} \\ &= -\frac{0{,}600000 \times 10^3}{0{,}200000 \times 10^1} = -0{,}300000 \times 10^3 \end{aligned} \tag{2.102}$$

e

$$\tilde{x}_2 = \frac{-0{,}300000 \times 10^3 + 0{,}300000 \times 10^3}{0{,}200000 \times 10^1} = 0{,}000000 \times 10^0 \tag{2.103}$$

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

$$x_1 = -0{,}300000 \times 10^3 \quad \text{e} \quad x_2 = 0{,}466667 \times 10^{-4}. \tag{2.104}$$

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

$$-b + \sqrt{b^2 - 4ac} \tag{2.105}$$

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:

$$\sqrt{1-x} = 1 - \frac{1}{2}x + O(x^2). \tag{2.106}$$

Substituindo esta aproximação na fórmula de Bhaskara, temos:

$$\begin{aligned} x &= \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} && (2.107)\\ &= \frac{-b \pm b\sqrt{1 - \frac{4ac}{b^2}}}{2a} && (2.108)\\ &\approx \frac{-b \pm b\left(1 - \frac{4ac}{2b^2}\right)}{2a} && (2.109)\\ & && (2.110) \end{aligned}$$

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

$$\begin{aligned} \tilde{x}_1 &= \frac{-b - b + \frac{4ac}{2b}}{2a} = -\frac{b}{a} + \frac{c}{b} && (2.111)\\ &= -\frac{0{,}300000 \times 10^3}{0{,}100000 \times 10^1} - \frac{0{,}140000 \times 10^{-1}}{0{,}300000 \times 10^3} && (2.112)\\ &= -0{,}300000 \times 10^3 - 0{,}466667 \times 10^{-4} && (2.113)\\ &= -0{,}300000 \times 10^3 && (2.114) \end{aligned}$$

$$\begin{aligned} \tilde{x}_2 &= \frac{-b + b - \frac{4ac}{2b}}{2a} && (2.115)\\ &= -\frac{4ac}{4ab} && (2.116)\\ &= -\frac{c}{b} = -\frac{-0{,}140000 \times 10^{-1}}{0{,}300000 \times 10^3} = 0{,}466667 \times 10^{-4} && (2.117)\\ & && (2.118) \end{aligned}$$

Observe que o efeito catastrófico foi eliminado.

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

$$\begin{aligned} x_2 &= \frac{-b + \sqrt{b^2 - 4ac}}{2a} = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \cdot \frac{-b - \sqrt{b^2 - 4ac}}{-b - \sqrt{b^2 - 4ac}} && (2.119)\\ &= \frac{b^2 - (b^2 - 4ac)}{2a\left(-b - \sqrt{b^2 - 4ac}\right)} = \frac{4ac}{2a\left(-b - \sqrt{b^2 - 4ac}\right)} && (2.120)\\ &= -\frac{2c}{\left(b + \sqrt{b^2 - 4ac}\right)} && (2.121) \end{aligned}$$

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

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

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

$$\Delta y = f(x + \Delta x) - f(x) \approx f'(x)\Delta x \tag{2.123}$$

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

$$\frac{\Delta y}{y} \approx \frac{f'(x)}{f(x)} \frac{x\Delta x}{x} \tag{2.124}$$

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*

$$\kappa_f(x) := \left| \frac{xf'(x)}{f(x)} \right| \tag{2.125}$$

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

$$\delta_f = |f(x_1,x_2,...,x_n) - f(\bar{x}_1,\bar{x}_2,...,\bar{x}_n)| \approx \sum_{i=1}^{n} \left|\frac{\partial f}{\partial x_i}(x_1,x_2,...,x_n)\right| \delta_{x_i} \tag{2.126}$$

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

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

$$\left|\frac{\Delta x}{x}\right| = \left|\frac{2 - 1{,}999}{2}\right| = 0{,}0005 \tag{2.127}$$

O número de condicionamento do problema calcular a raiz é

$$\kappa_f(x) := \left|\frac{xf'(x)}{f(x)}\right| = \left|\frac{x\frac{1}{2\sqrt{x}}}{\sqrt{x}}\right| = \frac{1}{2} \tag{2.128}$$

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

$$\frac{\Delta y}{y} = \frac{\sqrt{2} - \sqrt{1{,}999}}{\sqrt{2}} \approx 0{,}000250031... \tag{2.129}$$

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

$$y^* = \frac{10}{1 - (0{,}9995)^2} \approx 10002{,}500625157739705173 \tag{2.130}$$

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

$$\kappa_f(x) := \left|\frac{xf'(x)}{f(x)}\right| = \left|\frac{2x^2}{1 - x^2}\right| \tag{2.131}$$

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

$$\kappa_f(0{,}9995) \approx 1998{,}5 \tag{2.132}$$

Como o erro relativo na entrada é

$$\left|\frac{\Delta x}{x}\right| = \left|\frac{0{,}0001}{0{,}9995}\right| \approx 0{,}00010005... \tag{2.133}$$

temos que o erro na saída será aproximadamente

$$\left|\frac{\Delta y}{y}\right| \approx \kappa_f(x) \left|\frac{\Delta x}{x}\right| \approx 1998{,}5 \times 0{,}00010005... \approx 0{,}1999 \tag{2.134}$$

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

$$y_1^* \approx 8335{,}83 \tag{2.135}$$
$$y_2^* \approx 12520{,}50 \tag{2.136}$$

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:

$$\delta_f = |f'(x)|\delta_x. \tag{2.137}$$

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

$$\delta_f = |(1+x)e^x| \cdot \delta_x \tag{2.138}$$
$$= |3e^2| \cdot 0{,}05 = 1{,}1084. \tag{2.139}$$

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

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:

$$\frac{\partial f}{\partial x} = \frac{2x^3 - (2x^3 + 2x)}{x^4}e^{2y} = -\frac{2e^{2y}}{x^3} \tag{2.140}$$

e

$$\frac{\partial f}{\partial y} = 2\frac{x^2+1}{x^2}e^{2y} \tag{2.141}$$

Calculamos o erro absoluto em termos do erro relativo:

$$\frac{\delta_x}{|x|} = 0{,}1 \Rightarrow \delta_x = 3 \cdot 0{,}1 = 0{,}3 \tag{2.142}$$

$$\frac{\delta_y}{|y|} = 0{,}03 \Rightarrow \delta_y = 2 \cdot 0{,}03 = 0{,}06 \tag{2.143}$$

Aplicando a expressão para estimar o erro em f temos

$$\delta_f = \left|\frac{\partial f}{\partial x}\right| \delta_x + \left|\frac{\partial f}{\partial y}\right| \delta_y \tag{2.144}$$

$$= \frac{2e^4}{27} \cdot 0{,}3 + 2\frac{9+1}{9} e^4 \cdot 0{,}06 = 8{,}493045557 \tag{2.145}$$

Portanto, o erro relativo ao calcular f é estimado por

$$\frac{\delta f}{|f|} = \frac{8{,}493045557}{\frac{9+1}{9} e^4} = 14\% \tag{2.146}$$

$\Diamond$

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

$$A = \frac{bc}{2} = \frac{b\sqrt{a^2 - b^2}}{2}. \tag{2.147}$$

Agora calculamos as derivadas

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

$$\frac{\partial A}{\partial b} = \frac{\sqrt{a^2 - b^2}}{2} - \frac{b^2}{2\sqrt{a^2 - b^2}}, \tag{2.149}$$

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

$$\delta_A \approx \left|\frac{\partial A}{\partial a}\right| \delta_a + \left|\frac{\partial A}{\partial b}\right| \delta_b \tag{2.150}$$

$$\approx \frac{3\sqrt{5}}{5} \cdot 0{,}01 + \frac{\sqrt{5}}{10} \cdot 0{,}01 = 0{,}01565247584 \tag{2.151}$$

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

$$\frac{0{,}01565247584}{\frac{2\sqrt{3^2-2^2}}{2}} = 0{,}7\% \tag{2.152}$$

$\Diamond$

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:

$$z = \frac{y^4}{1+y^4} e^x. \tag{2.153}$$

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 ampéres e a tensão V em volts em uma lâmpada se relacionam conforme a seguinte expressão:

$$I = \left(\frac{V}{V_0}\right)^{\alpha}, \tag{2.154}$$

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:

$$x^{(1)} = \frac{1}{3} \tag{2.155}$$

$$x^{(n+1)} = 4x^{(n)} - 1, \quad n = 1, 2, \ldots \tag{2.156}$$

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 no `GNU Octave`, colocando:

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

e itere algumas vezes a linha de comando:

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

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:

$$\tilde{x}^{(1)} = \frac{1}{3} + \epsilon \tag{2.157}$$

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

$$\tilde{x}^{(3)} = 4x^{(2)} - 1 = 4\left(\frac{1}{3} + 4\epsilon\right) - 1 = \frac{1}{3} + 4^2\epsilon \tag{2.159}$$

$$\vdots \tag{2.160}$$

$$\tilde{x}^{(n)} = \frac{1}{3} + 4^{(n-1)}\epsilon \tag{2.161}$$

Portanto o limite da sequência diverge,

$$\lim_{x \to \infty} |\tilde{x}^{(n)}| = \infty \tag{2.162}$$

Qual o número de condicionamento desse problema?

Exemplo 2.9.2. Observe a seguinte identidade

$$f(x) = \frac{(1+x) - 1}{x} = 1 \tag{2.163}$$

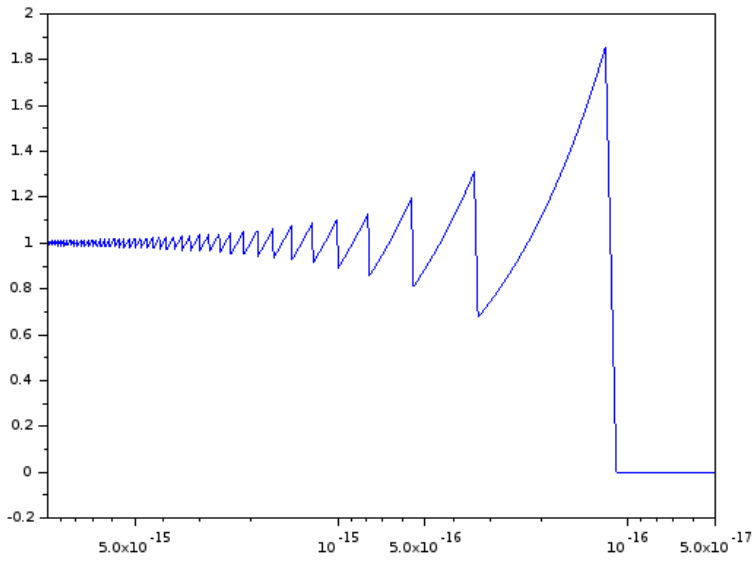


Figure 2.1: Gráfico na função do Exemplo **??**.

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 **??** 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

$$\left(1+\frac{1}{n}\right)^n \tag{2.164}$$

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 (**??**) quando n tende a infinito é o número de Euler:

$$\lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n = e = 2{,}718281828459... \tag{2.165}$$

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

$$\left(1+\frac{1}{1}\right)^1 < \left(1+\frac{1}{2}\right)^2 < \left(1+\frac{1}{3}\right)^3 < \cdots \tag{2.166}$$

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

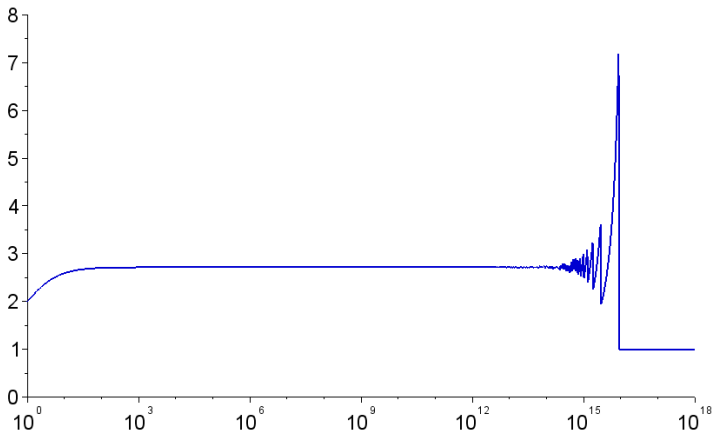


Figure 2.2: Gráfico de $\left(1+\frac{1}{n}\right)^n$ em função de n em escala linear-logarítmica variando de 10^0 até 10^{18}. Veja o Exemplo **??**.

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 **??**.

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:

$$n \ \rightarrow \ 1/n \ \rightarrow \ 1 + 1/n \ \rightarrow \ (1 + 1/n)^n \tag{2.167}$$

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 $2{,}22 \times 10^{-16}$.

No `GNU Octave`, o épsilon de máquina é a constante `eps`. Observe que:

```
>> printf('%1.25f\n', 1+eps)
1.0000000000000002220446049
```

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 (**??**), 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:

$$n \ \rightarrow \ 1/n \ \rightarrow \ 1/(1/n) \tag{2.168}$$

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

$$n \ \rightarrow \ 1 + n \ \rightarrow \ (1 + n) - 1. \tag{2.169}$$

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 `GNU Octave` quando $n = 7 \times 10^{13}$:

```
>> format('long')
>> n=7e13
n =  70000000000000
>> 1/n
ans =    1.42857142857143e-14
>> y=1+1/n
y =  1.00000000000001
```

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:

```
>> y-1
ans =    1.42108547152020e-14
>> 1/(y-1)
ans =  70368744177664
```

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:

$$v_1 = 270[A] + 30[B], \quad v_2 = 140[A] + 20[B] \text{ e } v_3 = 15[A] + 200[B] \tag{2.170}$$

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

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{2.171}$$

é inversível e sua inversa é dada por

$$A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}. \tag{2.172}$$

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 $[A] = [B] = 10milimol/l$. Dica: Você deve diferenciar as grandezas $[A]$ e $[B]$ em relação aos valores das tensões.

Solução. Em ambos casos, temos a seguinte estrutura:

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

De forma que

$$\begin{bmatrix} [A] \\ [B] \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} \\ S_{21} & S_{22} \end{bmatrix}^{-1} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \frac{1}{S_{11}S_{22} - S_{12}S_{21}} \begin{bmatrix} S_{22} & -S_{12} \\ -S_{21} & S_{11} \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \tag{2.174}$$

Portanto

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

$$[B] = \frac{-S_{21}v_1 + S_{11}v_2}{S_{11}S_{22} - S_{12}S_{21}} \tag{2.176}$$

Usando derivação logarítmica, temos

$$\frac{1}{[A]}\frac{\partial[A]}{\partial S_{11}} = -\frac{S_{22}}{S_{11}S_{22}-S_{12}S_{21}} \tag{2.177}$$

$$\frac{1}{[A]}\frac{\partial[A]}{\partial S_{12}} = -\frac{v_2}{S_{22}v_1-S_{12}v_2}+\frac{S_{21}}{S_{11}S_{22}-S_{12}S_{21}} \tag{2.178}$$

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

$$\frac{1}{[A]}\frac{\partial[A]}{\partial S_{21}} = \frac{S_{12}}{S_{11}S_{22}-S_{12}S_{21}} \tag{2.180}$$

$$\frac{1}{[A]}\frac{\partial[A]}{\partial S_{22}} = \frac{v_1}{S_{22}v_1-S_{12}v_2}-\frac{S_{11}}{S_{11}S_{22}-S_{12}S_{21}} \tag{2.181}$$

$$= \frac{[A]}{[B]}\cdot\frac{S_{12}}{S_{11}S_{22}-S_{12}S_{21}} \tag{2.182}$$

e

$$\frac{1}{[B]}\frac{\partial[B]}{\partial S_{11}} = \frac{v_2}{-S_{21}v_1+S_{11}v_2}-\frac{S_{22}}{S_{11}S_{22}-S_{12}S_{21}} \tag{2.183}$$

$$= \frac{[B]}{[A]}\frac{S_{21}}{S_{11}S_{22}-S_{12}S_{21}} \tag{2.184}$$

$$\frac{1}{[B]}\frac{\partial[B]}{\partial S_{12}} = \frac{S_{21}}{S_{11}S_{22}-S_{12}S_{21}} \tag{2.185}$$

$$\frac{1}{[B]}\frac{\partial[B]}{\partial S_{21}} = -\frac{v_1}{-S_{21}v_1+S_{11}v_2}+\frac{S_{21}}{S_{11}S_{22}-S_{12}S_{21}} \tag{2.186}$$

$$= -\frac{[B]}{[A]}\frac{S_{11}}{S_{11}S_{22}-S_{12}S_{21}} \tag{2.187}$$

$$\frac{1}{[B]}\frac{\partial[B]}{\partial S_{22}} = -\frac{S_{11}}{S_{11}S_{22}-S_{12}S_{21}} \tag{2.188}$$

$$\tag{2.189}$$

E o erro associado às medidas pode ser aproximado por

$$\frac{1}{[A]}\delta_{[A]} = \left|\frac{1}{[A]}\frac{\partial[A]}{\partial S_{11}}\right|\delta_{S_{11}}+\left|\frac{1}{[A]}\frac{\partial[A]}{\partial S_{12}}\right|\delta_{S_{12}} \tag{2.190}$$

$$+ \left|\frac{1}{[A]}\frac{\partial[A]}{\partial S_{21}}\right|\delta_{S_{21}}+\left|\frac{1}{[A]}\frac{\partial[A]}{\partial S_{22}}\right|\delta_{S_{22}} \tag{2.191}$$

$$= \frac{1}{|\det S|}\left[S_{22}\delta_{S_{11}}+\frac{[A]}{[B]}S_{22}\delta_{S_{12}}+S_{12}\delta_{S_{21}}+\frac{[A]}{[B]}S_{12}\delta_{S_{22}}\right] \tag{2.192}$$

Analogamente, temos:

$$\frac{1}{[B]}\delta_{[B]} = \frac{1}{|\det S|}\left[\frac{[B]}{[A]}S_{21}\delta_{S_{11}} + S_{21}\delta_{S_{11}} + \frac{[B]}{[A]}S_{11}\delta_{S_{21}} + S_{11}\delta_{S_{22}}\right] \quad (2.193)$$

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

Fazemos agora a aplicação numérica:

Caso do par 1-2:

$$\det S = \begin{vmatrix} 270 & 30 \\ 140 & 20 \end{vmatrix} = 1200 \quad (2.194)$$

$$\begin{aligned} \frac{1}{[A]}\delta_{[A]} &= \frac{1}{1200}\left[20 \times 270 \times 2\% + 20 \times 30 \times 2\%\right. && (2.195)\\ &\qquad \left. + 30 \times 140 \times 2\% + 30 \times 20 \times 2\%\right] && (2.196)\\ &= \frac{216}{1200} = 0.18 = 18\% && (2.197)\\ \frac{1}{[B]}\delta_{[B]} &= \frac{1}{1200}\left[140 \times 270 \times 2\% + 140 \times 30 \times 2\%\right. && (2.198)\\ &\qquad \left. + 270 \times 140 \times 2\% + 270 \times 20 \times 2\%\right] && (2.199)\\ &= \frac{426}{1200} = 0.355 = 35.5\% && (2.200) \end{aligned}$$

Caso do par 1-3:

$$\det S = \begin{vmatrix} 270 & 30 \\ 15 & 200 \end{vmatrix} = 53550 \quad (2.201)$$

$$\begin{aligned} \frac{1}{[A]}\delta_{[A]} &= \frac{1}{53550}\left[200 \times 270 \times 2\% + 200 \times 30 \times 2\%\right. && (2.202)\\ &\qquad \left. + 30 \times 15 \times 10\% + 30 \times 200 \times 10\%\right] && (2.203)\\ &= \frac{1804{,}6}{52550} \approx 0.0337 = 3.37\% && (2.204)\\ \frac{1}{[B]}\delta_{[B]} &= \frac{1}{53550}\left[15 \times 270 \times 2\% + 15 \times 30 \times 2\%\right. && (2.205)\\ &\qquad \left. + 270 \times 15 \times 10\% + 270 \times 200 \times 10\%\right] && (2.206)\\ &= \frac{5895}{53550} \approx 0.11 = 11\% && (2.207) \end{aligned}$$

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.

$\Diamond$

Exercícios

E 2.9.1. Considere as expressões:

$$\frac{\exp(1/\mu)}{1+\exp(1/\mu)} \tag{2.208}$$

e

$$\frac{1}{\exp(-1/\mu)+1} \tag{2.209}$$

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:

$$\frac{1-\cos(x)}{x^2} = \frac{1}{2}\left(\frac{\operatorname{sen}(x/2)}{x/2}\right)^2. \tag{2.210}$$

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:

$$\sqrt{e^{2x}+1} - e^x \qquad \text{e} \qquad \sqrt{e^{2x}+x^2} - e^x \tag{2.211}$$

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:

$$E = mc^2 \left(\frac{1}{\sqrt{1 - (v/c)^2}} - 1 \right), \tag{2.215}$$

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}?

Chapter 3

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)

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2 \\ &\vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n &= b_m \end{aligned} \tag{3.1}$$

onde m é o número de equações e n é o número de incógnitas. Este sistema pode ser escrito na **forma matricial**

$$Ax = b \tag{3.2}$$

onde:

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}, x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \text{ e } b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}, \tag{3.3}$$

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,

$$[A|b] = \left[\begin{array}{cccc|c} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} & b_m \end{array}\right]. \tag{3.4}$$

Salvo especificado ao contrário, assumiremos ao longo deste capítulo que a matriz dos coeficientes A é uma matriz real não singular (isto é, invertível).

Exemplo 3.0.1. Consideramos o seguinte sistema linear

$$\begin{aligned} x + y + z &= 1 \\ 4x + 4y + 2z &= 2 \\ 2x + y - z &= 0. \end{aligned} \tag{3.5}$$

Na sua forma matricial, este sistema é escrito como

$$Ax = b \Leftrightarrow \underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 4 & 4 & 2 \\ 2 & 1 & -1 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x \\ y \\ z \end{bmatrix}}_{\overline{x}} = \underbrace{\begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix}}_{b}. \tag{3.6}$$

A matriz estendida do sistema acima é

$$E := [A|b] = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 4 & 4 & 2 & 2 \\ 2 & 1 & -1 & 0 \end{bmatrix}. \tag{3.7}$$

3.1 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 trapezoidal (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 3.1.1. Resolva o sistema

$$\begin{aligned} x + y + z &= 1 \\ 4x + 4y + 2z &= 2 \\ 2x + y - z &= 0 \end{aligned} \tag{3.8}$$

pelo método de eliminação gaussiana.

Solução. A matriz estendida do sistema é escrita como

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 4 & 4 & 2 & 2 \\ 2 & 1 & -1 & 0 \end{bmatrix} \tag{3.9}$$

No primeiro passo, subtraímos da segunda linha o quádruplo da primeira e subtraímos da terceira linha o dobro da primeira linha:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & -2 & -2 \\ 0 & -1 & -3 & -2 \end{bmatrix} \tag{3.10}$$

No segundo passo, permutamos a segunda linha com a terceira:

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -1 & -3 & -2 \\ 0 & 0 & -2 & -2 \end{bmatrix} \tag{3.11}$$

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 **??**, 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 3.1.2. No Exemplo **??**, usamos o procedimento de eliminação gaussiana e obtivemos

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 4 & 4 & 2 & 2 \\ 2 & 1 & -1 & 0 \end{bmatrix}}_{\text{matriz estendida}} \sim \underbrace{\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -1 & -3 & -2 \\ 0 & 0 & -2 & -2 \end{bmatrix}}_{\text{matriz escalonada}}. \tag{3.12}$$

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

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & -1 & -3 & -2 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{3.13}$$

Então, somando da segunda linha o triplo da terceira e subtraindo da primeira a terceira linha, obtemos

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{3.14}$$

Fixamos, agora, na segunda linha. Dividimos esta linha pelo valor do elemento

em sua diagonal, isto nos fornece

$$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{3.15}$$

Por fim, subtraímos da primeira linha a segunda, obtendo a matriz escalonada reduzida

$$\begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{3.16}$$

Desta matriz escalonada reduzida temos, imediatamente, $x = 1$, $y = -1$ e $z = 1$, como no Exemplo **??**.

3.1.1 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 3.1.3. Resolva o sistema

$$\begin{aligned} x + y + z &= 1 \\ 2x + y - z &= 0 \\ 2x + 2y + z &= 1 \end{aligned} \tag{3.17}$$

por eliminação gaussiana com pivotamento parcial.

Prof. M.e Daniel Cassimiro

Solução. A matriz estendida do sistema é

$$\begin{aligned}
\begin{bmatrix} 1 & 1 & 1 & 1 \\ 2 & 1 & -1 & 0 \\ 2 & 2 & 1 & 1 \end{bmatrix} &\sim \begin{bmatrix} 2 & 1 & -1 & 0 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 1 & 1 \end{bmatrix} \\
&\sim \begin{bmatrix} 2 & 1 & -1 & 0 \\ 0 & 1/2 & 3/2 & 1 \\ 0 & 1 & 2 & 1 \end{bmatrix} \\
&\sim \begin{bmatrix} 2 & 1 & -1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 1/2 & 3/2 & 1 \end{bmatrix} \\
&\sim \begin{bmatrix} 2 & 1 & -1 & 0 \\ 0 & 1 & 2 & 1 \\ 0 & 0 & 1/2 & 1/2 \end{bmatrix}
\end{aligned} \tag{3.18}$$

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 $x = 1$.

No `Scilab`, podemos fazer estas computações da seguinte forma:

```
E = [1 1  1 1; 2 1 -1 0;2 2  1 1]
disp(E)

//L2 <-> L1
aux = E(2,:)
E(2,:) = E(1,:)
E(1,:) = aux
disp(E)

//zera E(2:3,1)
E(2:3,:) = E(2:3,:) - (E(2:3,1)/E(1,1))*E(1,:)
disp(E)

//zera E(3,2)
E(3,:) = E(3,:) - (E(3,2)/E(2,2))*E(2,:)
```

```
disp(E)

//subs regressiva
x = zeros(3,1)
x(3) = E(3,4)/E(3,3)
x(2) = (E(2,4) - E(2,3)*x(3))/E(2,2)
x(1) = (E(1,4) - E(1,3)*x(3) - E(1,2)*x(2))/E(1,1)
disp(x)
```

$\Diamond$

A técnica de eliminação gaussiana com pivotamento parcial ajuda a evitar a propagação dos erros de arredondamento. Vejamos o próximo exemplo.

Exemplo 3.1.4 (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 à aritmética de ponto flutuante quando $0 < |\epsilon| \ll 1$.

$$\begin{bmatrix} \varepsilon & 2 \\ 1 & \varepsilon \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 4 \\ 3 \end{bmatrix} \tag{3.19}$$

Solução. Vamos, primeiramente, executar a eliminação gaussiana sem pivotamento parcial para $\varepsilon \neq 0$ e $|\varepsilon| \ll 1$:

$$\left[\begin{array}{cc|c} \varepsilon & 2 & 4 \\ 1 & \varepsilon & 3 \end{array}\right] \sim \left[\begin{array}{cc|c} \varepsilon & 2 & 4 \\ 0 & \varepsilon - \frac{2}{\varepsilon} & 3 - \frac{4}{\varepsilon} \end{array}\right] \tag{3.20}$$

Temos

$$y = \frac{3 - 4/\varepsilon}{\varepsilon - 2/\varepsilon} \tag{3.21}$$

e

$$x = \frac{4 - 2y}{\varepsilon} \tag{3.22}$$

Observe que a expressão obtida para y se aproximada de 2 quando ε é pequeno:

$$y = \frac{3 - 4/\varepsilon}{\varepsilon - 2/\varepsilon} = \frac{3\varepsilon - 4}{\varepsilon^2 - 2} \longrightarrow \frac{-4}{-2} = 2, \quad \text{quando } \varepsilon \to 0. \tag{3.23}$$

Já expressão obtida para x depende justamente da diferença $2 - y$:

$$x = \frac{4 - 2y}{\varepsilon} = \frac{2}{\varepsilon}(2 - y) \tag{3.24}$$

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:

$$\left[\begin{array}{cc|c} \varepsilon & 2 & 4 \\ 1 & \varepsilon & 3 \end{array}\right] \sim \left[\begin{array}{cc|c} 1 & \varepsilon & 3 \\ \varepsilon & 2 & 4 \end{array}\right] \sim \left[\begin{array}{cc|c} 1 & \varepsilon & 3 \\ 0 & 2-\varepsilon^2 & 4-3\varepsilon \end{array}\right] \tag{3.25}$$

Continuando o procedimento, temos:

$$y = \frac{4-3\varepsilon}{2-\varepsilon^2} \tag{3.26}$$

e

$$x = 3 - \varepsilon y \tag{3.27}$$

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. $\Diamond$

Exercícios resolvidos

ER 3.1.1. Resolva o seguinte sistema por eliminação gaussiana com pivotamento parcial.

$$\begin{aligned} 2y + 2z &= 8 \\ x + 2y + z &= 9 \\ x + y + z &= 6 \end{aligned} \tag{3.28}$$

Solução. A forma matricial do sistema dado é

$$\left[\begin{array}{ccc} 0 & 2 & 2 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{array}\right] \left[\begin{array}{c} x \\ y \\ z \end{array}\right] = \left[\begin{array}{c} 8 \\ 9 \\ 6 \end{array}\right] \tag{3.29}$$

Construímos, então, a matriz completa e seguimos com o procedimento de eliminação gaussiana com pivotamento parcial:

$$\left[\begin{array}{ccc|c} 0 & 2 & 2 & 8 \\ 1 & 2 & 1 & 9 \\ 1 & 1 & 1 & 6 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 2 & 1 & 9 \\ 0 & 2 & 2 & 8 \\ 1 & 1 & 1 & 6 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 2 & 1 & 9 \\ 0 & 2 & 2 & 8 \\ 0 & -1 & 0 & -3 \end{array}\right] \tag{3.30}$$

$$\sim \left[\begin{array}{ccc|c} 1 & 2 & 1 & 9 \\ 0 & 2 & 2 & 8 \\ 0 & 0 & 1 & 1 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 2 & 0 & 8 \\ 0 & 2 & 0 & 6 \\ 0 & 0 & 1 & 1 \end{array}\right] \tag{3.31}$$

$$\sim \left[\begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 2 & 0 & 6 \\ 0 & 0 & 1 & 1 \end{array}\right] \tag{3.32}$$

Portanto $x = 2$, $y = 3$ e $z = 1$.

$\Diamond$

Exercícios

E 3.1.1. Resolva o seguinte sistema de equações lineares

$$\begin{aligned} x + y + z &= 0 \\ x + 10z &= -48 \\ 10y + z &= 25 \end{aligned} \tag{3.33}$$

Usando eliminação gaussiana com pivotamento parcial (não use o computador para resolver essa questão).

E 3.1.2. Resolva o seguinte sistema de equações lineares

$$x + y + z = 0 \tag{3.38}$$
$$x + 10z = -48 \tag{3.39}$$
$$10y + z = 25 \tag{3.40}$$

Usando eliminação gaussiana com pivotamento parcial (não use o computador para resolver essa questão).

E 3.1.3. Calcule a inversa da matriz

$$A = \begin{bmatrix} 1 & 2 & -1 \\ -1 & 2 & 0 \\ 2 & 1 & -1 \end{bmatrix} \tag{3.41}$$

usando eliminação gaussiana com pivotamento parcial.

E 3.1.4. Demonstre que se $ad \neq bc$, então a matriz A dada por:

$$A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \tag{3.42}$$

é inversível e sua inversa é dada por:

$$A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}. \tag{3.43}$$

E 3.1.5. Considere as matrizes

$$A = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \tag{3.44}$$

e

$$E = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.45}$$

e o vetor

$$v = \begin{bmatrix} 2 \\ 3 \\ 4 \end{bmatrix} \tag{3.46}$$

a) Resolva o sistema $Ax = v$ sem usar o computador.

b) Sem usar o computador e através da técnica algébrica de sua preferência, resolva o sistema $(A + \varepsilon E)x_\varepsilon = v$ considerando $|\varepsilon| \ll 1$ e obtenha a solução exata em função do parâmetro ε.

c) Usando a expressão analítica obtida acima, calcule o limite $\lim_{\varepsilon \to 0} x_\varepsilon$.

d) Resolva o sistema $(A+\varepsilon E)x = v$ no computador usando pivotamento parcial e depois sem usar pivotamento parcial para valores muito pequenos de ε como $10^{-10}, 10^{-15}, \ldots$. O que você observa?

E 3.1.6. Resolva o seguinte sistema de 5 equações lineares

$$\begin{aligned} x_1 - x_2 &= 0 && (3.51)\\ -x_{i-1} + 2.5x_i - x_{i+1} &= e^{-\frac{(i-3)^2}{20}}, \qquad 2 \leq i \leq 4 && (3.52)\\ 2x_5 - x_4 &= 0 && (3.53) \end{aligned}$$

representando-o como um problema do tipo $Ax = b$ no computador e usando um método pronto para resolvê-lo. Repita usando a rotina que implementa eliminação gaussiana.

E 3.1.7. Encontre a inversa da matriz

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -1 & 2 \\ 1 & 1 & 4 \end{bmatrix} \tag{3.54}$$

a) Usando eliminação gaussiana com pivotamento parcial à mão.

b) Usando a rotina de eliminação gaussiana que você implementou.

c) Usando a rotina pronta para inversão de matrizes na plataforma em uso.

3.2 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 grandes, isto é, o custo quando estas dimensões crescem infinitamente.

Exemplo 3.2.1 (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

$$a\boldsymbol{x} = [a \times x_1, a \times x_2, ..., a \times x_n] \tag{3.56}$$

usando n multiplicações, ou seja, um custo computacional, C, de

$$C = n \text{ flops.} \tag{3.57}$$

$\Diamond$

Exemplo 3.2.2 (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

$$\boldsymbol{x} \cdot \boldsymbol{y} = x_1 \times y_1 + x_2 \times y_2 + ... + x_n \times y_n \tag{3.58}$$

São realizadas n multiplicações (cada produto x_i por y_i) e $n-1$ somas, ou seja, o custo total de operações é de

$$C := (n) + (n-1) = 2n - 1 \text{ flops} \tag{3.59}$$

$\Diamond$

Exemplo 3.2.3 (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

$$\begin{bmatrix} a_{11} & a_{12} & \cdots a_{1n} \\ \vdots & & \vdots \\ a_{n1} & & \cdots a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} a_{11} \times x_1 + a_{12}x_2 + ... + a_{1n} \times x_n \\ \vdots \\ a_{n1} \times x_1 + a_{n2}x_2 + ... + a_{nn} \times x_n \end{bmatrix} \tag{3.60}$$

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]

$$C := n \cdot (2n - 1) = 2n^2 - n \text{ flops.} \tag{3.62}$$

À medida que $n \to \infty$, temos

$$\mathcal{O}(2n^2 - n) = \mathcal{O}(2n^2) = \mathcal{O}(n^2) \text{ flops.} \tag{3.63}$$

$\Diamond$

Exemplo 3.2.4 (Produto matriz-matriz)**.** Qual o custo para calcular o produto de duas matrizes A e B?

[1]Contando apenas multiplicações/divisões obtemos

$$n \cdot \mathcal{O}(n) = \mathcal{O}(n^2) \text{ flops.} \tag{3.61}$$

Solução. Sejam $A, B \in \mathbb{R}^{n\times n}$ temos que

$$\begin{bmatrix} a_{11} & a_{12} & \cdots a_{1n} \\ \vdots & & \vdots \\ a_{n1} & & \cdots a_{nn} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} & \cdots b_{1n} \\ \vdots & & \vdots \\ b_{n1} & & \cdots b_{nn} \end{bmatrix} = \begin{bmatrix} d_{11} & d_{12} & \cdots d_{1n} \\ \vdots & & \vdots \\ d_{n1} & & \cdots d_{nn} \end{bmatrix} \tag{3.64}$$

onde o elemento d_{ij} é o produto da linha i de A pela coluna j de B,

$$d_{ij} = a_{i1} \times b_{1j} + a_{i2} \times b_{2j} + ... + a_{in} \times b_{nj} \tag{3.65}$$

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]

$$C = n \times n \times (2n-1) = 2n^3 - n^2 \text{ flops.} \tag{3.67}$$

$\Diamond$

3.3 Sistemas triangulares

Considere um sistema linear onde a matriz é triangular superior, ou seja,

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ 0 & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \dots & 0 & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix} \tag{3.68}$$

tal que todos elementos abaixo da diagonal são iguais a zero.

Podemos resolver esse sistema iniciando pela última equação e isolando x_n, obtendo

$$x_n = b_n / a_{nn} \tag{3.69}$$

Substituindo x_n na penúltima equação

$$a_{n-1,n-1}x_{n-1} + a_{n-1,n}x_n = b_{n-1} \tag{3.70}$$

[2]Contando apenas $\times$ e / obtemos

$$n \times n \times (n) = n^3 \text{ flops.} \tag{3.66}$$

e isolando x_{n-1} obtemos

$$x_{n-1} = (b_{n-1} - a_{n-1,n}x_n)/a_{n-1,n-1}. \tag{3.71}$$

Continuando desta forma até a primeira equação, obteremos

$$x_1 = (b_1 - a_{12}x_2 \cdots - a_{1n}x_n)/a_{11}. \tag{3.72}$$

De forma geral, temos que

$$x_i = (b_i - a_{i,i+1}x_{i+1} \cdots - a_{i,n}x_n)/a_{i,i}, \quad i = 2, \ldots, n. \tag{3.73}$$

3.3.1 Código Scilab: resolução de um sistema triangular superior

Para resolver um sistema triangular superior, iniciamos da última linha em direção a primeira.

```
function [x]=solveU(U,b) // U:= matriz triangular superior
   n=size(U,1) // b:= vetor
   x(n)=b(n)/U(n,n)
   for i=n-1:-1:1
        x(i)=(b(i)-U(i,i+1:n)*x(i+1:n) )/U(i,i)
    end
endfunction
```

./cap_linsis/codes/scilab/solveU.sci

3.3.2 Código Scilab: resolução de um sistema triangular inferior

Para resolver um sistema triangular inferior, podemos fazer o processo inverso, iniciando da primeira equação.

```
function [x]=solveL(L,b) // L: matriz triangular inferior
    n=size(L,1) // b:  vetor
    x(1)=b(1)/L(1,1)
    for i=2:n
        x(i)=(b(i)-L(i,1:i-1)*x(1:i-1) )/L(i,i)
    end
endfunction
```

./cap_linsis/codes/scilab/solveL.sci

Custo computacional

Vamos contar o número total de flops para resolver um sistema triangular inferior. Note que o custo para um sistema triangular superior será o mesmo.

Na linha 3, temos uma divisão, portanto 1 flop.

Na linha 5 quando $i = 2$, temos

`x(2)=(b(2)-L(2,1:1)*x(1:1))/L(2,2)`,

ou seja, 1 subtração+1 multiplicação + 1 divisão = 3 flops.

Quando $i = 3$,

`x(3)=(b(3)-L(3,1:2)*x(1:2))/L(3,3)`

temos 1 subtração+(2 multiplicações + 1 soma) +1 divisão = 5 flops.

Quando $i = 4$, temos 1 subtração+(3 multiplicações + 2 somas) +1 divisão = 7 flops.

Até que para $i = n$, temos

`x(n)=(b(n)-L(n,1:n-1)*x(1:n-1))/L(n,n)`,

com 1 subtração+($n-1$ multiplicações + $n-2$ somas) + 1 divisão, ou seja, $1 + (n - 1 + n - 2) + 1 = 2n - 1$ flops.

Somando todos esses custos[3] temos que o custo para resolver um sistema triangular inferior é

$$1 + 3 + 5 + 7 + ... + 2n - 1 = \sum_{k=1}^{n}(2k - 1) = 2\sum_{k=1}^{n} k - \sum_{k=1}^{n} 1 \tag{3.75}$$

e utilizando que a soma dos k inteiros é a soma dos termos de uma progressão aritmética[4]

$$2(n(n + 1)/2) - n = n^2 \text{ flops.} \tag{3.76}$$

3.4 Fatoração LU

Considere um sistema linear $Ax = b$, onde a matriz A é densa[5]. 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, $A = LU$.

[3] Contando apenas multiplicações/divisões obtemos

$$(n^2 + n)/2 \text{ flops.} \tag{3.74}$$

[4] Temos que $\sum_{k=1}^{n} k = n(n + 1)/2, \qquad \sum_{k=1}^{n} 1 = n$

[5] Diferentemente de uma matriz esparsa, uma matriz densa possui a maioria dos elementos diferentes de zero.

Sendo assim, o sistema pode ser reescrito da seguinte forma:

$$\begin{aligned} Ax &= b && (3.77)\\ (LU)x &= b && (3.78)\\ L(Ux) &= b && (3.79)\\ Ly = b \quad &\text{e} \quad Ux = y && (3.80) \end{aligned}$$

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[6] 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

$$L_{21} = A_{21}/A_{11} \tag{3.81}$$

e fazemos

$$A_{2,:} \leftarrow A_{2,:} - L_{21}A_{1,:} \tag{3.82}$$

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

$$L_{31} = A_{31}/A_{11} \tag{3.83}$$

e fazemos

$$A_{3,:} \leftarrow A_{3,:} - L_{31}A_{1,:} \tag{3.84}$$

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[7].

Exemplo 3.4.1. Use a fatoração LU para resolver o seguinte sistema linear:

$$\begin{aligned} x_1 + x_2 + x_3 &= -2\\ 2x_1 + x_2 - x_3 &= 1\\ 2x_1 - x_2 + x_3 &= 3 \end{aligned} \tag{3.85}$$

[6] Não vamos usar pivotamento nesse primeiro exemplo.

[7] 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

Solução. Começamos fatorando a matriz A dos coeficientes deste sistema:

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & -1 \\ 2 & -1 & 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{I_{3,3}} \underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 2 & 1 & -1 \\ 2 & -1 & 1 \end{bmatrix}}_{A} \tag{3.86}$$

$$= \begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 2 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & -1 & -3 \\ 0 & -3 & -1 \end{bmatrix} \tag{3.87}$$

$$= \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 2 & 3 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} 1 & 1 & 1 \\ 0 & -1 & -3 \\ 0 & 0 & 8 \end{bmatrix}}_{U} \tag{3.88}$$

(3.89)

Completada a fatoração LU, resolvemos, primeiramente, o sistema $Ly = b$:

$$\begin{aligned} y_1 &= -2 \\ 2y_1 + y_2 &= 1 \\ 2y_1 + 3y_2 + y_3 &= 3 \end{aligned} \tag{3.90}$$

o qual nos fornece $y_1 = -2$, $y_2 = 5$ e $y_3 = -8$. Por fim, obtemos a solução resolvendo o sistema $Ux = y$:

$$\begin{aligned} x_1 + x_2 + x_3 &= -2 \\ -x_2 - 3x_3 &= 5 \\ 8x_3 &= -8 \end{aligned} \tag{3.91}$$

o qual fornece $x_3 = -1$, $x_2 = -2$ e $x_1 = 1$. ◊

3.4.1 Código Scilab: Fatoração LU

No `Scilab`, podemos implementar o algoritmo para fatoração LU da seguinte forma:

```
1 function [L,A]=fatoraLU(A)
2     n=size(A,1)
```

```
3     L=eye(n,n)
4     for j=1:n-1
5         for i=j+1:n
6             L(i,j )=A(i,j)/A(j,j)
7             A(i,j+1:n)=A(i,j+1:n)-L(i,j)*A(j,j+1:n)
8             A(i,j )=0
9        end
10    end
11 endfunction
```

./cap_linsis/codes/scilab/fatoraLU/fatoraLU.sci

Custo computacional

Podemos analisar o custo computacional, reduzindo o problema a problemas menores.

Na linha 4, iniciamos com $j = 1$. Desta forma i varia de 2 até n na linha 5.

A linha 6 terá sempre 1 flop.

A linha 7, com $j = 1$ tem um bloco de tamanho `2:n` contabilizando $n-1$ flops do produto e $n-1$ flops da subtração.

Nas linhas 6-8 são feitas $(2(n-1)+1) = 2n-1$ flops independente do valor de i. Como i varia de 2 até n, teremos que o bloco é repetido $n-1$ vezes, ou seja, o custo das linhas 5-9 é

$$(n-1) \times (2(n-1)+1) = 2(n-1)^2 + (n-1) \tag{3.92}$$

Voltamos a linha 4 quando $j = 2$. Da linha 7 teremos $n-2$ flops (o bloco terá um elemento a menos) que será repetido $n-2$ vezes, pois `i=3:n`, ou seja,

$$(n-2) \times (2(n-2)+1) = 2(n-2)^2 + (n-2) \tag{3.93}$$

Para $j = 3$, temos $2(n-3)^2 + (n-3)$.

Para $j = n-2$, temos $2(2)^2 + 2$.

Finalmente, para $j = n-1$, temos $2 \cdot 1^2 + 1$.

Somando todos esses custos, temos

$$\begin{aligned} 2(n-1)^2 + (n-1) &+ 2(n-2)^2 + (n-2) + \dots && (3.94)\\ \dots &+ 2(2)^2 + (2) + 2\cdot 1 + 1 && (3.95)\\ &= \sum_{k=1}^{n-1} 2k^2 + k && (3.96)\\ &= 2\sum_{k=1}^{n-1} k^2 + \sum_{k=1}^{n-1} k && (3.97)\\ &= 2\frac{(n-1)n(2n-1)}{6} + \frac{n(n-1)}{2} && (3.98)\\ &= \frac{2n^3}{3} - \frac{n^2}{2} - \frac{n}{6} \text{ flops.} && (3.99) \end{aligned}$$

3.4.2 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

$$\frac{2n^3}{3} - \frac{n^2}{2} - \frac{n}{6} \text{ flops.} \tag{3.100}$$

Depois devemos resolver os sistemas $Ly = b$ e $Ux = y$. O custo de resolver os dois sistemas é (devemos contar duas vezes)

$$2n^2 \text{ flops.} \tag{3.101}$$

Somando esses 3 custos, temos que o custo para resolver um sistema linear usando fatoração LU é

$$\frac{2n^3}{3} + \frac{3n^2}{2} - \frac{n}{6} \text{ flops.} \tag{3.102}$$

Quando n cresce, prevalessem os termos de mais alta ordem, ou seja,

$$\mathcal{O}(\frac{2n^3}{3} + \frac{3n^2}{2} - \frac{n}{6}) = \mathcal{O}(\frac{2n^3}{3} + \frac{3n^2}{2}) = \mathcal{O}(\frac{2n^3}{3}) \tag{3.103}$$

3.4.3 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

$$m(\frac{2n^3}{3} + \frac{3n^2}{2} - \frac{n}{6}) = \frac{2mn^3}{3} + \frac{3mn^2}{2} - \frac{mn}{6} \tag{3.104}$$

e com $m = n$ temos

$$\frac{2n^4}{3} + \frac{3n^3}{2} - \frac{n^2}{6}. \tag{3.105}$$

Porém, se estivermos resolvendo m 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

$$\frac{2n^3}{3} - \frac{n^2}{2} - \frac{n}{6} + 2mn^2 \tag{3.106}$$

que quando $m = n$ obtemos

$$\frac{8n^3}{3} - \frac{n^2}{2} - \frac{n}{6} \text{ flops.} \tag{3.107}$$

3.4.4 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 $\boldsymbol{e}_i = [0,...,0,1,0,....0]^T$, isto é,

$$A\boldsymbol{x}_i = \boldsymbol{e}_i, \qquad i = 1 : n \tag{3.108}$$

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

$$\frac{8n^3}{3} - \frac{n^2}{2} - \frac{n}{6}. \tag{3.109}$$

Exemplo 3.4.2. Qual o melhor método para resolver um sistema linear: via fatoração LU ou calculando a inversa de A e obtendo $x = A^{-1}b$?

3.5 Método da matriz tridiagonal

O método da matriz tridiagonal ou algoritmo de Thomas[8] ou ainda TDMA (do inglês *tridiagonal matrix algorithm*) é o caso particular da eliminação gaussiana aplicada a matrizes tridiagonais.

[8]Llewellyn Hilleth Thomas (21 de outubro de 1903 – 20 de abril de 1992) foi um matemático e físico britânico.

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:

$$\begin{bmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}. \tag{3.110}$$

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:

$$a_1 = c_n = 0. \tag{3.111}$$

Para resolver o sistema tridiagonal (**??**) pelo algoritmo de Thomas são utilizadas as seguintes expressões:

$$c'_i = \begin{cases} \frac{c_i}{b_i}, & i = 1 \\ \frac{c_i}{b_i - a_i c'_{i-1}}, & i = 2, 3, \ldots, n-1 \end{cases} \tag{3.112}$$

e

$$d'_i = \begin{cases} \frac{d_i}{b_i}, & i = 1 \\ \frac{d_i - a_i d'_{i-1}}{b_i - a_i c'_{i-1}}, & i = 2, 3, \ldots, n. \end{cases} \tag{3.113}$$

Finalmente, a solução final é obtida por substituição reversa:

$$x_n = d'_n \tag{3.114}$$

$$x_i = d'_i - c'_i x_{i+1}, \qquad i = n-1, n-2, \ldots, 1. \tag{3.115}$$

Teorema 3.5.1. *A aplicação da eliminação gaussiana sem pivotamento ao sistema* (**??**) *produz o algoritmo dado em* (**??**) *e* (**??**).

Proof. O primeiro passo consiste em dividir todos os elementos da primeira linha de (**??**) por b_1:

$$\begin{bmatrix} 1 & c'_1 & & & \\ a_2 & b_2 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d'_1 \\ d_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}, \tag{3.116}$$

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 ($l_2 \leftarrow l_2 - a_2 l_1$):

$$\begin{bmatrix} 1 & c'_1 & & & \\ 0 & b_2 - a_2 c'_1 & c_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d'_1 \\ d_2 - a_2 d'_1 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}. \tag{3.117}$$

Em seguida, dividimos a segunda linha por $b_2 - a_2 c'_1$, a fim de normalizar a diagonal principal:

$$\begin{bmatrix} 1 & c'_1 & & & \\ 0 & 1 & c'_2 & & \\ & a_3 & b_3 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d'_1 \\ d'_2 \\ d_3 \\ \vdots \\ d_n \end{bmatrix}. \tag{3.118}$$

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 ($l_3 \leftarrow l_3 - a_3 l_2$):

$$\begin{bmatrix} 1 & c'_1 & & & \\ 0 & 1 & c'_2 & & \\ & 0 & b_3 - a_3 c'_2 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d'_1 \\ d'_2 \\ d_3 - a_3 d'_2 \\ \vdots \\ d_n \end{bmatrix}. \tag{3.119}$$

A fim de normalizar o elemento da diagonal da terceira linha, dividimos toda a linha por $d_3 - a_3 d'_2$:

$$\begin{bmatrix} 1 & c'_1 & & & \\ 0 & 1 & c'_2 & & \\ & 0 & 1 & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d'_1 \\ d'_2 \\ d'_3 \\ \vdots \\ d_n \end{bmatrix}. \tag{3.120}$$

Este procedimento é realizado até que se atinja a última linha e temos o seguinte sistema:

$$\begin{bmatrix} 1 & c'_1 & & & \\ 0 & 1 & c'_2 & & \\ & 0 & 1 & \ddots & \\ & & \ddots & \ddots & c'_{n-1} \\ & & & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d'_1 \\ d'_2 \\ d'_3 \\ \vdots \\ d'_n \end{bmatrix}. \tag{3.121}$$

Neste estágio, podemos encontrar os x_n através de substituição reversa, isto é: a última linha diz

$$x_n = d'_n. \tag{3.122}$$

A penúltima linha diz

$$x_{n-1} + c'_{n-1} x_n = d'_{n-1} \Longrightarrow x_{n-1} = d'_{n-1} - c'_{n-1} x_n. \tag{3.123}$$

Esse mesmo procedimento aplicado à linha $i = 1, \ldots n-1$, nos dá

$$x_i = d'_i - c'_i x_{i+1}. \tag{3.124}$$

□

Exemplo 3.5.1. Considere a resolução do seguinte sistema tridiagonal pelo algoritmo de Thomas:

$$\begin{bmatrix} 2 & 1 & 0 & 0 & 0 \\ 1 & 2 & 1 & 0 & 0 \\ 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \\ 0 \\ 0 \\ 2 \end{bmatrix}. \tag{3.125}$$

Primeiramente identificamos os vetores a, b, c e d:

$$a = (0, 1, 1, 1, 1) \tag{3.126}$$
$$b = (2, 2, 2, 2, 2) \tag{3.127}$$
$$c = (1, 1, 1, 1, 0) \tag{3.128}$$
$$d = (4, 4, 0, 0, 2) \tag{3.129}$$

Agora, calculamos os vetores c' e d':

$$c'_1 = \frac{c_1}{b_1} = \frac{1}{2} \tag{3.130}$$
$$c'_2 = \frac{c_2}{b_2 - a_2 c'_1} = \frac{1}{2 - 1 \cdot \frac{1}{2}} = \frac{2}{3} \tag{3.131}$$
$$c'_3 = \frac{c_3}{b_3 - a_3 c'_2} = \frac{1}{2 - 1 \cdot \frac{2}{3}} = \frac{3}{4} \tag{3.132}$$
$$c'_4 = \frac{c_4}{b_4 - a_4 c'_3} = \frac{1}{2 - 1 \cdot \frac{3}{4}} = \frac{4}{5} \tag{3.133}$$
$$d'_1 = \frac{d_1}{b_1} = \frac{4}{2} = 2 \tag{3.134}$$
$$d'_2 = \frac{d_2 - a_2 d'_1}{b_2 - a_2 c'_1} = \frac{4 - 1 \cdot 2}{2 - 1 \cdot \frac{1}{2}} = \frac{4}{3} \tag{3.135}$$
$$d'_3 = \frac{d_3 - a_3 d'_2}{b_3 - a_3 c'_2} = \frac{0 - 1 \cdot \frac{4}{3}}{2 - 1 \cdot \frac{2}{3}} = -1 \tag{3.136}$$
$$d'_4 = \frac{d_4 - a_4 d'_3}{b_4 - a_4 c'_3} = \frac{0 - 1 \cdot (-1)}{2 - 1 \cdot \frac{3}{4}} = \frac{4}{5} \tag{3.137}$$
$$d'_5 = \frac{d_5 - a_5 d'_4}{b_5 - a_5 c'_4} = \frac{2 - 1 \cdot \frac{4}{5}}{2 - 1 \cdot \frac{4}{5}} = 1 \tag{3.138}$$

Finalmente, calculamos o vetor x:

$$x_5 = d'_5 = 1 \tag{3.139}$$

$$x_4 = d'_4 - c'_4 \cdot x_5 = \frac{4}{5} - \frac{4}{5} \cdot 1 = 0 \tag{3.140}$$

$$x_3 = d'_3 - c'_3 \cdot x_4 = -1 - \frac{3}{4} \cdot 0 = -1 \tag{3.141}$$

$$x_2 = d'_2 - c'_2 \cdot x_3 = \frac{4}{3} - \frac{2}{3} \cdot (-1) = 2 \tag{3.142}$$

$$x_1 = d'_1 - c'_1 \cdot x_2 = 2 - \frac{1}{2} \cdot 2 = 1 \tag{3.143}$$

E assim, obtemos o vetor $x = [1, 2, -1, 0, 1]$.

Código Scilab: Método da matriz tridiagonal

```
//entradas: vetores coluna a,b,c,d
//saida:    vetor coluna x
function x=TDMA(a,b,c,d)
    n=size(a,1) // Recupera ordem do sistema.

    cl=zeros(n,1) //Inicializa cl
    dl=zeros(n,1) //Inicializa dl
    x=zeros(n,1) //Inicializa x

    cl(1)=c(1)/b(1)
    for i=2:n-1
        cl(i)=c(i)/(b(i)-a(i)*cl(i-1))
    end

    dl(1)=d(1)/b(1)
    for i=2:n
        dl(i)=(d(i)-a(i)*dl(i-1))/(b(i)-a(i)*cl(i-1))
    end

    x(n)=dl(n)
    for i=n-1:-1:1
        x(i)=dl(i)-cl(i)*x(i+1)
    end
endfunction
```

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:

```
//entradas: vetores coluna a,b,c,d
//saida:    vetor coluna x
function x=TDMA2(a,b,c,d)
    n=size(a,1) // Recupera ordem do sistema.
    x=zeros(n,1) //Inicializa x

    c(1)=c(1)/b(1)
    for i=2:n-1
        c(i)=c(i)/(b(i)-a(i)*c(i-1))
    end

    d(1)=d(1)/b(1)
    for i=2:n
        d(i)=(d(i)-a(i)*d(i-1))/(b(i)-a(i)*c(i-1))
    end

    x(n)=d(n)
    for i=n-1:-1:1
        x(i)=d(i)-c(i)*x(i+1)
    end
endfunction
```

A solução do sistema do Exemplo **??** pode ser obtida através dos seguintes comandos:

```
-->a=[0; 1; 1; 1; 1];
-->b=[2; 2; 2; 2; 2];
-->c=[1; 1; 1; 1; 0];
-->d=[4; 4; 0; 0; 2];
-->TDMA(a,b,c,d)
```

E 3.5.1. Considere o problema linear tridiagonal dado por

$$\begin{bmatrix} 5 & 4 & 0 & 0 & 0 & 0 \\ 1 & 3 & 1 & 0 & 0 & 0 \\ 0 & 2 & 4 & 1 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 0 \\ 0 & 0 & 0 & 2 & 3 & 2 \\ 0 & 0 & 0 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{bmatrix} = \begin{bmatrix} 13 \\ 10 \\ 20 \\ 16 \\ 35 \\ 17 \end{bmatrix}. \tag{3.144}$$

Identifique os vetores a, b, c e d relativos ao algoritmo da matriz tridiagonal. Depois resolva o sistema usando o computador.

E 3.5.2. Considere o seguinte sistema de equações lineares:

$$\begin{aligned} x_1 - x_2 &= 0 \\ -x_{j-1} + 5x_j - x_{j+1} &= \cos(j/10), \ \ 2 \leq j \leq 10 \\ x_{11} &= x_{10}/2 \end{aligned} \tag{3.150}$$

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 **??**

3.6 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 3.6.1. Observe que o sistema

$$\begin{bmatrix} 71 & 41 \\ \lambda & 30 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 100 \\ 70 \end{bmatrix} \tag{3.158}$$

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.

3.6.1 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:

$$\|v\|_p := \left(\sum_{i=1}^{n} |v_i|^p \right)^{1/p} = \left(|v_1|^p + |v_2|^p + \cdots + |v_n|^p \right)^{1/p}, \quad 1 \leq p < \infty. \tag{3.159}$$

Para $p = \infty$, definimos a norma L^∞ (**norma do máximo**) por:

$$\|v\|_\infty = \max_{1\leq j\leq n} \{|v_j|\}. \tag{3.160}$$

Proposição 3.6.1 (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* $p \to \infty$.

Proof. 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:

$$|u_i| \leq \max_{1 \leq j \leq n} \{|u_j|\} = \|u\|_\infty = 0 \Rightarrow u_i = 0. \tag{3.161}$$

Isto é, $u = 0$. Agora, se $\|u\|_p = 0$, $1 \leq p < \infty$, então:

$$0 = \|u\|_p^p := \sum_{i=1}^{n} |u_i|^p \leq n \, \|u\|_\infty \Rightarrow \|u\|_\infty = 0. \tag{3.162}$$

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 3.6.2. Calcule a norma L^1, L^2 e L^∞ do vetor coluna $v = (1, 2, -3, 0)$.

Solução.

$$\|v\|_1 = 1 + 2 + 3 + 0 = 6 \tag{3.163}$$

$$\|v\|_2 = \sqrt{1 + 2^2 + 3^2 + 0^2} = \sqrt{14} \tag{3.164}$$

$$\|v\|_\infty = \max\{1,2,3,0\} = 3 \tag{3.165}$$

No `Scilab` podemos computar normas L^p's de vetores usando o comando `norm`. Neste exemplo, temos:

```
-->norm(v,1), norm(v,'inf'), norm(v,2)
 ans  =
    6.
 ans  =
    3.
 ans  =
    3.7416574
```

◊

3.6.2 Norma de matrizes

Definimos a norma induzida L^p de uma matriz $A = [a_{i,j}]_{i,j=1}^{n,n}$ da seguinte forma:

$$\|A\|_p = \sup_{\|v\|_p=1} \|Av\|_p, \tag{3.166}$$

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):

$$\begin{aligned}
\|A\|_p &= 0 \iff A = 0 && (3.167)\\
\|\lambda A\|_p &= |\lambda| \, \|A\|_p && (3.168)\\
\|A + B\|_p &\leq \|A\|_p + \|B\|_p \quad \text{(desigualdade do triângulo)} && (3.169)\\
\|Av\|_p &\leq \|A\|_p \, \|v\|_p && (3.170)\\
\|AB\|_p &\leq \|A\|_p \, \|B\|_p && (3.171)\\
\|I\|_p &= 1 && (3.172)\\
1 &= \|I\|_p = \|AA^{-1}\|_p \leq \|A\|_p \, \|A^{-1}\|_p \quad \text{(se A é inversível)} && (3.173)
\end{aligned}$$

Casos especiais:

$$\begin{aligned}
\|A\|_1 &= \max_{j=1}^{n} \sum_{i=1}^{n} |a_{ij}| && (3.174)\\
\|A\|_2 &= \sqrt{\max\{|\lambda| : \lambda \in \sigma(AA^*)\}} && (3.175)\\
\|A\|_\infty &= \max_{i=1}^{n} \sum_{j=1}^{n} |a_{ij}| && (3.176)
\end{aligned}$$

onde $\sigma(M)$ é o conjunto de autovalores da matriz M.

Exemplo 3.6.3. Calcule as normas 1, 2 e ∞ da seguinte matriz:

$$A = \begin{bmatrix} 3 & -5 & 7 \\ 1 & -2 & 4 \\ -8 & 1 & -7 \end{bmatrix} \tag{3.177}$$

Solução.

$$\begin{aligned}
\|A\|_1 &= \max\{12, 8, 18\} = 18 && (3.178)\\
\|A\|_\infty &= \max\{15, 7, 16\} = 16 && (3.179)\\
\|A\|_2 &= \sqrt{\max\{0{,}5865124, 21{,}789128, 195{,}62436\}} = 13{,}98657 && (3.180)
\end{aligned}$$

No `Scilab` podemos computar normas L^p's de matrizes usando o comando `norm`. Neste exemplo, temos:

```
-->A = [3 -5 7;1 -2 4;-8 1 -7];
-->norm(A,1), norm(A,'inf'), norm(A,2)
 ans  =
    18.
 ans  =
    16.
 ans  =
    13.986578
```

$\Diamond$

3.6.3 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

$$A(x + \delta_x) = y + \delta_y \tag{3.181}$$

Aqui, δ_x representa uma variação (erro) em x e δ_y representa uma variação em y (erro). Temos:

$$Ax + A\delta_x = y + \delta_y \tag{3.182}$$

e, portanto,

$$A\delta_x = \delta_y. \tag{3.183}$$

Queremos avaliar a razão entre o erro relativo em x e o erro relativo em y, isto é

$$\frac{\|\delta_x\| \,/\, \|x\|}{\|\delta_y\| \,/ \|y\|} = \frac{\|\delta_x\|}{\|x\|} \frac{\|y\|}{\|\delta_y\|} \tag{3.184}$$

$$= \frac{\|A^{-1}\delta_y\|}{\|x\|} \frac{\|Ax\|}{\|\delta_y\|} \tag{3.185}$$

$$\leq \frac{\|A^{-1}\| \, \|\delta_y\|}{\|x\|} \frac{\|A\| \, \|x\|}{\|\delta_y\|} \tag{3.186}$$

$$= \|A\| \, \|A^{-1}\| \tag{3.187}$$

Definição 3.6.1 (Número de condicionamento)**.** *O número de condicionamento de uma matriz não-singular A é*

$$k_p(A) := \|A\|_p \, \|A^{-1}\|_p \tag{3.188}$$

Observação 3.6.1.

- 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 3.6.4. No Exemplo **??** estudamos a solução de sistemas lineares com as seguintes matrizes de coeficientes:

$$A_1 = \begin{bmatrix} 71 & 41 \\ 51 & 30 \end{bmatrix} \quad \text{e} \quad A_2 = \begin{bmatrix} 71 & 41 \\ 52 & 30 \end{bmatrix}. \tag{3.189}$$

Calcule os números de condicionamento destes sistemas na norma L^p para $p = 1$, 2 e ∞.

Solução. Para a matriz A_1, temos:

$$\begin{aligned} k_1(A_1) &:= \|A_1\|\|A_1^{-1}\| \approx 350{,}36, \\ k_2(A_1) &:= \|A_2\|\|A_2^{-1}\| \approx 262{,}12, \\ k_\infty(A_1) &:= \|A_\infty\|\|A_\infty^{-1}\| \approx 350{,}36. \end{aligned} \tag{3.190}$$

Para a matriz A_2, temos:

$$\begin{aligned} k_1(A_2) &:= \|A_1\|_1\|A_1^{-1}\|_1 \approx 6888{,}0, \\ k_2(A_2) &:= \|A_1\|_2\|A_1^{-1}\|_2 \approx 5163{,}0, \\ k_\infty(A_2) &:= \|A_1\|_\infty\|A_1^{-1}\|_\infty \approx 6888{,}0. \end{aligned} \tag{3.191}$$

No `Scilab`, podemos computar estes números de condicionamento para a matriz A_1 com o seguinte código:

```
A1 = [71 41;51 30];
cond(A1,1)
cond(A1,2)
cond(A1,'inf')
```

e analogamente para a matriz A_2. ◊

Exercícios

E 3.6.1. Calcule o valor de λ para o qual o problema

$$\begin{cases} 71x + 41y = 10 \\ \lambda x + 30y = 4 \end{cases} \tag{3.192}$$

é impossível, depois calcule os números de condicionamento com norma 1,2 e ∞ quando $\lambda = 51$ e $\lambda = 52$.

E 3.6.2. Calcule o número de condicionamento da matriz

$$A = \begin{bmatrix} 3 & -5 & 7 \\ 1 & -2 & 4 \\ -8 & 1 & -7 \end{bmatrix} \tag{3.193}$$

nas normas 1, 2 e ∞.

E 3.6.3. Calcule o número de condicionamento das matrizes

$$\begin{bmatrix} 71 & 41 \\ 52 & 30 \end{bmatrix} \tag{3.194}$$

e

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 4 & 5 & 5 \end{bmatrix} \tag{3.195}$$

usando as normas 1, 2 e ∞.

E 3.6.4. Usando a norma 1, calcule o número de condicionamento da matriz

$$A = \begin{bmatrix} 1 & 2 \\ 2+\varepsilon & 4 \end{bmatrix} \tag{3.196}$$

em função de ε quando $0 < \varepsilon < 1$. Interprete o limite $\varepsilon \to 0$.

E 3.6.5. Considere os sistemas:

$$\begin{cases} 100000x - 9999.99y = -10 \\ -9999.99x + 1000.1y = 1 \end{cases} \quad \text{e} \quad \begin{cases} 100000x - 9999.99y = -9.999 \\ -9999.99x + 1000.1y = 1.01 \end{cases} \tag{3.197}$$

Encontre a solução de cada um e discuta.

E 3.6.6. Considere os vetores de 10 entradas dados por

$$x_j = \text{sen}\,(j/10), \quad y_j = j/10 \quad z_j = j/10 - \frac{(j/10)^3}{6}, \;\; j = 1,\ldots,10 \tag{3.198}$$

Use o `Scilab` para construir os seguintes vetores de erro:

$$e_j = \frac{|x_j - y_j|}{|x_j|} \quad f_j = \frac{|x_j - z_j|}{x_j} \tag{3.199}$$

Calcule as normas 1, 2 e ∞ de e e f

3.7 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.

3.7.1 Método de Jacobi

O método de Jacobi pode ser obtido a partir do sistema linear

$$\begin{aligned}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= y_1 && (3.200)\\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= y_2 && (3.201)\\
&\vdots && (3.202)\\
a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= y_n && (3.203)
\end{aligned}$$

Isolando o elemento x_1 da primeira equação temos

$$x_1^{(k+1)} = \frac{y_1 - \left(a_{12}x_2^{(k)} + \cdots + a_{1n}x_n^{(k)}\right)}{a_{11}} \tag{3.204}$$

Note que utilizaremos os elementos $x_i^{(k)}$ da iteração k (à 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

$$x_1^{(k+1)} = \frac{y_1 - \left(a_{12}x_2^{(k)} + \cdots + a_{1n}x_n^{(k)}\right)}{a_{11}} \tag{3.205}$$

$$x_2^{(k+1)} = \frac{y_2 - \left(a_{21}x_1^{(k)} + a_{23}x_3^{(k)} + \cdots + a_{2n}x_n^{(k)}\right)}{a_{22}} \tag{3.206}$$

$$\vdots \tag{3.207}$$

$$x_n^{(k+1)} = \frac{y_n - \left(a_{n1}x_1^{(k)} + \cdots + a_{n,n-2}x_{n-2}^{(k)} + a_{n,n-1}x_{n-1}^{(k)}\right)}{a_{nn}} \tag{3.208}$$

Em notação mais compacta, o método de Jacobi consiste na iteração

$$x^{(1)} = \text{aproximação inicial} \tag{3.209}$$

$$x_i^{(k+1)} = \left(y_i - \sum_{\substack{j=1 \\ j\neq i}}^{n} a_{ij}x_j^{(k)}\right)/a_{ii} \tag{3.210}$$

Exemplo 3.7.1. Resolva o sistema

$$10x + y = 23 \tag{3.211}$$

$$x + 8y = 26 \tag{3.212}$$

usando o método de Jacobi iniciando com $x^{(1)} = y^{(1)} = 0$.

$$x^{(k+1)} = \frac{23 - y^{(k)}}{10} \tag{3.213}$$

$$y^{(k+1)} = \frac{26 - x^{(k)}}{8} \tag{3.214}$$

$$x^{(2)} = \frac{23 - y^{(1)}}{10} = 2{,}3 \tag{3.215}$$

$$y^{(2)} = \frac{26 - x^{(1)}}{8} = 3{,}25 \tag{3.216}$$

$$x^{(3)} = \frac{23 - y^{(2)}}{10} = 1{,}975 \tag{3.217}$$

$$y^{(3)} = \frac{26 - x^{(2)}}{8} = 2{,}9625 \tag{3.218}$$

Exemplo 3.7.2. Considere o seguinte sistema

$$\begin{aligned} -3x_1 + x_2 + x_3 &= 2 \\ 2x_1 + 5x_2 + x_3 &= 5 \\ 2x_1 + 3x_2 + 7x_3 &= -17 \end{aligned} \tag{3.219}$$

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
$\vdots$	$\vdots$	$\vdots$
10	$(-1{,}00,\ 2{,}00,\ -3{,}00)$	6,0E$-$3

Verifique a resposta.

Código Scilab: Método de Jacobi

```
function [x,deltax]=jacobi(A,b,x,tol,N)
n=size(A,1)
xnew     =x
convergiu=%F                //FALSE;

k=1
while k<=N & ~convergiu
  xnew(1)=(b(1) - A(1,2:n)*x(2:n))/A(1,1)
  for i=2:n-1
    xnew(i)=(b(i) -A(i,1:i-1)*x(1:i-1) ...
                  -A(i,i+1:n)*x(i+1:n) )/A(i,i)
  end
  xnew(n)=  (b(n) -A(n,1:n-1)*x(1:n-1) )/A(n,n)

  deltax=max( abs(x-xnew) )
  if deltax<tol then
     convergiu=%T       //TRUE
  end
  k=k+1
  x=xnew                 // atualiza x
  disp([k,x',deltax])    // depuracao
end
```

```
if ~convergiu then
    error('Nao convergiu')
end

endfunction
```

3.7.2 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

$$x_2^{(k+1)} = \frac{y_2 - \left(a_{21}x_1^{(k+1)} + a_{23}x_3^{(k)} + \cdots + a_{2n}x_n^{(k)}\right)}{a_{22}} \tag{3.220}$$

Aplicando esse raciocínio, podemos construir o método de Gauss-Seidel como

$$x_1^{(k+1)} = \frac{y_1 - \left(a_{12}x_2^{(k)} + \cdots + a_{1n}x_n^{(k)}\right)}{a_{11}} \tag{3.221}$$

$$x_2^{(k+1)} = \frac{y_2 - \left(a_{21}x_1^{(k+1)} + a_{23}x_3^{(k)} + \cdots + a_{2n}x_n^{(k)}\right)}{a_{22}} \tag{3.222}$$

$$\vdots \tag{3.223}$$

$$x_n^{(k+1)} = \frac{y_n - \left(a_{n1}x_1^{(k+1)} + \cdots + a_{n(n-1)}x_{n-1}^{(k+1)}\right)}{a_{nn}} \tag{3.224}$$

Em notação mais compacta, o método de Gauss-Seidel consiste na iteração:

$$x^{(1)} = \text{aproximação inicial} \tag{3.225}$$

$$x_i^{(k+1)} = \frac{y_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}}{a_{ii}} \tag{3.226}$$

Exemplo 3.7.3. Resolva o sistema

$$10x + y = 23 \tag{3.227}$$

$$x + 8y = 26 \tag{3.228}$$

usando o método de Gauss-Seidel com condições iniciais $x^{(1)} = y^{(1)} = 0$.

$$x^{(k+1)} = \frac{23 - y^{(k)}}{10} \tag{3.229}$$

$$y^{(k+1)} = \frac{26 - x^{(k+1)}}{8} \tag{3.230}$$

$$x^{(2)} = \frac{23 - y^{(1)}}{10} = 2{,}3 \tag{3.231}$$

$$y^{(2)} = \frac{26 - x^{(2)}}{8} = 2{,}9625 \tag{3.232}$$

$$x^{(3)} = \frac{23 - y^{(2)}}{10} = 2{,}00375 \tag{3.233}$$

$$y^{(3)} = \frac{26 - x^{(3)}}{8} = 2{,}9995312 \tag{3.234}$$

Código Scilab: Método de Gauss-Seidel

```
function [x,deltax]=gauss_seidel(A,b,x,tol,N)
n=size(A,1)
xnew     =x
convergiu=%F                //FALSE;

k=1
while k<=N & ~convergiu
  xnew(1)=(b(1) - A(1,2:n)*x(2:n))/A(1,1)
  for i=2:n-1
    xnew(i)=(b(i) -A(i,1:i-1)*xnew(1:i-1) ...
                  -A(i,i+1:n)*x(i+1:n) )/A(i,i)
  end
  xnew(n)=(b(n) -A(n,1:n-1)*xnew(1:n-1) )/A(n,n)

  deltax=max( abs(x-xnew) )
  if deltax<tol then
     convergiu=%T       //TRUE
  end
  k=k+1
  x=xnew                 // atualiza x
  disp([k,x',deltax])    // depuracao
end
if ~convergiu then
    error('Nao convergiu')
```

```
end

endfunction
```

3.7.3 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 o 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:

$$Ax = b \Leftrightarrow x = Tx + c, \tag{3.235}$$

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:

$$x^{(k+1)} = Tx^{(k)} + c, \quad k \geq 1, \tag{3.236}$$

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:

$$A = L + D + U, \tag{3.237}$$

onde D é a matriz diagonal $D = (a_{11}, a_{22}, \ldots, a_{nn})$, isto é:

$$D := \begin{bmatrix} a_{11} & 0 & 0 & \cdots & 0 \\ 0 & a_{22} & 0 & \cdots & 0 \\ 0 & 0 & a_{33} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & a_{nn} \end{bmatrix}, \tag{3.238}$$

e, respectivamente, L e U são as seguintes matrizes triangular inferior e superior:

$$L := \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 \\ a_{21} & 0 & 0 & \cdots & 0 \\ a_{31} & a_{32} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \cdots & 0 \end{bmatrix}, \quad U := \begin{bmatrix} 0 & a_{12} & a_{13} & \cdots & a_{1n} \\ 0 & 0 & a_{23} & \cdots & a_{2n} \\ 0 & 0 & 0 & \cdots & a_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 0 \end{bmatrix}. \tag{3.239}$$

Exemplo 3.7.4. Considere o seguinte sistema linear:

$$3x_1 + x_2 - x_3 = 2 \tag{3.240}$$
$$-x_1 - 4x_2 + x_3 = -10 \tag{3.241}$$
$$x_1 - 2x_2 - 5x_3 = 10 \tag{3.242}$$

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:

$$A = \begin{bmatrix} 3 & 1 & -1 \\ -1 & -4 & 1 \\ 1 & -2 & -5 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad \text{e} \quad b = \begin{bmatrix} 2 \\ -10 \\ 10 \end{bmatrix}. \tag{3.243}$$

A decomposição da matriz A nas matrizes L triangular inferior, D diagonal e U triangular superior é:

$$\underbrace{\begin{bmatrix} 3 & 1 & -1 \\ -1 & -4 & 1 \\ 1 & -2 & -5 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 0 \\ 1 & -2 & 0 \end{bmatrix}}_{L} + \underbrace{\begin{bmatrix} 3 & 0 & 0 \\ 0 & -4 & 0 \\ 0 & 0 & -5 \end{bmatrix}}_{D} + \underbrace{\begin{bmatrix} 0 & 1 & -1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}}_{U}. \tag{3.244}$$

No `Scilab`, podemos construir as matrizes L, D e U, da seguinte forma:

```
-->A = [3 1 -1;-1 -4 1;1 -2 -5];
-->D = eye(A).*A;
-->L = tril(A)-D;
-->U=triu(A)-D;
```

$\Diamond$

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:

$$\begin{aligned} Ax = b &\Leftrightarrow (L + D + U)x = b && (3.245)\\ &\Leftrightarrow Dx = -(L + U)x + b && (3.246)\\ &\Leftrightarrow x = \underbrace{-D^{-1}(L + U)}_{=:T_J} x + \underbrace{D^{-1}b}_{=:c_J}. && (3.247) \end{aligned}$$

Ou seja, a iteração do método de Jacobi escrita na forma matricial é:

$$x^{(k+1)} = T_J x^{(k)} + c_J, \quad k \geq 1, \tag{3.248}$$

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 3.7.5. Construa a matriz de iteração T_J e o vetor de iteração c_J do método de Jacobi para o sistema dado no Exemplo **??**.

Solução. A matriz de iteração é dada por:

$$T_J := -D^{-1}(L + U) = -\underbrace{\begin{bmatrix} \frac{1}{3} & 0 & 0 \\ 0 & -\frac{1}{4} & 0 \\ 0 & 0 & -\frac{1}{5} \end{bmatrix}}_{D^{-1}} \underbrace{\begin{bmatrix} 0 & 1 & -1 \\ -1 & 0 & 1 \\ 1 & 2 & 0 \end{bmatrix}}_{(L+U)} = \begin{bmatrix} 0 & -\frac{1}{3} & \frac{1}{3} \\ -\frac{1}{4} & 0 & \frac{1}{4} \\ \frac{1}{5} & \frac{2}{5} & 0 \end{bmatrix}. \tag{3.249}$$

O vetor da iteração de Jacobi é:

$$c_J := D^{-1}b = \underbrace{\begin{bmatrix} \frac{1}{3} & 0 & 0 \\ 0 & -\frac{1}{4} & 0 \\ 0 & 0 & -\frac{1}{5} \end{bmatrix}}_{D^{-1}} \underbrace{\begin{bmatrix} 2 \\ -10 \\ 10 \end{bmatrix}}_{b} = \begin{bmatrix} \frac{2}{3} \\ \frac{5}{2} \\ -2 \end{bmatrix}. \tag{3.250}$$

No `Scilab`, podemos computar T_J e c_J da seguinte forma:

```
-->TJ = -inv(D)*(L+U);
-->cJ = inv(D)*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:

$$\begin{aligned} Ax = b &\Leftrightarrow (L + D + U)x = b &(3.251)\\ &\Leftrightarrow (L + D)x = -Ux + b &(3.252)\\ &\Leftrightarrow x = \underbrace{-(L + D)^{-1}U}_{=:T_G} x + \underbrace{(L + D)^{-1}b}_{=:c_G} &(3.253)\end{aligned}$$

Ou seja, a iteração do método de Gauss-Seidel escrita na forma matricial é:

$$x^{(k+1)} = T_G x^{(k)} + c_G, \quad k \geq 1, \tag{3.254}$$

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 3.7.6. 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 **??**.

Solução. A matriz de iteração é dada por:

$$T_G = -(L + D)^{-1}U = -\underbrace{\begin{bmatrix} 3 & 0 & 0 \\ -1 & -4 & 0 \\ 1 & -2 & -5 \end{bmatrix}^{-1}}_{(L+D)^{-1}} \underbrace{\begin{bmatrix} 0 & 1 & -1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}}_{U} = \begin{bmatrix} 0 & -\frac{1}{3} & \frac{1}{3} \\ 0 & \frac{1}{12} & \frac{1}{6} \\ 0 & -\frac{1}{10} & 0 \end{bmatrix}. \tag{3.255}$$

O vetor da iteração de Gauss-Seidel é:

$$c_G := (L + D)^{-1}b = \underbrace{\begin{bmatrix} 3 & 0 & 0 \\ -1 & -4 & 0 \\ 1 & -2 & -5 \end{bmatrix}^{-1}}_{(L+D)^{-1}} \underbrace{\begin{bmatrix} 2 \\ -10 \\ 10 \end{bmatrix}}_{b} = \begin{bmatrix} \frac{2}{3} \\ \frac{7}{3} \\ -\frac{28}{10} \end{bmatrix}. \tag{3.256}$$

No `Scilab`, podemos computar T_G e c_G da seguinte forma:

```
-->TG = -inv(L+D)*U;
-->cG = inv(L+D)*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:

$$x^{(k+1)} = Tx^{(k)} + c, \quad k \geq 1, \tag{3.257}$$

$x^{(1)}$ 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 3.7.1. *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, $\rho(T) < 1$.*

Proof. Aqui, fazemos apenas um esboço da demonstração. Para mais detalhes, veja [**?**], Teorema 4, pág. 14.

Primeiramente, suponhamos que $\|T\|_p < 1$, $1 \leq p \leq \infty$. Como (veja [**?**], lema 2, pág. 12):

$$\rho(T) \leq \|T\|_p, \tag{3.258}$$

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 [**?**], Teorema 3, página 12):

$$\|T\|_p \leq \rho(T) + \epsilon < 1. \tag{3.259}$$

Assim, temos:

$$\lim_{k\to\infty} \|T^k\|_p \leq \lim_{k\to\infty} \|T\|_p^m = 0. \tag{3.260}$$

Da equivalência entre as normas segue a recíproca. □

Observação 3.7.1. Observamos que:

$$\lim_{k\to\infty} \|T^k\|_p = 0, \quad ,1 \leq p \leq \infty, \Leftrightarrow \lim_{k\to\infty} t_{ij}^k = 0, \quad 1 \leq i,j \leq n. \tag{3.261}$$

Lema 3.7.2. *Se $\rho(T) < 1$, então existe $(I - T)^{-1}$ e:*

$$(I - T)^{-1} = \sum_{k=0}^{\infty} T^k. \tag{3.262}$$

Proof. 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 **??** e da Observação **??** temos:

$$(I - T)\sum_{k=0}^{\infty} T^k = \lim_{m\to\infty}(I - T)\sum_{k=0}^{m} T^k = \lim_{m\to\infty}(I - T^{m+1}) = I, \tag{3.263}$$

o que mostra que $(I-T)^{-1} = \sum_{k=0}^{\infty} T^k$. □

Teorema 3.7.1. *A sequência recursiva* $\{x^{(k)}\}_{k\in\mathbb{N}}$ *dada por:*

$$x^{(k+1)} = Tx^{(k)} + c \tag{3.264}$$

converge para solução de $x = Tx + c$ *para qualquer escolha de* $x^{(1)}$ *se, e somente se,* $\rho(T) < 1$.

Proof. Primeiramente, assumimos que $\rho(T) < 1$. Observamos que:

$$\begin{aligned} x^{(k+1)} &= Tx^{(k)} + c = T(Tx^{(k-1)} + c) + c && (3.265)\\ &= T^2x^{(k-1)} + (I+T)c && (3.266)\\ &\vdots && (3.267)\\ &= T^{(k)}x^{(1)} + \left(\textstyle\sum_{k=0}^{k-1} T^k\right)c. && (3.268) \end{aligned}$$

Daí, do Lema **??** e do Lema **??** temos:

$$\lim_{k\to\infty} x^{(k)} = (I-T)^{(-1)}c. \tag{3.269}$$

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:

$$\begin{aligned} x^* - x^{(k+1)} &= (Tx^* + c) - (Tx^{(k)} + c) && (3.270)\\ &= T(x^* - x^{(k)}) && (3.271)\\ &\vdots && (3.272)\\ &= T^{(k)}(x^* - x^{(1)}) = T^{(k)}y. && (3.273) \end{aligned}$$

Logo, para qualquer $1 \le p \le \infty$, temos, :

$$0 = \lim_{k\to\infty} x^* - x^{(k+1)} = \lim_{k\to\infty} T^{(k)}y. \tag{3.274}$$

Como y é arbitrário, da Observação **??** temos $\lim_{k\to\infty} \|T^{(k)}\|_p = 0$, $1 \le p \le \infty$. Então, o Lema **??** garante que $\rho(T) < 1$. □

Observação 3.7.2. Pode-se mostrar que tais métodos iterativos tem taxa de convergência super linear com:

$$\|x^{(k+1)} - x^*\| \approx \rho(T)^k \|x^{(1)} - x^*\|. \tag{3.275}$$

Para mais detalhes, veja [**?**], pág. 61-64.

Exemplo 3.7.7. 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 **??**.

Solução. Do Teorema **??**, 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. $\Diamond$

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 3.7.1. *Uma matriz A é **estritamente diagonal dominante** quando:*

$$|a_{ii}| > \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, i = 1,...,n \tag{3.276}$$

Definição 3.7.2. *Uma matriz A é **diagonal dominante** quando*

$$|a_{ii}| \geq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|, i = 1,...,n \tag{3.277}$$

e para ao menos um i, a_{ii} é estritamente maior que a soma dos elementos fora da diagonal.

Teorema 3.7.2. *Se a matriz A for diagonal dominante[9], 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:

$$\frac{\|x - x^{(k)}\|}{\|x\|}. \tag{3.278}$$

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.

[9] Ou se for estritamente diagonal dominante e, consequentemente, diagonal dominante.

Teorema 3.7.3. *O erro relativo e o resíduo estão relacionados como (veja [?])*

$$\frac{\|x - x^{(k)}\|}{\|x\|} \leq \kappa(A)\frac{\|r\|}{\|b\|} \tag{3.279}$$

onde $k(A)$ é o número de condicionamento.

Exemplo 3.7.8. Ambos os métodos de Jacobi e Gauss-Seidel são convergentes para o sistema dado no Exemplo **??**, pois a matriz dos coeficientes deste é uma matriz estritamente diagonal dominante.

Exercícios

E 3.7.1. Considere o problema de 5 incógnitas e cinco equações dado por

$$\begin{aligned} x_1 - x_2 &= 1 && (3.280)\\ -x_1 + 2x_2 - x_3 &= 1 && (3.281)\\ -x_2 + (2+\varepsilon)x_3 - x_4 &= 1 && (3.282)\\ -x_3 + 2x_4 - x_5 &= 1 && (3.283)\\ x_4 - x_5 &= 1 && (3.284) \end{aligned}$$

a) Escreva na forma $Ax = b$ e resolva usando eliminação gaussiana para $\varepsilon = 10^{-3}$ no `Scilab`.

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 3.7.2. Resolva o seguinte sistema pelo método de Jacobi e Gauss-Seidel:

$$\begin{cases} 5x_1 + x_2 + x_3 &= 50\\ -x_1 + 3x_2 - x_3 &= 10\\ x_1 + 2x_2 + 10x_3 &= -30 \end{cases} \tag{3.285}$$

Use como critério de paragem tolerância inferior a 10^{-3} e inicialize com $x^0 = y^0 = z^0 = 0$.

E 3.7.3. Refaça o Exercício **??** construindo um algoritmo que implemente os métodos de Jacobi e Gauss-Seidel.

E 3.7.4. Considere o seguinte sistema de equações lineares:

$$\begin{aligned} x_1 - x_2 &= 0 \\ -x_{j-1} + 5x_j - x_{j+1} &= \cos(j/10), \ \ 2 \leq j \leq 10 \\ x_{11} &= x_{10}/2 \end{aligned} \tag{3.286}$$

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 **??**

E 3.7.5. Faça uma permutação de linhas no sistema abaixo e resolva pelos métodos de Jacobi e Gauss-Seidel:

$$x_1 + 10x_2 + 3x_3 = 27 \tag{3.287}$$

$$4x_1 + x_3 = 6 \tag{3.288}$$

$$2x_1 + x_2 + 4x_3 = 12 \tag{3.289}$$

3.8 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 $A = UDU^{-1}$.

3.8.1 Método da potência

O método da potência para cálculo do maior autovalor (em módulo) consiste na iteração

$$\begin{cases} x^{(1)} &= \text{aprox. inicial do autovetor}, \\ \lambda^{(1)} &= {x^{(1)}}^T A x^{(1)}, \\ \end{cases} \\ \begin{cases} x^{(k+1)} &= \frac{Ax^{(k)}}{\|Ax^{(k)}\|_2}, \\ \lambda^{(k+1)} &= {x^{(k+1)}}^T A x^{(k+1)}, \end{cases} \quad , \quad k \geq 1. \tag{3.290}$$

Observação 3.8.1. Observe que na iteração do método da potência (**??**), temos $Ax^{(k)} = A^{(k)}x^{(1)}$.

Exemplo 3.8.1. A seguinte matriz

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 2 & 3 & 0 \\ 3 & 4 & 2 \end{bmatrix} \tag{3.291}$$

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)	15
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
⋮	⋮	⋮
14	(0,00, 0,24, 0,97)	3,00

Código Scilab: Método das Potências

```
function [autovalor, autovetor] = maior_eigen(matriz, vetor_inicial, tolerancia, itera
    // Através do método da potência, retorna o maior autovalor de uma dada matriz e s
    autovetor = vetor_inicial
    autovalor = 0
    for k=1:iteracoes_maximas
        vetor_produto = matriz * autovetor // y = Ax
        norma_vetor_produto = sqrt(sum(vetor_produto .^ 2)) // ||y||
        autovetor = vetor_produto / norma_vetor_produto // x = y / ||y||
        autovalor_atual = autovetor' * vetor_produto //  lambda = x.T * y
        if abs(autovalor - autovalor_atual) < tolerancia then
            return
        end
        autovalor = autovalor_atual
    end
    disp('Quantidade máxima de iterações excedida.')
    return
```

```
endfunction


// matriz (A)
matriz = [
    1, 0, 0
    2, 3, 0
    3, 4, 2
]
// vetor inicial (x)
vetor_inicial = [1; 1; 1]
[autovalor, autovetor] = maior_eigen(matriz=matriz, vetor_inicial=vetor_inicial,
disp(autovalor) // 3.0021537
disp(autovetor) // [9.399D-10, 0.2427675, 0.9700845]
```

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,

$$A = \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_3 & \cdots & 0 \\ & \vdots & & \ddots & \\ 0 & 0 & 0 & \cdots & \lambda_n \end{bmatrix}. \tag{3.292}$$

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

$$A^k x^{(1)} = A^k \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \vdots \\ \xi_n \end{bmatrix} = \begin{bmatrix} \lambda_1^k \xi_1 \\ \lambda_2^k \xi_2 \\ \lambda_3^k \xi_3 \\ \vdots \\ \lambda_n^k \xi_n \end{bmatrix} = \lambda_1^k \xi_1 \begin{bmatrix} 1 \\ \frac{\xi_2}{\xi_1}\left(\frac{\lambda_2}{\lambda_1}\right)^k \\ \frac{\xi_3}{\xi_1}\left(\frac{\lambda_3}{\lambda_1}\right)^k \\ \vdots \\ \frac{\xi_n}{\xi_1}\left(\frac{\lambda_n}{\lambda_1}\right)^k \end{bmatrix}, \tag{3.293}$$

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,

$$x^{(k+1)} = \frac{A^k x^{(1)}}{\|A^k x^{(1)}\|_2} \tag{3.294}$$

converge para $\pm e_1$, $e_1 = (1, 0, 0, \ldots, 0)$. Também, a sequência

$$\lambda^{(k)} = {x^{(k)}}^T A x^{(k)} \tag{3.295}$$

converge para λ_1, pois

$$\lim_{k\to\infty} \lambda^{(k)} = (\pm e_1)^T A(\pm e_1) = \lambda_1 e_1^T e_1 = \lambda_1. \tag{3.296}$$

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:

$$D = \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_3 & \cdots & 0 \\ & \vdots & & \ddots & \\ 0 & 0 & 0 & \cdots & \lambda_n \end{bmatrix}. \tag{3.297}$$

Considere a mesma hipótese sobre os autovalores: $|\lambda_1| > |\lambda_2| \geq |\lambda_3| \geq \cdots \geq |\lambda_n|$. Então

$$A^k = (UDU^{-1})(UDU^{-1})(UDU^{-1})\cdots(UDU^{-1}) = UD^kU^{-1} \tag{3.298}$$

visto que $UU^{-1} = I$. Suponha que o chute inicial $x^{(1)}$ pode ser escrito da forma

$$x^{(1)} = UU^{-1}x^{(1)} = U[\xi_1 \;\; \xi_2 \;\; \xi_3 \;\; \cdots \;\; \xi_n]^T \tag{3.299}$$

com $\xi_1 \neq 0$. Então

$$A^k x^{(1)} = (UD^kU^{-1})U \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \vdots \\ \xi_n \end{bmatrix} = U \begin{bmatrix} \lambda_1^k\xi_1 \\ \lambda_2^k\xi_2 \\ \lambda_3^k\xi_3 \\ \vdots \\ \lambda_n^k\xi_n \end{bmatrix} = \lambda_1^k\xi_1 U \begin{bmatrix} 1 \\ \frac{\xi_2}{\xi_1}\left(\frac{\lambda_2}{\lambda_1}\right)^k \\ \frac{\xi_3}{\xi_1}\left(\frac{\lambda_3}{\lambda_1}\right)^k \\ \vdots \\ \frac{\xi_n}{\xi_1}\left(\frac{\lambda_n}{\lambda_1}\right)^k \end{bmatrix}. \tag{3.300}$$

Como na discussão anterior, o último vetor converge para $\pm e_1$ e

$$x^{(k)} = \frac{A^k x^{(1)}}{\|A^k x^{(1)}\|_2} \tag{3.301}$$

converge para $v_1 = \pm U e_1$ que é um múltiplo do autovetor associado a λ_1. Também, a sequência

$$\lambda^{(k)} = {x^{(k)}}^T A x^{(k)} \tag{3.302}$$

converge para o autovalor dominante λ_1:

$$\lim_{k\to\infty} \lambda^{(k)} = v_1^T A v_1 = \lambda_1 v_1^T v_1 = \lambda_1. \tag{3.303}$$

Observação 3.8.2. 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:

$$x = \sum_{j=1}^{n} \beta_j v_j. \tag{3.304}$$

Observamos que se x está na forma (**??**), então $A^k x$ pode ser escrito como

$$A^k x = \sum_{j=1}^{n} \beta_j A^k v_j = \sum_{j=1}^{n} \beta_j \lambda_j^k v_j = \beta_1 \lambda_1^k \left(v_1 + \sum_{j=2}^{n} \frac{\beta_j}{\beta_1} \left(\frac{\lambda_j}{\lambda_1} \right)^k v_j \right). \tag{3.305}$$

Como $\left|\frac{\lambda_j}{\lambda_1}\right| < 1$ para todo $j \geq 2$, temos

$$\sum_{j=2}^{n} \frac{\beta_j}{\beta_1} \left(\frac{\lambda_j}{\lambda_1} \right)^k v_j \to 0. \tag{3.306}$$

Assim,

$$\frac{A^k x}{\|A^k x\|_2} = \frac{\beta_1 \lambda_1^k}{\|A^k x\|} \left(v_1 + O\left(\left| \frac{\lambda_2}{\lambda_1} \right|^k \right) \right). \tag{3.307}$$

Como a norma de $\frac{A^k x}{\|A^k x\|}$ é igual a um, temos

$$\left\| \frac{\beta_1 \lambda_1^k}{\|A^k x\|} v_1 \right\| \to 1 \tag{3.308}$$

e, portanto,

$$\left| \frac{\beta_1 \lambda_1^k}{\|A^k x\|} \right| \to \frac{1}{\|v_1\|}. \tag{3.309}$$

Ou seja, se definimos $\alpha^{(k)} = \frac{\beta_1 \lambda_1^k}{\|A^k x\|}$, então

$$|\alpha^{(k)}| \to 1. \tag{3.310}$$

Retornando a (**??**), temos:

$$\frac{A^k x}{\|A^k x\|} - \alpha^{(k)} v_1 \to 0. \tag{3.311}$$

Observe que um múltiplo de autovetor também é um autovetor e, portanto,

$$x^{(k)} = \frac{A^k x^{(1)}}{\|A^k x^{(1)}\|}. \tag{3.312}$$

é um esquema que oscila entre os autovetores ou converge para o autovetor v_1.

3.8.2 Método da iteração inversa

Nesta seção, vamos estudar a sequência

$$x_0, \frac{(A-\sigma I)^{-1} x_0}{\|(A-\sigma I)^{-1} x_0\|_2}, \frac{(A-\sigma I)^{-2} x_0}{\|(A-\sigma I)^{-2} x_0\|_2}, \frac{(A-\sigma I)^{-3} x_0}{\|(A-\sigma I)^{-3} x_0\|_2}, \cdots \tag{3.313}$$

para valores iniciais x_0 e σ. Para o caso onde A é diagonalizável podemos escrever $A = UDU^{-1}$ com D diagonal,

$$D = \begin{bmatrix} \lambda_1 & 0 & 0 & \cdots & 0 \\ 0 & \lambda_2 & 0 & \cdots & 0 \\ 0 & 0 & \lambda_3 & \cdots & 0 \\ & \vdots & & \ddots & \\ 0 & 0 & 0 & \cdots & \lambda_n \end{bmatrix}. \tag{3.314}$$

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

$$x_0 = UU^{-1}x_0 = U[\xi_1 \;\; \xi_2 \;\; \xi_3 \;\; \cdots \;\; \xi_n]^T \tag{3.315}$$

com $\xi_k \neq 0$. Então

$$(A - \sigma I)^{-k}x_0 = (U(D - \sigma I)^{-k}U^{-1})U \begin{bmatrix} \xi_1 \\ \xi_2 \\ \xi_3 \\ \vdots \\ \xi_n \end{bmatrix} \tag{3.316}$$

$$= U \begin{bmatrix} (\lambda_1 - \sigma)^{-k}\xi_1 \\ (\lambda_2 - \sigma)^{-k}\xi_2 \\ (\lambda_3 - \sigma)^{-k}\xi_3 \\ \vdots \\ (\lambda_n - \sigma)^{-k}\xi_n \end{bmatrix} = (\lambda_i - \sigma)^{-k}\xi_1 U \begin{bmatrix} \frac{\xi_1}{\xi_i}\left(\frac{\lambda_i - \sigma}{\lambda_1 - \sigma}\right)^k \\ \cdots \\ 1 \\ \vdots \\ \cdots \\ \frac{\xi_n}{\xi_i}\left(\frac{\lambda_i - \sigma}{\lambda_n - \sigma}\right)^k \end{bmatrix} \tag{3.317}$$

Como $\left|\frac{\lambda_i - \sigma}{\lambda_j - \sigma}\right| < 1$, o último vetor converge para $\pm e_i$ e

$$x_k = \frac{(A - \sigma I)^{-k}x_0}{\|(A - \sigma I)^{-k}x_0\|_2} \tag{3.318}$$

converge para $Ue_i = v_i$ que é um múltiplo do autovetor associado a λ_i. Também, a sequência

$$\lambda_k = x_k^T A x_k \tag{3.319}$$

converge para λ_i:

$$\lim_{k \to \infty} \lambda_k = v_i^T A v_i = \lambda_i v_i^T v_i = \lambda_i. \tag{3.320}$$

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 3.8.1. Calcule o autovalor dominante e o autovetor associado da matriz

$$\begin{bmatrix} 4 & 41 & 78 \\ 48 & 28 & 21 \\ 26 & 13 & 11 \end{bmatrix}. \tag{3.321}$$

Expresse sua resposta com seis dígitos significativos.

E 3.8.2. Calcule o autovalor dominante e o autovetor associado da matriz

$$\begin{bmatrix} 3 & 4 \\ 2 & -1 \end{bmatrix} \tag{3.322}$$

usando o método da potência inciando com o vetor $x = [1 \quad 1]^T$.

E 3.8.3. 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:

$$\|A\|_2 = \sqrt{\max\{|\lambda| : \lambda \in \sigma(A^*A)\}}. \tag{3.323}$$

Use o método da potência para obter a norma L_2 da seguinte matriz:

$$A = \begin{bmatrix} 69 & 84 & 88 \\ 15 & -40 & 11 \\ 70 & 41 & 20 \end{bmatrix}. \tag{3.324}$$

Expresse sua resposta com seis dígitos significativos.

E 3.8.4. Os autovalores de uma matriz triangular são os elementos da diagonal principal. Verifique o método da potência aplicada à seguinte matriz:

$$\begin{bmatrix} 2 & 3 & 1 \\ 0 & 3 & -1 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.325}$$

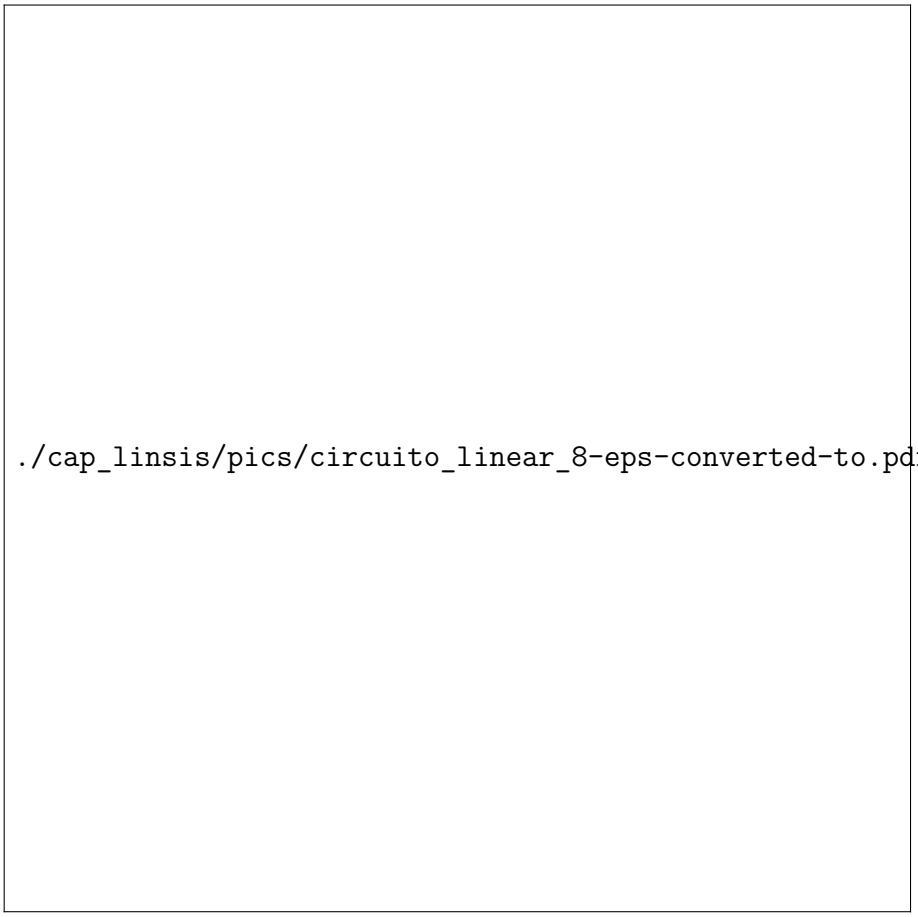
3.9 Exercícios finais

E 3.9.1. O circuito linear da Figura **??** 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 $R_6 = R_7 = R_8 = 100$

$$\begin{aligned} V_1 &= V && (3.326)\\ \frac{V_1 - V_2}{R_1} + \frac{V_3 - V_2}{R_2} - \frac{V_2}{R_5} &= 0 && (3.327)\\ \frac{V_2 - V_3}{R_2} + \frac{V_4 - V_3}{R_3} - \frac{V_3}{R_6} &= 0 && (3.328)\\ \frac{V_3 - V_4}{R_3} + \frac{V_5 - V_4}{R_4} - \frac{V_4}{R_7} &= 0 && (3.329)\\ \frac{V_4 - V_5}{R_4} - \frac{V_5}{R_8} &= 0 && (3.330) \end{aligned}$$



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 3.9.2. Resolva o Problema **??** pelos métodos de Jacobi e Gauss-Seidel.

E 3.9.3. (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 $f(4) = 4$.

Chapter 4

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 **??** 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 $x^{(k)}$:

$$f(x) = f(x^{(k)}) + (x - x^{(k)})f'(x^{(k)}) + O\left(|x - x^{(k)}|^2\right) \tag{4.1}$$

- 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:

$$\begin{aligned}
f_1(x_1,x_2,\ldots,x_n) &= 0 && (4.2)\\
f_2(x_1,x_2,\ldots,x_n) &= 0 && (4.3)\\
&\vdots && (4.4)\\
f_n(x_1,x_2,\ldots,x_n) &= 0 && (4.5)
\end{aligned}$$

Podemos escrever este problema na forma vetorial definindo o vetor $x = [x_1,x_2,\ldots,x_n]^T$

e a função vetorial

$$F(x) = \begin{bmatrix} f_1(x_1,x_2,\ldots,x_n) \\ f_2(x_1,x_2,\ldots,x_n) \\ \vdots \\ f_n(x_1,x_2,\ldots,x_n) \end{bmatrix}. \tag{4.6}$$

Exemplo 4.0.1. Suponha que queiramos resolver numericamente o seguinte sistema de duas equações e duas incógnitas:

$$\frac{x_1^2}{3} + x_2^2 = 1 \tag{4.7}$$

$$x_1^2 + \frac{x_2^2}{4} = 1 \tag{4.8}$$

Então definimos

$$F(x) = \begin{bmatrix} \frac{x_1^2}{3} + x_2^2 - 1 \\ \\ x_1^2 + \frac{x_2^2}{4} - 1 \end{bmatrix} \tag{4.9}$$

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

$$F(x) - F(x^{(0)}) = \begin{bmatrix} f_1(x_1,x_2,\ldots,x_n) - f_1(x_1^{(0)},x_2^{(0)},\ldots,x_n^{(0)}) \\ f_2(x_1,x_2,\ldots,x_n) - f_2(x_1^{(0)},x_2^{(0)},\ldots,x_n^{(0)}) \\ \vdots \\ f_n(x_1,x_2,\ldots,x_n) - f_n(x_1^{(0)},x_2^{(0)},\ldots,x_n^{(0)}) \end{bmatrix} \tag{4.10}$$

Usamos a regra da cadeia

$$df_i = \frac{\partial f_i}{\partial x_1}dx_1 + \frac{\partial f_i}{\partial x_2}dx_2 + \cdots + \frac{\partial f_i}{\partial x_n}dx_n = \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}dx_j \tag{4.11}$$

e aproximamos as diferenças por derivadas parciais:

$$f_i(x_1,x_2,\ldots,x_n) - f_i(x_1^{(0)},x_2^{(0)},\ldots,x_n^{(0)}) \approx \sum_{j=1}^{n} \frac{\partial f_i}{\partial x_j}\left(x_j - x_j^{(0)}\right) \tag{4.12}$$

Portanto,

$$F(x) - F(x^{(0)}) \approx \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \begin{bmatrix} x_1 - x_1^{(0)} \\ x_2 - x_2^{(0)} \\ \vdots \\ x_n - x_n^{(0)} \end{bmatrix}, \tag{4.13}$$

Definimos, então, a matriz jacobiana por

$$J_F = \frac{\partial(f_1, f_2, \ldots, f_n)}{\partial(x_1, x_2, \ldots, x_n)} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}. \tag{4.14}$$

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:

$$(J_F)_{ij} = \frac{\partial f_i}{\partial x_j}. \tag{4.15}$$

Nestes termos, podemos reescrever (**??**) como

$$F(x) \approx F(x^{(0)}) + J_F(x^{(0)})(x - x^{(0)}) \tag{4.16}$$

Esta expressão é chamada de linearização de $F(x)$ no ponto $x^{(0)}$ e generaliza a linearização em uma dimensão dada por $f(x) \approx f(x^{(0)}) + f'(x^{(0)})(x - x^{(0)})$.

4.1 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 $x^{(k)}$:

$$F(x) = F(x^{(k)}) + J_F\left(x^{(k)}\right)\left(x - x^{(k)}\right) + O\left(\|x - x^{(k)}\|^2\right) \tag{4.17}$$

- 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:

$$F(x^{(k)}) + J_F\left(x^{(k)}\right)\left(x^{(k+1)} - x^{(k)}\right) = 0 \tag{4.18}$$

Supondo que a matriz jacobina seja inversível no ponto $x^{(k)}$, temos:

$$J_F\left(x^{(k)}\right)\left(x^{(k+1)} - x^{(k)}\right) = -F(x^{(k)}) \tag{4.19}$$

$$x^{(k+1)} - x^{(k)} = -J_F^{-1}\left(x^{(k)}\right)F(x^{(k)}) \tag{4.20}$$

$$x^{(k+1)} = x^{(k)} - J_F^{-1}\left(x^{(k)}\right)F(x^{(k)}) \tag{4.21}$$

Desta forma, o método iterativo de Newton-Raphson para encontrar as raízes de $F(x) = 0$ é dado por:

$$\begin{cases} x^{(k+1)} = x^{(k)} - J_F^{-1}\left(x^{(k)}\right)F(x^{(k)}), \ \ k > 0 \\ x^{(0)} = \text{dado inicial} \end{cases} \tag{4.22}$$

Observação 4.1.1. 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 4.1.2. A notação $J_F^{-1}\left(x^{(k)}\right)$ enfatiza que a jacobiana deve ser calculada a cada passo.

Observação 4.1.3. Podemos definir o passo $\Delta^{(k)}$ como

$$\Delta^{(k)} = x^{(k+1)} - x^{(k)} \tag{4.23}$$

Assim, $\Delta^{(k)} = -J_F^{-1}\left(x^{(k)}\right) F(x^{(k)})$, ou seja, $\Delta^{(k)}$ resolve o problema linear:

$$J_F\left(x^{(k)}\right)\Delta^{(k)} = -F(x^{(k)}) \tag{4.24}$$

Em geral, é menos custoso resolver o sistema acima do que calcular o inverso da jacobiana e multiplicar pelo vetor $F(x^{(k)})$.

Exemplo 4.1.1. Retornamos ao nosso exemplo inicial, isto é, resolver numericamente o seguinte sistema não linear:

$$\frac{x_1^2}{3} + x_2^2 = 1 \tag{4.25}$$

$$x_1^2 + \frac{x_2^2}{4} = 1 \tag{4.26}$$

Para tal, definimos a função $F(x)$:

$$F(x) = \begin{bmatrix} \frac{x_1^2}{3} + x_2^2 - 1 \\ x_1^2 + \frac{x_2^2}{4} - 1 \end{bmatrix} \tag{4.27}$$

cuja jacobiana é:

$$J_F = \begin{bmatrix} \frac{2x_1}{3} & 2x_2 \\ 2x_1 & \frac{x_2}{2} \end{bmatrix} \tag{4.28}$$

Faremos a implementação numérica no `Scilab`. Para tal definimos as funções que implementarão $F(x)$ e a $J_F(x)$

```
function y=F(x)
    y(1)=x(1)^2/3+x(2)^2-1
    y(2)=x(1)^2+x(2)^2/4-1
endfunction

function y=JF(x)
    y(1,1)=2*x(1)/3
    y(1,2)=2*x(2)
    y(2,1)=2*x(1)
    y(2,2)=x(2)/2
endfunction
```

Alternativamente, estas funções poderiam ser escritas como

```
function y=F(x)
    y=[x(1)^2/3+x(2)^2-1; x(1)^2+x(2)^2/4-1]
endfunction

function y=JF(x)
    y=[2*x(1)/3  2*x(2); 2*x(1) x(2)/2]
endfunction
```

Desta forma, se x é uma aproximação para a raiz, pode-se calcular a próxima aproximação através dos comandos:

```
delta=-JF(x)\F(x)
x=x+delta
```

Ou simplesmente

```
x=x-JF(x)\F(x)
```

Observe que as soluções exatas desse sistema são $\left(\pm\sqrt{\frac{9}{11}}, \pm\sqrt{\frac{8}{11}}\right)$.

Exemplo 4.1.2. Encontre uma aproximação para a solução do sistema

$$x_1^2 = \cos(x_1 x_2) + 1 \tag{4.29}$$
$$\operatorname{sen}(x_2) = 2\cos(x_1) \tag{4.30}$$

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:

$$x^{(n+1)} = x^{(n)} - J_F^{-1}(x)F(x), \quad n \geq 1. \tag{4.31}$$

onde

$$F(x) = \begin{bmatrix} x_1^2 - \cos(x_1 x_2) - 1 \\ \operatorname{sen}(x_2) - 2\cos(x_1) \end{bmatrix} \tag{4.32}$$

e sua jacobiana é

$$J_F(x) = \begin{bmatrix} 2x_1 + x_2 \operatorname{sen}(x_1 x_2) & x_1 \operatorname{sen}(x_1 x_2) \\ 2\operatorname{sen}(x_1) & \cos(x_2) \end{bmatrix} \tag{4.33}$$

As iterações convergem para $x = (1{,}3468109,\ 0{,}4603195)$.

No `Scilab`, podemos implementá-las com o seguinte código:

```
function y=F(x)
    y(1) = x(1)^2-cos(x(1)*x(2))-1
    y(2) = sin(x(2))-2*cos(x(1))
endfunction

function y=JF(x)
    y(1,1) = 2*x(1)+x(2)*sin(x(1)*x(2))
    y(1,2) = x(1)*sin(x(1)*x(2))

    y(2,1) = 2*sin(x(1))
    y(2,2) = cos(x(2))
endfunction
```

E agora, basta iterar:

```
x=[1.5; .5]
x=x-JF(x)\F(x) //(5 vezes)
```

$\Diamond$

4.1.1 Código Scilab: Newton para sistemas

```
function [x] = newton(F,JF,x0,TOL,N)
  x = x0
  k = 1
  //iteracoes
  while (k <= N)
    //iteracao de Newton
    delta = -inv(JF(x))*F(x)
    x = x + delta
    //criterio de parada
    if (norm(delta,'inf')<TOL) then
      return x
    end
    k = k+1
  end
  error('Num. de iter. max. atingido!')
endfunction
```

Exercícios resolvidos

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

Veja como em:

https://github.com/livroscolaborativos/CalculoNumerico

Exercícios

E 4.1.1. Faça o que se pede:

a) Encontre o gradiente da função

$$f(x,y) = x^2y + \cos(xy) - 4 \tag{4.34}$$

b) Encontre a matriz jacobiana associada à função

$$F(x,y) = \begin{bmatrix} x\cos(x) + y \\ e^{-2x+y} \end{bmatrix}. \tag{4.35}$$

c) Encontre a matriz jacobiana associada à função

$$L(x) = \begin{bmatrix} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 - y_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 - y_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 - y_3 \end{bmatrix}. \tag{4.36}$$

E 4.1.2. Encontre uma aproximação numérica para o seguinte problema não linear de três equações e três incógnitas:

$$2x_1 - x_2 = \cos(x_1) \tag{4.39}$$
$$-x_1 + 2x_2 - x_3 = \cos(x_2) \tag{4.40}$$
$$-x_2 + x_3 = \cos(x_3) \tag{4.41}$$

Partindo das seguintes aproximações iniciais:

a) $x^{(0)} = [1,\ 1,\ 1]^T$

b) $x^{(0)} = [-0{,}5,\ -2,\ -3]^T$

Prof. M.e Daniel Cassimiro

c) $x^{(0)} = [-2,\ -3,\ -4]^T$

d) $x^{(0)} = [0,\ 0,\ 0]^T$

E 4.1.3. 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 4.1.4. 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 `Scilab`.

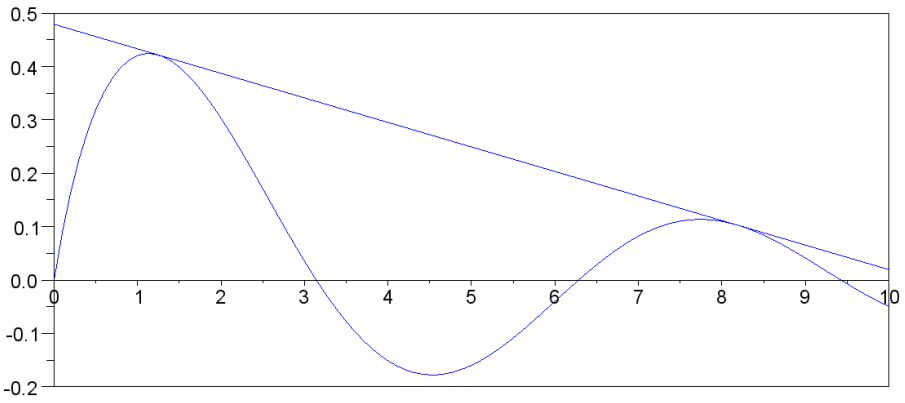


Figure 4.1: Reta bitangente a uma curva.

g) Transforme o sistema em um problema de uma única variável e compare com a resposta do Problema **??**.

E 4.1.5. Encontre uma aproximação com erro inferior a 10^{-5} em cada incógnita para a solução próxima da origem do sistema

$$\begin{aligned} 6x - 2y + e^z &= 2 && (4.42)\\ \operatorname{sen}(x) - y + z &= 0 && (4.43)\\ \operatorname{sen}(x) + 2y + 3z &= 1 && (4.44) \end{aligned}$$

E 4.1.6. (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 4.1.7. 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 **??**.

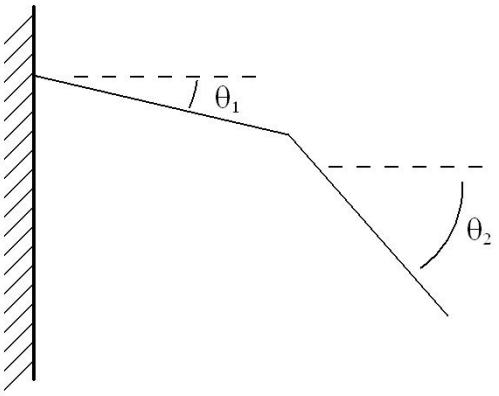


Figure 4.2: Sistema mecânico com dois segmentos.

E 4.1.8. (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 **??**.

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:

$$k\theta_1 = \frac{3PL}{2}\cos\theta_1 + k\left(\theta_2 - \theta_1\right) \tag{4.51}$$

$$k\left(\theta_2 - \theta_1\right) = \frac{PL}{2}\cos\theta_2 \tag{4.52}$$

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 4.1.9. (estática - problemas de três variáveis) Considere, agora, o sistema mecânico semelhante ao do Problema **??**, porém constituído de três segmentos de mesmo comprimento L presos entre si e a uma parede por articulações.

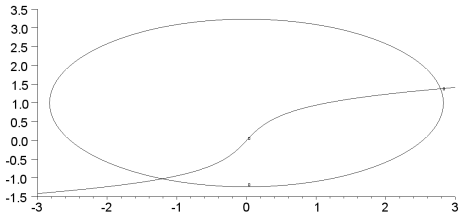


Figure 4.3: intersecção entre duas curvas.

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:

$$k\theta_1 = \frac{5PL}{2}\cos\theta_1 + k(\theta_2 - \theta_1) \tag{4.53}$$

$$k(\theta_2 - \theta_1) = \frac{3PL}{2}\cos\theta_2 + k(\theta_3 - \theta_2) \tag{4.54}$$

$$k(\theta_3 - \theta_2) = \frac{PL}{2}\cos\theta_3 \tag{4.55}$$

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 4.1.10. Considere o problema de encontrar os pontos de intersecção das curvas descritas por (ver Figura **??**):

$$\frac{x^2}{8} + \frac{(y-1)^2}{5} = 1 \tag{4.56}$$

$$\tag{4.57}$$

$$\tan^{-1}(x) + x = y + y^3 \tag{4.58}$$

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 4.1.11. Considere o sistema de equações dado por

$$\frac{(x-3)^2}{16} + \frac{(y-1)^2}{36} = 1 \tag{4.59}$$

$$\tanh(x) + x = 2\,\text{sen}\; y - 0.01y^3 \tag{4.60}$$

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 4.1.12. (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

$$C_1(x) = 10 + .3x + 10^{-4}x^2 + 3.4 \cdot 10^{-9}x^4 \quad (4.61)$$
$$C_2(x) = 50 + .25x + 2 \cdot 10^{-4}x^2 + 4.3 \cdot 10^{-7}x^3 \quad (4.62)$$
$$C_3(x) = 500 + .19x + 5 \cdot 10^{-4}x^2 + 1.1 \cdot 10^{-7}x^4 \quad (4.63)$$

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:

$$C(x_1,x_2,x_3) = C_1(x_1) + C_2(x_2) + C_3(x_3) \quad (4.64)$$

restrita à condição

$$G(x_1,x_2,x_3) = x_1 + x_2 + x_3 - 1500 = 0. \quad (4.65)$$

Pelos multiplicadores de Lagrange, temos que resolver o sistema dado por:

$$\nabla C(x_1,x_2,x_3) = \lambda \nabla G(x_1,x_2,x_3) \quad (4.66)$$
$$G(x_1,x_2,x_3) = 0 \quad (4.67)$$

E 4.1.13. 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

$$R = [3{,}1 - f(0)]^2 + [4{,}4 - f(1)]^2 + [6{,}7 - f(2)]^2 . \quad (4.68)$$

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 4.1.14. Encontre o valor máximo da função

$$f(x,y) = -x^4 - y^6 + 3xy^3 - x \tag{4.69}$$

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 4.1.15. 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 4.1.16. Encontre as raízes do problema

$$\begin{aligned} 3x - \cos(yz + z) - 1/2 &= 0 && (4.70)\\ 4x^2 - 25y^2 + 0.4y + 2 &= 0 && (4.71)\\ e^{-xy} + 2x - 5z &= 10 && (4.72) \end{aligned}$$

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 4.1.17. Considere o seguinte sistema de equações não lineares:

$$\begin{aligned} x_1 - x_2 &= 0\\ -x_{j-1} + 5(x_j + x_j^3) - x_{j+1} &= 10\exp(-j/3), \;\; 2 \leq j \leq 10\\ x_{11} &= 1 \end{aligned} \tag{4.73}$$

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 4.1.18. Considere a função

$$f(x,y) = \frac{e^{-(x-1)^2-(y-2)^2}}{1+x^2+y^2} \tag{4.76}$$

a) Encontre o valor máximo desta função.

b) Usando multiplicadores de Lagrange, encontre o valor máximo desta função restrito à condição

$$(x-1)^2 + (y-2)^2 = 1. \tag{4.77}$$

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.

4.2 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.

4.2.1 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:

$$f(x) = x_1^2 + x_2^2/4 \tag{4.78}$$

Para construirmos a linearização, fixemos uma direção no espaço $\mathbb{R}^n$, ou seja, um vetor v:

$$v = [v_1, v_2, \cdots, v_n]^T \tag{4.79}$$

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

$$x = x^{(0)} + hv \tag{4.80}$$

e definimos a função auxiliar

$$g(h) = f(x^0 + hv). \tag{4.81}$$

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

$$g(h) = g(0) + hg'(0) + O(h^2) \tag{4.82}$$

Observamos que $g(h) = f(x^{(0)} + hv)$ e $g(0) = f(x^{(0)})$. Precisamos calcular $g'(0)$:

$$g'(h) = \frac{d}{dh}g(h) = \frac{d}{dh}f(x^{(0)} + hv). \tag{4.83}$$

Pela regra da cadeia temos:

$$\frac{d}{dh}f(x^{(0)} + hv) = \sum_{j=1}^{n} \frac{\partial f}{\partial x_j}\frac{dx_j}{dh}. \tag{4.84}$$

Observamos que $x_j = x_j^{(0)} + hv_j$, portanto

$$\frac{dx_j}{dh} = v_j \tag{4.85}$$

Assim:

$$\frac{d}{dh}f(x^{(0)} + hv) = \sum_{j=1}^{n} \frac{\partial f}{\partial x_j}v_j. \tag{4.86}$$

Observamos que esta expressão pode ser vista como o produto interno entre o gradiente de f e o vetor v:

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \qquad v = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \tag{4.87}$$

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

$$f(x^{(0)} + hv) = f(x^{(0)}) + h\nabla^T f(x^{(0)})\, v + O(h^2) \tag{4.88}$$

Finalmente, escrevemos $x = x^{(0)} + hv$, ou seja, $hv = x - x^{(0)}$

$$f(x) = f(x^{(0)}) + \nabla^T f(x^{(0)})\,(x - x^{(0)}) + O(\|x - x^{(0)}\|^2) \tag{4.89}$$

Observação 4.2.1. 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 $x^{(0)}$:

$$\nabla^T f(x^{(0)}) = \left[\frac{\partial f\left(x^{(0)}\right)}{\partial x_1}, \ \frac{\partial f\left(x^{(0)}\right)}{\partial x_2}, \ \cdots, \frac{\partial f\left(x^{(0)}\right)}{\partial x_n} \right] \tag{4.90}$$

4.2.2 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 $f_j : \mathbb{R}^n \to \mathbb{R}$:

$$F(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{bmatrix} \tag{4.91}$$

Linearizando cada uma das funções f_j, temos:

$$F(x) = \underbrace{\begin{bmatrix} f_1\left(x^{(0)}\right) + \nabla^T f_1(x^{(0)}) \left(x - x^{(0)}\right) + O(\|x - x^{(0)}\|^2) \\ f_2\left(x^{(0)}\right) + \nabla^T f_2(x^{(0)}) \left(x - x^{(0)}\right) + O(\|x - x^{(0)}\|^2) \\ \vdots \\ f_n\left(x^{(0)}\right) + \nabla^T f_n(x^{(0)}) \left(x - x^{(0)}\right) + O(\|x - x^{(0)}\|^2) \end{bmatrix}}_{\text{Vetor coluna}} \tag{4.92}$$

ou, equivalentemente:

$$F(x) = \underbrace{\begin{bmatrix} f_1\left(x^{(0)}\right) \\ f_2\left(x^{(0)}\right) \\ \vdots \\ f_n\left(x^{(0)}\right) \end{bmatrix}}_{\text{Vetor coluna}} + \underbrace{\begin{bmatrix} \nabla^T f_1(x^{(0)}) \\ \nabla^T f_2(x^{(0)}) \\ \vdots \\ \nabla^T f_n(x^{(0)}) \end{bmatrix}}_{\text{Matriz jacobiana}} \underbrace{\left(x - x^{(0)}\right)}_{\text{Vetor coluna}} + O(\|x - x^{(0)}\|^2) \quad (4.93)$$

Podemos escrever a linearização de $F(x)$ na seguinte forma mais enxuta:

$$F(x) = F\left(x^{(0)}\right) + J_F(x^{(0)})\left(x - x^{(0)}\right) + O\left(\left\|x - x^{(0)}\right\|^2\right) \quad (4.94)$$

A matriz jacobiana J_F é matriz cujas linhas são os gradientes transpostos de f_j, ou seja:

$$J_F = \frac{\partial(f_1, f_2, \ldots, f_n)}{\partial(x_1, x_2, \ldots, x_n)} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \quad (4.95)$$

A matriz jacobiana de uma função ou simplesmente, o jacobiano de uma função $F(x)$ é a matriz formada pelas suas derivadas parciais:

$$(J_F)_{ij} = \frac{\partial f_i}{\partial x_j} \quad (4.96)$$

Exemplo 4.2.1. Calcule a matriz jacobiana da função

$$F(x) = \begin{bmatrix} \frac{x_1^2}{3} + x_2^2 - 1 \\ \\ x_1^2 + \frac{x_2^2}{4} - 1 \end{bmatrix} \tag{4.97}$$

$$J_F = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} = \begin{bmatrix} \frac{2x_1}{3} & 2x_2 \\ \\ 2x_1 & \frac{x_2}{2} \end{bmatrix} \tag{4.98}$$

Chapter 5

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

$$f(x_i) = y_i, \quad i = 1, 2, \ldots, n. \tag{5.1}$$

Chamamos uma tal f de **função interpoladora** dos pontos dados. Ou ainda, dizemos que f interpola os pontos dados.

Exemplo 5.0.1. 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 $\mathcal{F}_{[1,2]}$:

$$\mathcal{F}_{[1,2]} = \{f : [1,2] \to \mathbb{R} \ ; \ [1,2] \ni x \mapsto f(x) = a + bx; \ a,b \in \mathbb{R}\}. \tag{5.2}$$

Para que uma f na família seja a função interpoladora do conjunto de pontos dados, precisamos que

$$\begin{array}{l} a + bx_1 = y_1 \\ a + bx_2 = y_2 \end{array} \quad \text{isto é} \quad \begin{array}{l} a + b = 1 \\ a + 2b = 2 \end{array} \tag{5.3}$$

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 **??**.

Um problema de interpolação cuja família de funções constitui-se de polinômios é chamado de problema de interpolação polinomial.

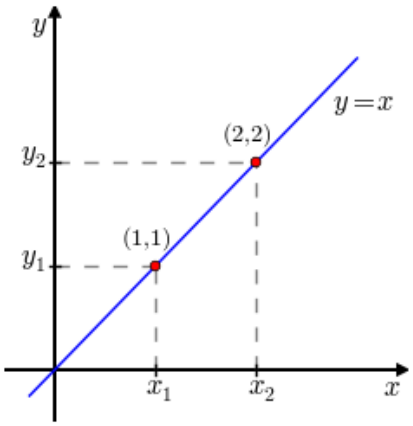


Figure 5.1: Exemplo de interpolação de dois pontos por uma reta, veja o Exemplo **??**.

5.1 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

$$p(x) = \sum_{k=0}^{n} a_k x^k, \tag{5.4}$$

é possível reescrevê-lo como a sequência de operações dada por

$$a_0 + x\left(a_1 + x\left(a_2 + x\left(\ldots + x\left(a_{n-1} + xa_n\right)\ldots\right)\right)\right). \tag{5.5}$$

Também, derivadas e primitivas de polinômios são também polinômios cuja relação algébrica com o original é simples. Além disso, o teorema da aproximação 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 5.1.1 (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 $x \in [a,b]$,*

$$|f(x) - p(x)| < \delta. \tag{5.6}$$

[1]William George Horner, 1786 - 1837, matemático britânico.

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:

$$\mathbb{P}_{n-1} := \left\{ p : x \mapsto p(x) = \sum_{k=0}^{n-1} a_k x^k;\ \{a_0,a_1,\ldots,a_{n-1}\} \in \mathbb{R} \right\}, \tag{5.7}$$

isto é, a família dos polinômios reais de grau menor ou igual a $n-1$.

O Exemplo **??** 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.

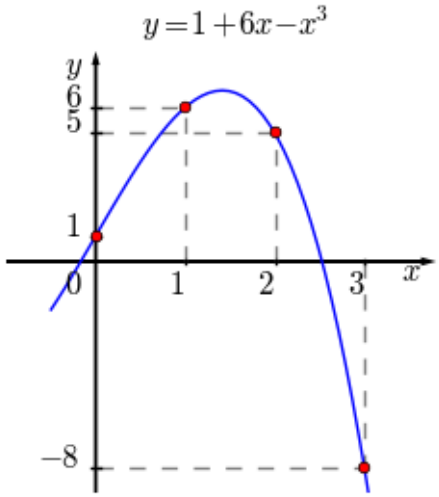


Figure 5.2: Polinômio interpolador do conjunto de pontos $\{(0,1),\ (1,6),\ (2,5),\ (3,-8)\}$. Veja o Exemplo **??**.

Exemplo 5.1.1. 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:

$$p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3. \tag{5.8}$$

As condições de interpolação são $p(x_i) = y_i$, $i = 0, 1, 2, 3$, o que nos leva ao sistema linear:

$$\begin{array}{rcrcrcrcl} a_0 & & & & & & & = & 1 \\ a_0 & + & a_1 & + & a_2 & + & a_3 & = & 6 \\ a_0 & + & 2a_1 & + & 4a_2 & + & 8a_3 & = & 5 \\ a_0 & + & 3a_1 & + & 9a_2 & + & 27a_3 & = & -8 \end{array} \tag{5.9}$$

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 **??**.

No `Scilab`, podemos encontrar o polinômio interpolador e esboçar seu gráfico com os seguintes comandos:

```
-->xi = [0 1 2 3]';
-->yi = [1 6 5 -8]';
-->A = [xi.^0 xi.^1 xi.^2 xi.^3];
-->a = A\yi;
-->p = poly(a,'x','c')
  p  =
                3
     1 + 6x - x
-->xx = linspace(-0.5,3.25);
-->plot(xi,yi,'ro',xx,horner(p,xx),'b-');xgrid
```

◊

Teorema 5.1.2. *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, $p(x_i) = y_i, i = 1, \ldots, n$.*

Proof. Observe que o problema de encontrar os coeficientes a_0, a_1,..., a_{n-1} do polinômio

$$p(x) = a_0 + a_1x + a_2x^2 + \cdots a_{n-1}x^{n-1} = \sum_{k=0}^{n-1} a_kx^k \tag{5.10}$$

tal que $p(x_i) = y_i$ é equivalente a resolver o sistema linear com n equações e n incógnitas dado por

$$\begin{array}{l} a_0 + a_1x_1 + a_1x_1^2 + \cdots + a_{n-1}x_1^{n-1} = y_1, \\ a_0 + a_1x_2 + a_2x_2^2 + \cdots + a_{n-1}x_2^{n-1} = y_2, \\ \qquad\qquad\qquad\qquad\qquad\qquad \vdots \\ a_0 + a_1x_n + a_2x_n^2 + \cdots + a_{n-1}x_n^{n-1} = y_n. \end{array} \tag{5.11}$$

O qual pode ser escrito na forma matricial como

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & \cdots & x_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ \vdots \\ a_{n-1} \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} \tag{5.12}$$

A matriz envolvida é uma **matriz de Vandermonde**[2] de ordem n cujo determinante é dado pelo produtório duplo

$$\prod_{1 \leq i < j \leq n} (x_j - x_i) \tag{5.13}$$

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 **??** e na demonstração do Teorema **??** 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 [**?**]), 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

ER 5.1.1. 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 $\{(1, 1), (2, 2{,}1), (3, 3)\}$.

[2] Alexandre-Théophile Vandermonde, 1735 - 1796, matemático francês.

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:

$$\begin{aligned} a_1 + a_2x_1 + a_3x_1^2 &= y_1 \\ a_1 + a_2x_2 + a_3x_2^2 &= y_2 \end{aligned} \tag{5.14}$$

Sem perda de generalidade, para cada $a_3 \in \mathbb{R}$ dado, temos:

$$\begin{aligned} a_1 + a_2x_1 &= y_1 - a_3x_1^2 \\ a_1 + a_2x_2 &= y_2 - a_3x_2^2, \end{aligned} \tag{5.15}$$

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:

$$\begin{aligned} a_1 + a_2 &= 1 \\ a_1 + 2a_2 &= 2{,}1, \\ a_1 + 3a_2 &= 3 \end{aligned} \tag{5.16}$$

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

Prof. M.e Daniel Cassimiro

E 5.1.1. 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 5.1.2. Encontre o polinômio interpolador para o conjunto de pontos $\{(-1,1{,}25)$, $(0{,}5,0{,}5)$, $(1,1{,}25)$, $(1{,}25,1{,}8125)\}$.

5.2 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

$$\begin{aligned} p(x) = a_1 + a_2(x-x_1) + a_3(x-x_1)(x-x_2) + \cdots \\ + a_n(x-x_1)(x-x_2)\cdots(x-x_{n-1}). \end{aligned} \tag{5.17}$$

Como $p(x_i) = y_i$, $i = 1,2,\ldots,n$, os coeficientes a_i satisfazem o seguinte sistema triangular inferior:

$$\begin{aligned} a_1 &= y_1 \\ a_1 + a_2(x_2-x_1) &= y_2 \\ a_1 + a_2(x_3-x_1) + a_3(x_3-x_1)(x_3-x_2) &= y_3 \\ &\vdots \\ a_1 + a_2(x_n-x_1) + \cdots + a_n(x_n-x_1)\cdots(x_n-x_{n-1}) &= y_n \end{aligned} \tag{5.18}$$

Resolvendo de cima para baixo, obtemos

$$\begin{aligned} a_1 &= y_1 \\ a_2 &= \frac{y_2-a_1}{x_2-x_1} = \frac{y_2-y_1}{x_2-x_1} \\ a_3 &= \frac{y_3-a_2(x_3-x_1)-a_1}{(x_3-x_1)(x_3-x_2)} = \frac{\frac{y_3-y_2}{(x_3-x_2)} - \frac{y_2-y_1}{(x_2-x_1)}}{(x_3-x_1)} \\ &\cdots \end{aligned} \tag{5.19}$$

Note que os coeficientes são obtidos por diferenças das ordenadas divididas por diferenças das abscissas dos pontos dados. Para vermos isso mais claramente,

Table 5.1: 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	$f[x_3] = y_3$		

introduzimos a seguinte notação:

$$f[x_j] := y_j \tag{5.20}$$

$$f[x_j, x_{j+1}] := \frac{f[x_{j+1}] - f[x_j]}{x_{j+1} - x_j} \tag{5.21}$$

$$f[x_j, x_{j+1}, x_{j+2}] := \frac{f[x_{j+1}, x_{j+2}] - f[x_j, x_{j+1}]}{x_{j+2} - x_j} \tag{5.22}$$

$$\vdots \tag{5.23}$$

$$f[x_j, x_{j+1}, \ldots, x_{j+k}] := \frac{f[x_{j+1}, x_{j+2}, \ldots, x_{j+k}] - f[x_j, x_{j+1}, \ldots, x_{j+k-1}]}{x_{j+k} - x_j} \tag{5.24}$$

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 (**??**) nos mostra que

$$a_k = f[x_1,x_2,\ldots,x_k] \tag{5.25}$$

Isto nos permite esquematizar o método conforme apresentado na Tabela **??**.

Exemplo 5.2.1. 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 **??**, 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

$$p(x) = 3 - 2(x+1) + 2(x+1)x + (x+1)x(x-1) \tag{5.26}$$

ou, equivalentemente, $p(x) = x^3 + 2x^2 - x + 1$.

$\Diamond$

5.3 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

$$L_k(x_j) = \begin{cases} 1, & \text{se } k = j \\ 0, & \text{se } k \neq j \end{cases} \tag{5.27}$$

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

$$p(x) = y_1 L_1(x) + y_2 L_2(x) + \cdots + y_n L_n(x) = \sum_{k=1}^{n} y_k L_k(x). \tag{5.28}$$

Para construir os polinômios de Lagrange, podemos analisar a sua forma fatorada, ou seja:

$$L_k(x) = c_k \prod_{\substack{j=1 \\ j \neq k}}^{n} (x - x_j) \tag{5.29}$$

onde o coeficiente c_k é obtido da condição $L_k(x_k) = 1$:

$$L_k(x_k) = c_k \prod_{\substack{j=1 \\ j\neq k}}^{n} (x_k - x_j) \Longrightarrow c_k = \frac{1}{\prod_{\substack{j=1 \\ j\neq k}}^{n} (x_k - x_j)} \tag{5.30}$$

Portanto,

$$L_k(x) = \prod_{\substack{j=1 \\ j\neq k}}^{n} \frac{(x - x_j)}{(x_k - x_j)} \tag{5.31}$$

Observação 5.3.1. O problema de interpolação quando escrito usando como base os polinômios de Lagrange produz um sistema linear diagonal.

Exemplo 5.3.1. 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:

$$L_1(x) = \frac{(x-1)(x-2)(x-3)}{(0-1)(0-2)(0-3)} = -\frac{1}{6}x^3 + x^2 - \frac{11}{6}x + 1 \tag{5.32}$$

$$L_2(x) = \frac{x(x-2)(x-3)}{1(1-2)(1-3)} = \frac{1}{2}x^3 - \frac{5}{2}x^2 + 3x \tag{5.33}$$

$$L_3(x) = \frac{x(x-1)(x-3)}{2(2-1)(2-3)} = -\frac{1}{2}x^3 + 2x^2 - \frac{3}{2}x \tag{5.34}$$

$$L_4(x) = \frac{x(x-1)(x-2)}{3(3-1)(3-2)} = \frac{1}{6}x^3 - \frac{1}{2}x^2 + \frac{1}{3}x \tag{5.35}$$

Assim, temos:

$$P(x) = 0 \cdot L_1(x) + 1 \cdot L_2(x) + 4 \cdot L_3(x) + 9 \cdot L_4(x) = x^2 \tag{5.36}$$

$\Diamond$

5.4 Aproximação de funções reais por polinômios interpoladores

Teorema 5.4.1. *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*

$$f(x) = P(x) + \frac{f^{(n+1)}(\xi(x))}{(n+1)!}(x - x_0)(x - x_1)\cdots(x - x_n), \tag{5.37}$$

onde $P(x)$ é o polinômio interpolador. Em especial, pode-se dizer que

$$|f(x) - P(x)| \leq \frac{M}{(n+1)!} |(x - x_0)(x - x_1) \cdots (x - x_n)|, \tag{5.38}$$

onde

$$M = \max_{x \in [a,b]} |f^{(n+1)}(\xi(x))| \tag{5.39}$$

Exemplo 5.4.1. 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

$$\begin{aligned} P(x) &= 1\frac{(x-\frac{1}{2})(x-1)}{(0-\frac{1}{2})(0-1)} \\ &+ \cos\left(\frac{1}{2}\right)\frac{(x-0)(x-1)}{(\frac{1}{2}-0)(\frac{1}{2}-1)} \\ &+ \cos(1)\frac{(x-0)(x-\frac{1}{2})}{(1-0)(1-\frac{1}{2})} \qquad (5.40) \\ &\approx 1 - 0{,}0299720583066x - 0{,}4297256358252x^2 \qquad (5.41) \end{aligned}$$

No `Scilab`, podemos computar o polinômio interpolador da seguinte forma:

```
L1=poly([.5 1],'x');L1=L1/horner(L1,0)
L2=poly([0 1],'x');L2=L2/horner(L2,0.5)
L3=poly([0 .5],'x');L3=L3/horner(L3,1)
P=L1+cos(.5)*L2+cos(1)*L3
x=[0:.05:1]
plot(x,cos)
plot(x,horner(P,x),'red')
plot(x,horner(P,x)-cos(x))
```

Para estimar o erro máximo, precisamos estimar a derivada terceira de $f(x)$:

$$|f'''(x)| = |\operatorname{sen}(x)| \leq \operatorname{sen}(1) < 0{,}85 \tag{5.42}$$

e, assim,

$$\max_{x \in [0,1]} \left| x\left(x - \frac{1}{2}\right)(x-1) \right|. \tag{5.43}$$

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:

$$\max_{x\in[0,1]} \left| x\left(x - \frac{1}{2}\right)(x-1)\right| \leq \max\{|Q(x_1)|,\, |Q(x_2)|\} \approx 0{,}0481125. \tag{5.44}$$

Portanto:

$$|f(x) - P(x)| < \frac{0{,}85}{3!} 0{,}0481125 \approx 0{,}0068159 < 7 \cdot 10^{-3} \tag{5.45}$$

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 5.4.2. 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

$$\begin{aligned} P(x) &= f(0)\frac{(x-\frac{1}{2})(x-1)}{(0-\frac{1}{2})(0-1)} + f\left(\frac{1}{2}\right)\frac{(x-0)(x-1)}{(\frac{1}{2}-0)(\frac{1}{2}-1)} + f(1)\frac{(x-0)(x-\frac{1}{2})}{(1-0)(1-\frac{1}{2})} && (5.46)\\ &= f(0)(2x^2-3x+1) + f\left(\frac{1}{2}\right)(-4x^2+4x) + f(1)(2x^2-x) && (5.47)\end{aligned}$$

e a integral de $P(x)$ é:

$$\begin{aligned} \int_0^1 P(x)dx &= \left[f(0)\left(\frac{2}{3}x^3 - \frac{3}{2}x^2 + x\right)\right]_0^1 + \left[f\left(\frac{1}{2}\right)\left(-\frac{4}{3}x^3 + 2x^2\right)\right]_0^1 && (5.48)\\ &+ \left[f(1)\left(\frac{2}{3}x^3 - \frac{1}{2}x^2\right)\right]_0^1 && (5.49)\\ &= f(0)\left(\frac{2}{3} - \frac{3}{2} + 1\right) + f\left(\frac{1}{2}\right)\left(-\frac{4}{3} + 2\right) + f(1)\left(\frac{2}{3} - \frac{1}{2}\right) && (5.50)\\ &= \frac{1}{6}f(0) + \frac{2}{3}f\left(\frac{1}{2}\right) + \frac{1}{6}f(1) && (5.51)\end{aligned}$$

Para fazer a estimativa de erro usando o Teorema **??** e temos

$$\begin{aligned}
\left|\int_0^1 f(x)dx - \int_0^1 P(x)dx\right| &= \left|\int_0^1 f(x) - P(x)dx\right| && (5.52)\\
&\leq \int_0^1 |f(x) - P(x)|dx && (5.53)\\
&\leq \frac{M}{6}\int_0^1 \left|x\left(x-\frac{1}{2}\right)(x-1)\right|dx && (5.54)\\
&= \frac{M}{6}\left[\int_0^{1/2} x\left(x-\frac{1}{2}\right)(x-1)dx\right. && (5.55)\\
&- \left.\int_{1/2}^1 x\left(x-\frac{1}{2}\right)(x-1)dx\right] && (5.56)\\
&= \frac{M}{6}\left[\frac{1}{64} - \left(-\frac{1}{64}\right)\right] = \frac{M}{192}. && (5.57)
\end{aligned}$$

Lembramos que $M = \max_{x\in[0,1]} |f'''(x)|$.

Observação 5.4.1. 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 5.4.3. 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

$$\int_0^1 \ln(x+1)dx \approx 0{,}39 \pm \frac{1}{96} \quad (5.58)$$

$$\int_0^1 e^{-x^2}dx \approx 0{,}75 \pm \frac{3{,}87}{192} \quad (5.59)$$

$\Diamond$

Exercícios

E 5.4.1. Use as mesmas técnicas usadas o resultado do Exemplo **??** para obter uma aproximação do valor de:

$$\int_0^1 f(x)dx \quad (5.60)$$

através do polinômio interpolador que coincide com $f(x)$ nos pontos $x = 0$ e $x = 1$.

5.5 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

$$P_i(x) = y_i\frac{(x_{i+1}-x)}{(x_{i+1}-x_i)} + y_{i+1}\frac{(x-x_i)}{(x_{i+1}-x_i)} \tag{5.61}$$

O resultado da interpolação linear segmentada é a seguinte função contínua definida por partes no intervalo $[x_1,x_n]$:

$$f(x) = P_i(x), \quad x \in [x_i,x_{i+1}] \tag{5.62}$$

Exemplo 5.5.1. 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:

$$f(x) = \begin{cases} 0\frac{x-1}{0-1} + 1\frac{x-0}{1-0}, & 0 \leq x < 1 \\ 4\frac{x-2}{1-2} + 3\frac{x-1}{2-1}, & 1 \leq x < 2 \\ 3\frac{x-3}{2-3} + 0\frac{x-2}{3-2}, & 2 \leq x < 3 \\ 0\frac{x-4}{3-4} + 2\frac{x-3}{4-3}, & 3 \leq x < 4 \\ 2\frac{x-5}{4-5} + 0\frac{x-4}{5-4}, & 4 \leq x \leq 5 \end{cases} \tag{5.63}$$

Simplificando, obtemos:

$$f(x) = \begin{cases} x, & 0 \leq x < 1 \\ -x+5, & 1 \leq x < 2 \\ -3x+9, & 2 \leq x < 3 \\ 2x-6, & 3 \leq x < 4 \\ -2x+10, & 4 \leq x \leq 5 \end{cases} \tag{5.64}$$

A Figura **??** é um esboço da função $f(x)$ obtida.

Ela foi gerada no `Scilab` usando os comandos:

```
//pontos fornecidos
xi = [0;1;2;3;4;5]
yi = [0;4;3;0;2;0]
```

```
//numero de pontos
n = 6
//funcao interpoladora
function [y] = f(x)
  for i=1:n-2
    if ((x>=xi(i)) & (x<xi(i+1))) then
      y = yi(i)*(x-xi(i+1))/(xi(i) - xi(i+1)) ...
          + yi(i+1)*(x-xi(i))/(xi(i+1) - xi(i));
    end
  end

  if ((x>=xi(n-1)) & (x<=xi(n))) then
    y = yi(n-1)*(x-xi(n))/(xi(n-1) - xi(n)) ...
        + yi(n)*(x-xi(n-1))/(xi(n) - xi(n-1));
  end
endfunction
//graficando
xx = linspace(xi(1),xi(n),500)';
clear yy
for i=1:max(size(xx))
  yy(i) = f(xx(i))
end
plot(xi,yi,'r.',xx,yy,'b-')
```

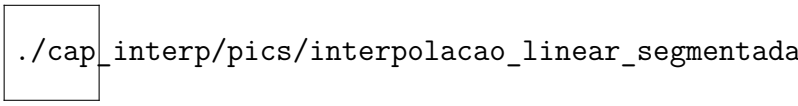


Figure 5.3: Interpolação linear segmentada.

5.6 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 5.6.1 (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

$$\frac{d^4y}{dx^4} = 0, \tag{5.65}$$

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 (**??**) no intervalo (x_j,x_{j+1}) sujeita às condições de contorno

$$y(x_j) = y_j, \quad y'(x_j) = y'_j, \quad y(x_{j+1}) = y_{j+1} \quad \text{e} \quad y'(x_{j+1}) = y'_{j+1}. \tag{5.66}$$

A solução desse problema de contorno é escrita de modo conveniente como

$$s_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3, \tag{5.67}$$

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:

$$a_j = y_j \quad \text{e} \quad b_j = y'_j. \tag{5.68}$$

As duas últimas são obtidas pela solução do sistema de equações formado pelas condições de contorno em x_{j+1}:

$$c_j = 3\frac{y_{j+1} - y_j}{(x_{j+1} - x_j)^2} - \frac{y'_{j+1} + 2y'_j}{x_{j+1} - x_j} \quad \text{e} \quad d_j = -2\frac{y_{j+1} - y_j}{(x_{j+1} - x_j)^3} + \frac{y'_{j+1} + y'_j}{(x_{j+1} - x_j)^2} \tag{5.69}$$

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 5.6.1. *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:*

$$s_j(x) = a_j + b_j(x - x_j) + c_j(x - x_j)^2 + d_j(x - x_j)^3, \tag{5.70}$$

onde

$$\begin{aligned} a_j &= y_j, \quad c_j = 3\frac{y_{j+1} - y_j}{h_j^2} - \frac{y'_{j+1} + 2y'_j}{h_j}, \\ b_j &= y'_j, \quad d_j = -2\frac{y_{j+1} - y_j}{h_j^3} + \frac{y'_{j+1} + y'_j}{h_j^2} \end{aligned} \tag{5.71}$$

e

$$h_j = x_{j+1} - x_j, \quad j = 1,2,\ldots,n-1 \tag{5.72}$$

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 (??), (??) e (??) 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

$$\lim_{x \nearrow x_{j+1}} s''_j(x) = s''_{j+1}(x_{j+1}) \tag{5.73}$$

em todos os pontos internos $j = 1,2,\ldots,n-2$. Em termos dos coeficientes dos polinômios cúbicos (??), a equação anterior assume a forma

$$2c_j + 6d_j h_j = 2c_{j+1}, \quad j = 1,2,\ldots,n-2. \tag{5.74}$$

Esta última equação e (??) permitem construir um sistema de equações lineares para as variáveis y'_j:

Proposição 5.6.2. *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*

$$h_j y'_{j-1} + 2(h_{j-1} + h_j)y'_j + h_{j-1}y'_{j+1} = 3\left(h_j \frac{y_j - y_{j-1}}{h_{j-1}} + h_{j-1}\frac{y_{j+1} - y_j}{h_j}\right), \quad (5.75)$$

onde $j = 2,3,\ldots,n-1$ *e* $h_j = x_{j+1} - x_j$.

O sistema de equações (**??**) é subdeterminado. São n variáveis e $n-2$ equações. A inclusão de duas equações adicionais linearmente independentes das $n-2$ equações (**??**) 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.

5.6.1 Spline natural

Uma forma de definir as duas equações adicionais para completar o sistema (**??**) é impor condições de fronteira livres (ou naturais), ou seja,

$$s''(x_1) = s''(x_n) = 0. \quad (5.76)$$

De acordo com (**??**) essas equações implicam respectivamente

$$c_1 = 0 \quad \text{e} \quad 2c_{n-1} + 6d_{n-1}h_{n-1} = 0, \quad (5.77)$$

ou seja,

$$\begin{cases} 2y'_1 + y'_2 = 3\dfrac{y_2 - y_1}{h_1} \\ \\ y'_{n-1} + 2y'_n = 3\dfrac{y_n - y_{n-1}}{h_{n-1}} \end{cases}. \quad (5.78)$$

Essas duas equações em conjunto com as equações (**??**) formam um sistema de n equações algébricas lineares $Ay' = z$, onde

$$A = \begin{bmatrix} 2 & 1 & 0 & 0 & \cdots & 0 & 0 \\ h_2 & 2(h_1+h_2) & h_1 & 0 & \cdots & 0 & 0 \\ 0 & h_3 & 2(h_2+h_3) & h_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & h_{n-1} & 2(h_{n-1}+h_{n-2}) & h_{n-2} \\ 0 & 0 & 0 & \cdots & 0 & 1 & 2 \end{bmatrix}, \quad (5.79)$$

$$y' = \begin{bmatrix} y'_1 \\ y'_2 \\ \vdots \\ y'_n \end{bmatrix} \quad \text{e} \quad z = 3 \begin{bmatrix} \frac{y_2 - y_1}{h_1} \\ h_2 \frac{y_2 - y_1}{h_1} + h_1 \frac{y_3 - y_2}{h_2} \\ h_3 \frac{y_3 - y_2}{h_2} + h_2 \frac{y_4 - y_3}{h_3} \\ \vdots \\ h_{n-1} \frac{y_{n-1} - y_{n-2}}{h_{n-2}} + h_{n-2} \frac{y_n - y_{n-1}}{h_{n-1}} \\ \frac{y_n - y_{n-1}}{h_{n-1}} \end{bmatrix}. \tag{5.80}$$

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 (**??**).

Exemplo 5.6.1. 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:

$$s(x) = \begin{cases} a_1 + b_1(x-2) + c_1(x-2)^2 + d_1(x-2)^3, & 2 \leq x < 5 \\ a_2 + b_2(x-5) + c_2(x-5)^2 + d_2(x-5)^3, & 5 \leq x < 9 \\ a_3 + b_3(x-9) + c_3(x-9)^2 + d_3(x-9)^3, & 9 \leq x \leq 12 \end{cases}. \tag{5.81}$$

As variáveis y'_1, y'_2, y'_3 e y'_4 resolvem o sistema $Ay' = z$, onde

$$A = \begin{bmatrix} 2 & 1 & 0 & 0 \\ 4 & 2(4+3) & 3 & 0 \\ 0 & 3 & 2(3+4) & 4 \\ 0 & 0 & 1 & 2 \end{bmatrix} = \begin{bmatrix} 2 & 1 & 0 & 0 \\ 4 & 14 & 3 & 0 \\ 0 & 3 & 14 & 4 \\ 0 & 0 & 1 & 2 \end{bmatrix}, \tag{5.82}$$

$$y = \begin{bmatrix} y'_1 \\ y'_2 \\ y'_3 \\ y'_4 \end{bmatrix} \quad \text{e} \quad z = 3 \begin{bmatrix} \frac{1}{3}(-1{,}9 - 4{,}5) \\ \frac{4}{3}(-1{,}9 - 4{,}5) + \frac{3}{4}(0{,}5 - (-1{,}9)) \\ \frac{3}{4}(0{,}5 - (-1{,}9)) + \frac{4}{3}(-0.5 - (0{,}5)) \\ \frac{1}{3}(-0{,}5 - (0.5)) \end{bmatrix} = \begin{bmatrix} -6{,}4 \\ -20{,}2 \\ 1{,}4 \\ -1 \end{bmatrix}. \tag{5.83}$$

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 (**??**):

$$\begin{array}{lllllll} a_1 & = & y_1 = 4{,}5, & & b_1 & = & y_1' = -2{,}8\bar{3}, \\ a_2 & = & y_2 = -1{,}9, & & b_2 & = & y_2' = -0{,}7\bar{3}, \\ a_3 & = & y_3 = 0{,}5. & & b_3 & = & y_3' = 0{,}4\bar{6}, \end{array} \tag{5.84}$$

$$\begin{array}{lllllll} c_1 & = & 0, & & d_1 & = & 0{,}0\bar{7}, \\ c_2 & = & 0{,}7, & & d_2 & = & -0{,}091\bar{6}, \\ c_3 & = & -0{,}4, & & d_3 & = & 0{,}0\bar{4}. \end{array}$$

Portanto:

$$S(x) = \begin{cases} 4{,}5 - 2{,}8\bar{3}(x-2) + 0{,}0\bar{7}(x-2)^3 & , 2 \leq x < 5 \\ -1{,}9 - 0{,}7\bar{3}(x-5) + 0{,}7(x-5)^2 - 0{,}091\bar{6}(x-5)^3 & , 5 \leq x < 9 \\ 0{,}5 + 0{,}4\bar{6}(x-9) - 0{,}4(x-9)^2 + 0{,}0\bar{4}(x-9)^3 & , 9 \leq x \leq 12 \end{cases} . \tag{5.85}$$

No `Scilab`, podemos utilizar:

```
xi = [2;5;9;12]
yi = [4.5;-1.9;0.5;-0.5]
hi = xi(2:4)-xi(1:3)
A = [2 1 0 0;hi(2) 2*(hi(1)+hi(2)) hi(1) 0; ...
   0 hi(3) 2*(hi(2)+hi(3)) hi(2);0 0 1 2 ]
z = 3*[(yi(2)-yi(1))/hi(1); ...
   hi(2)/hi(1)*(yi(2)-yi(1))+hi(1)/hi(2)*(yi(3)-yi(2));...
   hi(3)/hi(2)*(yi(3)-yi(2))+hi(2)/hi(3)*(yi(4)-yi(3));...
   (yi(4)-yi(3))/hi(3)]
dyi = A\z
a=yi(1:3)
b=dyi(1:3)
c(1)=0
c(2:3)=3*(yi(3:4)-yi(2:3))./hi(2:3).^2 ...
         - (dyi(3:4)+2*dyi(2:3))./hi(2:3)
d=-2*(yi(2:4)-yi(1:3))./hi.^3 + (dyi(2:4)+dyi(1:3))./hi.^2
for i=1:3
    P(i) = poly([a(i) b(i) c(i) d(i)],'x','coeff')
    z = [xi(i):.01:xi(i+1)]
    plot(z,horner(P(i),z-xi(i)))
end
```

O mesmo resultado é obtido através das instruções `splin` e `interp` do `Scilab`:

```
xi = [2;5;9;12]
yi = [4.5;-1.9;0.5;-0.5]
dyi=splin(xi,yi,'natural')
z=linspace(xi(1),xi(($))
plot(z,interp(z,xi,yi,dyi))
```

$\Diamond$

5.6.2 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 (**??**) formam um sistema de $n-2$ equações $Ay' = z$, onde

$$A = \begin{bmatrix} 2(h_1+h_2) & h_1 & 0 & 0 & \cdots & 0 & 0 \\ h_3 & 2(h_2+h_3) & h_2 & 0 & \cdots & 0 & 0 \\ 0 & h_4 & 2(h_3+h_4) & h_3 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & h_{n-2} & 2(h_{n-3}+h_{n-2}) & h_{n-3} \\ 0 & 0 & 0 & \cdots & 0 & h_{n-1} & 2(h_{n-2}+h_{n-1}) \end{bmatrix}, \tag{5.86}$$

$$y' = \begin{bmatrix} y'_2 \\ y'_3 \\ \vdots \\ y'_{n-1} \end{bmatrix} \quad \text{e} \quad z = 3\begin{bmatrix} h_2\frac{y_2-y_1}{h_1} + h_1\frac{y_3-y_2}{h_2} - h_2y'_1 \\ h_3\frac{y_3-y_2}{h_2} + h_2\frac{y_4-y_3}{h_3} \\ \vdots \\ h_{n-2}\frac{y_{n-2}-y_{n-3}}{h_{n-3}} + h_{n-3}\frac{y_{n-1}-y_{n-2}}{h_{n-2}} \\ h_{n-1}\frac{y_{n-1}-y_{n-2}}{h_{n-2}} + h_{n-2}\frac{y_n-y_{n-1}}{h_{n-1}} - h_{n-2}y'_n \end{bmatrix}. \tag{5.87}$$

Observe que a matriz A é diagonal dominante estrita e, portanto, o sistema $Ay' = z$ possui solução única.

5.6.3 Spline *not-a-knot*

O spline *not-a-knot* é definido com um spline cúbico que satisfaz as equações adicionais

$$\lim_{x \nearrow x_2} s'''_1(x) = s'''_2(x_2) \quad \text{e} \quad \lim_{x \nearrow x_{n-1}} s'''_{n-2}(x) = s'''_{n-1}(x_{n-1}). \tag{5.88}$$

Em termos dos coeficientes (**??**), as equações anteriores correspondem a

$$d_1 = d_2 \quad \text{e} \quad d_{n-2} = d_{n-1}, \tag{5.89}$$

ou seja,

$$\begin{cases} h_2^2 y_1' + (h_2^2 - h_1^2) y_2' - h_1^2 y_3' = 2\left(h_2^2 \dfrac{y_2 - y_1}{h_1} - h_1^2 \dfrac{y_3 - y_2}{h_2}\right) \\ \\ h_{n-1}^2 y_{n-2}' + (h_{n-1}^2 - h_{n-2}^2) y_{n-1}' - h_{n-2}^2 y_n' = 2\left(h_{n-1}^2 \dfrac{y_{n-1} - y_{n-2}}{h_{n-2}} - h_{n-2}^2 \dfrac{y_n - y_{n-1}}{h_{n-1}}\right) \end{cases}. \tag{5.90}$$

Essas duas equações agregadas às equações (**??**) formam um sistema de n equações $Ay' = z$, onde

$$A = \begin{bmatrix} h_2^2 & h_2^2 - h_1^2 & -h_1^2 & 0 & \cdots & 0 & 0 \\ h_2 & 2(h_1 + h_2) & h_1 & 0 & \cdots & 0 & 0 \\ 0 & h_3 & 2(h_2 + h_3) & h_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & h_{n-1} & 2(h_{n-2} + h_{n-1}) & h_{n-2} \\ 0 & 0 & 0 & \cdots & h_{n-1}^2 & h_{n-1}^2 - h_{n-2}^2 & -h_{n-2}^2 \end{bmatrix}, \tag{5.91}$$

$$y' = \begin{bmatrix} y_1' \\ y_2' \\ \vdots \\ y_n' \end{bmatrix} \quad \text{e} \quad z = \begin{bmatrix} 2\left(h_2^2 \dfrac{y_2 - y_1}{h_1} - h_1^2 \dfrac{y_3 - y_2}{h_2}\right) \\ 3\left(h_2 \frac{y_2 - y_1}{h_1} + h_1 \frac{y_3 - y_2}{h_2}\right) \\ \vdots \\ 3\left(h_{n-1} \frac{y_{n-1} - y_{n-2}}{h_{n-2}} + h_{n-2} \frac{y_n - y_{n-1}}{h_{n-1}}\right) \\ 2\left(h_{n-1}^2 \dfrac{y_{n-1} - y_{n-2}}{h_{n-2}} - h_{n-2}^2 \dfrac{y_n - y_{n-1}}{h_{n-1}}\right) \end{bmatrix}. \tag{5.92}$$

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.

5.6.4 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

$$s_1'(x_1) = s_{n-1}'(x_n) \quad \text{e} \quad s_1''(x_1) = s_{n-1}''(x_n). \tag{5.93}$$

Em termos dos coeficientes (**??**)

$$b_1 = b_{n-1} \quad \text{e} \quad 2c_1 = 2c_{n-1} + 6d_{n-1}h_{n-1}, \tag{5.94}$$

ou seja,

$$\begin{cases} y_1' - y_n' = 0 \\ \\ 2h_{n-1}y_1' + h_{n-1}y_2' + h_1y_{n-1}' + 2h_1y_n' = 3\left(h_{n-1}\dfrac{y_2 - y_1}{h_1} + h_1\dfrac{y_n - y_{n-1}}{h_{n-1}}\right) \end{cases}. \tag{5.95}$$

Essas duas equações agregadas às equações (**??**) formam um sistema de n equações $Ay' = z$, onde

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & -1 \\ h_2 & 2(h_1+h_2) & h_1 & 0 & \cdots & 0 & 0 \\ 0 & h_3 & 2(h_2+h_3) & h_2 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & h_{n-1} & 2(h_{n-2}+h_{n-1}) & h_{n-2} \\ 2h_{n-1} & h_{n-1} & 0 & \cdots & 0 & h_1 & 2h_1 \end{bmatrix}, \tag{5.96}$$

$$y' = \begin{bmatrix} y_1' \\ y_2' \\ \vdots \\ y_n' \end{bmatrix} \quad \text{e} \quad z = 3\begin{bmatrix} 0 \\ h_2\frac{y_2-y_1}{h_1} + h_1\frac{y_3-y_2}{h_2} \\ \vdots \\ h_{n-1}\frac{y_{n-1}-y_{n-2}}{h_{n-2}} + h_{n-2}\frac{y_n-y_{n-1}}{h_{n-1}} \\ h_{n-1}\frac{y_2-y_1}{h_1} + h_1\frac{y_n-y_{n-1}}{h_{n-1}} \end{bmatrix}. \tag{5.97}$$

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.

Appendix A

Rápida introdução à linguagem Julia

Neste apêndice, abordaremos a linguagem computacional `Julia` que nos auxiliará no processo de obtenção de resultados de operações e computações utilizando o computador.

A.1 Sobre a linguagem Julia

A linguagem de programação Julia é uma linguagem moderna e poderosa que foi criada para atender aos requisitos da computação numérica de alto desempenho e científica. Seus criadores quiseram reunir na linguagem as principais (melhores) características de outras linguagens:

- Ruby;
- matlab;
- R;
- Julia;
- C.

Julia é uma linguagem de programação de alto nível, interpretada e multi-paradigma.

Para mais informações, consulte:

- Página oficial da linguagem Julia: `https://julialang.org/`
- Manual Julia: `https://docs.julialang.org/en/v1/manual/getting-started/`

- GitHub do Professor Daniel: https://github.com/Daniel-C-Fernandes/Numerico/blob/main/Mini-curso-Julia.ipynb
- Julia for Data science: https://juliadatascience.io/pt/
- Julia Academy: https://juliaacademy.com/
- GitHub Julia: https://github.com/JuliaLang/julia
- Introdução: https://julia.vento.eng.br/
- Curso Julia USP: https://edisciplinas.usp.br/pluginfile.php/7879038/mod_resource/content/4/Tutorial%20para%20a%20Linguagem%20Julia.pdf
- Tutoriais Julia: https://julialang.org/learning/tutorials/

Para saber mais: https://www.youtube.com/watch?v=Xzdg9czOxD8&t=29s

A.1.1 Características Principais

- **Tipagem Dinâmica:** Suas variáveis podem receber qualquer tipo de dado, e sua sintaxe se aproxima mais da linguagem humana do que da linguagem de máquina.
- **Multiparadigma:** Suporta diversos paradigmas de programação, como orientação a objetos e programação funcional.
- **Alto Nível:** Possui uma sintaxe expressiva e amigável.
- **Gratuita e Open Source:** Julia é distribuída sob a licença MIT.
- **Suporte a Unicode e UTF-8:** Permite o uso de símbolos matemáticos durante a escrita de programas. Gerenciador de Pacotes Prático: Facilita a instalação e atualização de pacotes.

Aplicações:

- **Data Science:** Julia é usada para análise de dados e descoberta de conhecimento a partir de grandes conjuntos de dados. Machine Learning: Possui pacotes poderosos para criação de modelos de aprendizado de máquina.
- **Computação Científica:** Ideal para construir modelos matemáticos e soluções numéricas.

- **Desenvolvimento Geral:** Pode ser aplicada no desenvolvimento de aplicações web, desktop e outras áreas. Em resumo, Julia é uma linguagem versátil que combina alta performance com uma sintaxe amigável, tornando-a uma escolha popular entre cientistas de dados, desenvolvedores e entusiastas da programação.

Visão geral sobre linguagens de programação: Qual a melhor linguagem de programação? Veja https://www.youtube.com/watch?v=DjUB-yVWT2A.

Para iniciantes, recomendamos o curso EAD gratuito no site Goggo dot jl:

https://www.youtube.com/watch?v=0oChN11wf_4

A.1.2 Instalação e execução

Para versões `Linux Ubuntu` ou `Debian`, basta utilizar o comando a seguir e esperar a instalação:

```
> sudo apt install julia -y
```

Para versões `Windows`, utilize o Prompt de comando `PowerShell` para inserir o comando de instalação a seguir:

```
> winget install julia -s msstore
```

Execute o modo interativo da linguagem `Julia` simplesmente digitando o nome da linguagem seguida de Enter:

```
> julia
```

Dessa forma, abre-se o interpretador interativo da linguagem `Julia`. Pode-se então, como primeira interação, realizar uma operação matemática:

```
julia> 2 + 3
5
```

No site oficial da linguagem Julia estão disponíveis para *download* os interpretadores para os principais sistemas operacionais, `Linux`, `Mac OS` e `Windows`.

Além disso, no GitHub do professor Daniel, há um tutorial de instalação para a versão Windows do intepretador de linguagem `Julia` juntamente com o ambiente `Jupyter Notebook` e uma introdução à linguagem de programação com exemplos.

A.1.3 Usando Julia

O uso da linguagem `Julia` pode ser feito de três formas básicas:

- usando um **console Julia** de modo interativo;

- executando um código `codigo.jl` no console `Julia`;
- executando um código `Julia codigo.jl` diretamente no terminal;

Execução no terminal de um código salvo em .jl

Para se executar um código diretamente no terminal de comando do sistema operacional, basta escrevermos o código que desejamos em um arquivo texto de extensão **.jl**.

Dessa forma, por exemplo, salvaremos o seguinte código num arquivo chamado **ola.jl**:

```
1  println("Hello world!!")
```

Após o salvamento, estando o prompt de comando no mesmo diretório do arquivo salvo, basta executarmos o seguinte comando, obtendo-se a resposta que se segue:

```
1 > julia ola.jl
2 Hello world!!
```

Utilização do Console interativo Julia

Para executarmos qualquer comando no console interativo da linguagem `Julia`, iniciaremos o console interativo da seguinte forma:

```
1 > julia
```

Estando o modo interativo rodando no prompt de comando, basta inserirmos o comando desejado e verificar o resultado da saída do comando registrado na linha seguinte do prompt:

```
1 julia> println("Hello world!!")
2 Hello world!!
```

Execução no console de um código salvo em .jl

Para executarmos qualquer um código salvo com extensão **.jl** no console interativo da linguagem `Julia`, iniciaremos o console interativo da seguinte forma:

```
1 > julia
```

Estando o modo interativo rodando no prompt de comando, basta inserirmos o seguinte comando e verificar o resultado da saída do arquivo registrado na linha seguinte do prompt (observe que o arquivo deve estar localizado em ...):

```
julia> include ola.jl
Hello world!!
```

A.2 Elementos da linguagem

`Julia` é uma linguagem de alto nível de tipagem dinâmica, ou seja, uma variável é criada quando um valor é atribuído a ela, não sendo necessário especificar explicitamente cada tipo de variável, `Julia` vai inferir o tipo de cada variável por você. Porém, também é possível especificar a declaração da variável a ser criada, se for desejável.

A.2.1 Variáveis

Variáveis são valores armazenados pelo computador atrelados a um nome específico, para seja possível recuperar ou alterar seu valor posteriormente.

Alguns tipos de variáveis em `Julia`:

- Números inteiros: Int64
- Números reais: Float64
- Matrizes inteiras: Matrix{Int64}
- Matrizes reais: Matrix{Float64}
- Booleanas: Bool
- Strings: String

Por padrão, números são armazenados usando 64 bits, sendo possível aumentar ou reduzir a precisão, utilizando os tipos Int8 ou Int128, por exemplo.

Criamos novas variáveis escrevendo o nome da variável à esquerda e seu valor à direita, e no meio usamos o operador de atribuição =.

Vejamos alguns exemplos:

```
julia> x = 1
1
julia> y = x * 2.0
2.0
```

variaveis com emogi

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 `Julia` é `==`. Veja os seguintes comandos:

```
julia> print(x, "-", y)
1-2.0

julia> typeof(x), typeof(y)
(Int64, Float64)
```

Comentários e continuação de linha de comando são usados como no seguinte exemplo:

```
julia> # Isto é um comentário

julia> x = 1
1

julia> print(x)
1
```

Utilizando Julia como calculadora....

A.3 Repositórios

Tartaruga

A.4 Estruturas de ramificação e repetição

A linguagem `Julia` 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 Julia:

```
#!/usr/bin/env Julia
# -*- coding:  utf-8 -*-
```

```
3
4  i = 2
5  if (i == 1):
6      print("Olá!")
7  elif (i == 2):
8      print("Hallo!")
9  elif (i == 3):
10     print("Hello!")
11 else:
12     print("Ça Va!")
```

Qual é a saída apresentada pelo código? Por quê?

Observamos que, em `Julia`, 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:

```
1  for i in range(6):
2      print(i)
```

Qual é a saída deste código? Por quê?

Exemplo A.4.3. Veja o seguinte código:

```
1  import numpy as np
2  for i in np.arange(1,8,2):
3      print(i)
```

Qual é a saída deste código? Por quê?

Exemplo A.4.4. Veja o seguinte código:

```
1  for i in np.arange(10,0,-3):
2    print(i)
```

O que é mostrado no console do Julia?

Exemplo A.4.5. Veja o seguinte código:

```
1  import numpy as np
2  for i in np.arange(10,1,-3):
3      print(i)
```

O que é mostrado no console do Julia?

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 Julia:

```
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 `Julia` (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 `Julia`:

```
def h(x,y):
    if (x < y):
        return y - x
    else:
        return x - y
```

define a função:

$$h(x,y) = \begin{cases} y - x & , x < y \\ x - y & , x \geq y \end{cases} \tag{A.1}$$

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 $J(x_1,x_2) := \frac{(f_1,f_2)}{(x_1,x_2)}$ da função:

$$\boldsymbol{f}(x_1,x_2) = (x_1^2 + x_2^2,\ \cos(x_1 x_2)). \tag{A.2}$$

A.5.1 Operações matemáticas elementares

Em `Julia`, os operadores matemáticos elementares são os seguintes:

```
+ # adição
- # subtração
* # multiplicação
/ # divisão de a por b
\ # divisão de b por a
% # Resto da divisão euclidiana
÷ # Quociente inteiro da divisão euclidiana (alt + 246)
(ou \div + tab)
^ #potenciação
// # Frações
exp() #potenciação de base e
Log() #Logarítimo Neperiano
Log10() #Logarítimo de base 10
```

A.5.2 Funções e constantes elementares

Várias funções e constantes elementares estão disponíveis no pacote módulo Julia math. Por exemplo:

```
julia> π = pi (\pi + tab)
π = 3.1415926535897...

julia> cos(π)
-1.0

```

```
julia> exp(1)
2.718281828459045

julia> log(exp(1))
1.0
```

Observamos que `log` é a função logaritmo natural, isto é, $f(x) = \ln(x)$, enquanto que a implementação Julia de $f(x) = \log(x)$ é:

```
julia> log10(10)
1.0
```

Veja mais na documentação Julia.

A.5.3 Operadores lógicos

Em `Julia`, o valor lógico verdadeiro é escrito como `True` e o valor lógico falso como `False`. Temos os seguintes operadores lógicos disponíveis:

```
&& # e lógico
|| # ou lógico
! # negação
== # igualdade
!= # diferente
< # menor que
> # maior que
<= # menor ou igual que
>= # maior ou igual que
```

Veja mais em https://acervolima.com/operadores-em-julia/.

A.6 Matrizes

Em `Julia`, temos um ótimo suporte para computação científica com o pacote numpy. Uma matriz $A = [a_{i,j}]_{i,j=1}^{m,n}$ em Julia é definida usando-se a seguinte sintaxe:

```
julia> A = [
          a11  a12  ...  a1n ,
          a21  a22  ...  a2n ,
          ⋮
          am1  am2  ...  amn
          ]
```

Exemplo A.6.1. Defina a matriz:

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

Solução. Em `Julia`, digitamos:

```
julia> A = [1 2 3,
            4 5 6]

julia> print(A)
2×3 Matrix{Int64}:
1 2 3
4 5 6
```

◊

A seguinte lista contém uma série de funções que geram matrizes particulares:

```
eye # matriz identidade
linspace # vetor de elementos linearmente espaçados
ones # matriz cheia de uns
zeros # matriz nula
```

A.6.1 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.6.2. 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 Julia, 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 `Julia` 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.6.3. 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.6.2 Operações matriciais e elemento-a-elemento

Em `Julia` 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.6.4. 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.7 Gráficos

Para criar um esboço do gráfico de uma função de uma variável real $y = f(x)$, podemos usar a biblioteca `Julia` 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.7.1. 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) $4,\overline{6}$.

E 2.1.3. $(101,1)_2$.

E 2.1.4. $(11,1C)_{16}$.

E 2.1.5. a) $(12,3\overline{1})_5$; 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. No `GNU Octave`, temos:

```
>> printf("%1.7e\n", 299792.458)
2.9979458e+04
>> printf("%1.5e\n", 66.2607)
6.62607e+01
>> printf("%1.3e\n", 0.6674)
6.674e-01
>> printf("%1.5e\n", 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 esquerda) tem peso negativo.

E 2.4.2. a) 25186; b) 7453; c) -7453; d) -1.

E 2.4.3. a) 70; b) -72; c) 72.

E 2.4.4. a) 17990; b) -18248; c) 18248.

E 2.4.5. a) 3,75; b) $-5{,}75$.

E 2.4.7. a) 3,75; b) $-5{,}75$.

E 2.4.8. 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.

$$\begin{aligned} 0{,}1x - 0{,}01 &= 12 && (2.84)\\ 0{,}1x &= 12 + 0{,}01 = 12{,}01 && (2.85)\\ x &= 120{,}1 && (2.86) \end{aligned}$$

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:

$$\sqrt{e^{2x}+1}-e^x = \sqrt{e^{2x}+1}-e^x \cdot \frac{\sqrt{e^{2x}+1}+e^x}{\sqrt{e^{2x}+1}+e^x} \tag{2.212}$$

$$= \frac{e^{2x}+1-e^{2x}}{\sqrt{e^{2x}+1}+e^x} = \frac{1}{\sqrt{e^{2x}+1}+e^x} \tag{2.213}$$

e, de forma análoga:

$$\sqrt{e^{2x}+x^2}-e^x = \frac{x^2}{\sqrt{e^{2x}+x^2}+e^x}. \tag{2.214}$$

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. Escrevemos o sistema na forma matricial e resolvemos:

$$\left[\begin{array}{ccc|c} 1 & 1 & 1 & 0 \\ 1 & 0 & 10 & -48 \\ 0 & 10 & 1 & 25 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 1 & 1 & 0 \\ 0 & -1 & 9 & -48 \\ 0 & 10 & 1 & 25 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 1 & 1 & 0 \\ 0 & 10 & 1 & 25 \\ 0 & -1 & 9 & -48 \end{array}\right] \sim \tag{3.34}$$

$$\sim \left[\begin{array}{ccc|c} 1 & 1 & 1 & 0 \\ 0 & 10 & 1 & 25 \\ 0 & 0 & 9.1 & -45.5 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 1 & 1 & 0 \\ 0 & 10 & 1 & 25 \\ 0 & 0 & 1 & -5 \end{array}\right] \sim \tag{3.35}$$

$$\sim \left[\begin{array}{ccc|c} 1 & 1 & 0 & 5 \\ 0 & 10 & 0 & 30 \\ 0 & 0 & 1 & -5 \end{array}\right] \sim \left[\begin{array}{ccc|c} 1 & 1 & 0 & 5 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -5 \end{array}\right] \sim \tag{3.36}$$

$$\sim \left[\begin{array}{ccc|c} 1 & 0 & 0 & 2 \\ 0 & 1 & 0 & 3 \\ 0 & 0 & 1 & -5 \end{array}\right] \tag{3.37}$$

Portanto $x=2$, $y=3$, $z=-5$

E 3.1.5.

a) $x = [4 \;\; 3 \;\; 2]^T$

b) O sistema é equivalente a

$$\begin{array}{lcllcll} \varepsilon x_1 & + & \varepsilon x_2 & + & (1+\varepsilon)x_3 & = & 2 \\ \varepsilon x_1 & + & (1+\varepsilon)x_2 & + & \varepsilon x_3 & = & 3 \\ (1+\varepsilon)x_1 & + & \varepsilon x_2 & + & \varepsilon x_3 & = & 4 \end{array} \tag{3.47}$$

Somando as três equações temos

$$(1+3\varepsilon)(x_1+x_2+x_3)=9 \Longrightarrow x_1+x_2+x_3 = \frac{9}{1+3\varepsilon} \tag{3.48}$$

Subtraímos $\varepsilon(x_1+x_2+x_3)$ da cada equação do sistema original e temos:

$$\begin{array}{l} x_3 = 2-\frac{9\varepsilon}{1+3\varepsilon} \\ x_2 = 3-\frac{9\varepsilon}{1+3\varepsilon} \\ x_1 = 4-\frac{9\varepsilon}{1+3\varepsilon} \end{array} \tag{3.49}$$

Assim, temos:

$$x_\varepsilon = [4 \;\; 3 \;\; 2]^T - \frac{9\varepsilon}{1+3\varepsilon}[1 \;\; 1 \;\; 1]^T \tag{3.50}$$

E 3.1.6. $x = [1.6890368 \quad 1.6890368 \quad 1.5823257 \quad 1.2667776 \quad 0.6333888]^T$

E 3.1.7.

$$\begin{bmatrix} 1 & 1/2 & -1/2 \\ 1/3 & -1/2 & 1/6 \\ -1/3 & 0 & 1/3 \end{bmatrix} \tag{3.55}$$

E 3.5.1.

$$a = (0, 1, 2, 1, 2, 1) \tag{3.145}$$
$$b = (5, 3, 4, 2, 3, 2) \tag{3.146}$$
$$c = (4, 1, 1, 1, 2, 0) \tag{3.147}$$
$$d = (13, 10, 20, 16, 35, 17) \tag{3.148}$$
$$x = (1, 2, 3, 4, 5, 6) \tag{3.149}$$

E 3.5.2.

$$a = (0, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1/2) \tag{3.151}$$
$$b = (1, 5, 5, 5, 5, 5, 5, 5, 5, 5, 1) \tag{3.152}$$
$$c = (-1, -1, -1, -1, -1, -1, -1, -1, -1, -1, 0) \tag{3.153}$$
$$d = (0, \cos(2/10), \cos(3/10), \cos(4/10), \cos(5/10), \tag{3.154}$$
$$\cos(6/10), \cos(7/10), \cos(8/10), \cos(9/10), \cos(1),0) \tag{3.155}$$
$$x = (0{,}324295, 0{,}324295, 0{,}317115, 0{,}305943, 0{,}291539, \tag{3.156}$$
$$0{,}274169, 0{,}253971, 0{,}230846, 0{,}20355, 0{,}165301, 0{,}082650) \tag{3.157}$$

E 3.6.1.
$\lambda = \frac{71 \times 30}{41} \approx 51.95122$, 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 3.6.2.
$k_1(A) = 36$, $k_2(A) = 18{,}26$, $K_\infty(A) = 20{,}8$.

E 3.6.3.
$k_1 = k_\infty = 6888$, $k_2 = \sqrt{26656567}$ e $k_1 = 180$, $k_2 = 128{,}40972$ e $k_\infty = 210$

E 3.6.4.
$\frac{18}{\varepsilon} + 3$. Quando $\varepsilon \to 0+$, a matriz converge para uma matriz singular e o número de condicionamento diverge para $+\infty$.

E 3.6.5.
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 3.6.6. 0,695; 0,292; 0,188; 0,0237; 0,0123; 0,00967

Exemplo de implementação:

```
J=[1:1:10]
x=sin(J/10)
y=J/10
z=y-y.^3/6
e=abs(x-y)./x
f=abs(x-z)./x
norm(e,1)
norm(e,2)
norm(e,'inf')
norm(f,1)
norm(f,2)
norm(f,'inf')
```

E 3.7.1.

```
epsilon=1e-3;

A=[1 -1 0 0 0; -1 2 -1 0 0; 0 -1 (2+epsilon) -1 0; 0 0 -1 2 -1; 0 0 0 1 -1]

v=[1 1 1 1 1]'
xgauss=gauss([A v])

function x=q_Jacobi()
    x0=[0 0 0 0 0]'

    i=0
    controle=0
    while controle<3 & i<1000
    i=i+1

    x(1)=1+x0(2)
    x(2)=(1+x0(3)+x0(1))/2
    x(3)=(1+x0(2)+x0(4))/(2+epsilon)
    x(4)=(1+x0(3)+x0(5))/2
    x(5)=x0(4)-1

    delta=norm(x-x0,2)
    if delta<1e-6 then
        controle=controle+1
    else
        controle=0
    end
    mprintf('i=%d, x1=%f, x5=%f, tol=%.12f\n', i, x(1), x(5), delta)
    x0=x;
    end

endfunction

function x=q_Gauss_Seidel()
    x0=[0 0 0 0 0]'

    i=0
    controle=0
    while controle<3 & i<15000
    i=i+1

    x(1)=1+x0(2)
    x(2)=(1+x0(3)+x(1))/2
    x(3)=(1+x(2)+x0(4))/(2+epsilon)
    x(4)=(1+x(3)+x0(5))/2
    x(5)=x(4)-1

    delta=norm(x-x0,2)
    if delta<1e-2 then
        controle=controle+1
    else
        controle=0
    end
    mprintf('i=%d, x1=%f, x5=%f, tol=%.12f\n',i,x(1),x(5),delta)
    x0=x;
    end

endfunction
```

E 3.7.4.

0,324295, 0,324295, 0,317115, 0,305943, 0,291539, 0,274169, 0,253971, 0,230846, 0,203551, 0,165301, 0,082650
Exemplos de rotinas:

```
function x=jacobi()
    x0=zeros(11,1)
    k=0;
    controle=0;
    while controle<3 & k<1000
        k=k+1
        x(1)=x0(2)
        for j=2:10
        x(j)=(cos(j/10)+x0(j-1)+x0(j+1))/5
        end
        x(11)=x0(10)/2


        delta=norm(x-x0) //norma 2
        if delta<1e-5 then
```

```
            controle=controle+1
        else
            controle=0;
        end
        mprintf('k=%d, x=[%f,%f,%f], tol=%.12f\n',k,x(1),x(2),x(3),delta)
        x0=x;
    end


endfunction

function x=gs()
    x0=zeros(11,1)
    k=0;
    controle=0;
    while controle<3 & k<1000
        k=k+1
        x(1)=x0(2)
        for j=2:10
        x(j)=(cos(j/10)+x(j-1)+x0(j+1))/5
        end
        x(11)=x0(10)/2

        delta=norm(x-x0) //norma 2
        if delta<1e-5 then
            controle=controle+1
        else
            controle=0;
        end
        mprintf('k=%d, x=[%f,%f,%f], tol=%.12f\n',k,x(1),x(2),x(3),delta)
        x0=x;
    end
endfunction
```

E 3.7.5.
Permute as linhas 1 e 2.

E 3.8.1. $\lambda = 86.1785$ associado ao autovetor dado por $v_1 = [0.65968 \quad 0.66834 \quad 0.34372]^T$.

E 3.8.2.

Este exercício está sem resposta sugerida. Proponha uma resposta. Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

E 3.8.3. 158,726

E 3.8.4.

Este exercício está sem resposta sugerida. Proponha uma resposta. Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

E 3.9.1. a)$V_5 = 98.44V$ b) $V_5 = 103.4V$
O problema com cinco incógnitas pode ser escrito na forma matricial conforme a seguir:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ \frac{1}{R_1} & -\left(\frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_5}\right) & \frac{1}{R_2} & 0 & 0 \\ 0 & \frac{1}{R_2} & -\left(\frac{1}{R_2}+\frac{1}{R_3}+\frac{1}{R_6}\right) & \frac{1}{R_3} & 0 \\ 0 & 0 & \frac{1}{R_3} & -\left(\frac{1}{R_3}+\frac{1}{R_4}+\frac{1}{R_7}\right) & \frac{1}{R_4} \\ 0 & 0 & 0 & \frac{1}{R_4} & -\left(\frac{1}{R_4}+\frac{1}{R_8}\right) \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ v_4 \\ V_5 \end{bmatrix} = \begin{bmatrix} V \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.331}$$

Este problema pode ser implementado no `Scilab` (para o item a) com o seguinte código:

```
R1=2, R2=2, R3=2, R4=2, R5=100, R6=100, R7=100, R8=50, V=127

A=[1      0                  0                  0                 0;
   1/R1  -(1/R1+1/R2+1/R5)   1/R2               0                 0;
   0      1/R2              -(1/R2+1/R3+1/R6)   1/R3              0;
   0      0                  1/R3              -(1/R3+1/R4+1/R7)  1/R4;
   0      0                  0                  1/R4             -(1/R4+1/R8)]
v=[V; 0; 0; 0; 0]
y=A\v
```

O problema com quatro incógnitas pode ser escrito na forma matricial conforme a seguir:

$$\begin{bmatrix} -\left(\frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_5}\right) & \frac{1}{R_2} & 0 & 0 \\ \frac{1}{R_2} & -\left(\frac{1}{R_2}+\frac{1}{R_3}+\frac{1}{R_6}\right) & \frac{1}{R_3} & 0 \\ 0 & \frac{1}{R_3} & -\left(\frac{1}{R_3}+\frac{1}{R_4}+\frac{1}{R_7}\right) & \frac{1}{R_4} \\ 0 & 0 & \frac{1}{R_4} & -\left(\frac{1}{R_4}+\frac{1}{R_8}\right) \end{bmatrix} \begin{bmatrix} V_2 \\ V_3 \\ v_4 \\ V_5 \end{bmatrix} = \begin{bmatrix} -\frac{V}{R1} \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{3.332}$$

Cuja implementação pode ser feita conforme

```
A=[  -(1/R1+1/R2+1/R5)    1/R2               0                 0;
       1/R2              -(1/R2+1/R3+1/R6)   1/R3              0;
       0                  1/R3              -(1/R3+1/R4+1/R7)  1/R4;
       0                  0                  1/R4             -(1/R4+1/R8)]

v=[-V/R1; 0; 0; 0]
y=A\v
```

E 3.9.3. 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 4.1.1. $\nabla f = [2xy - y\,\mathrm{sen}\,(xy), x^2 - x\,\mathrm{sen}\,(xy)]^T$

$$J_F = \begin{bmatrix} \cos(x) - x\,\mathrm{sen}\,(x) & 1 \\ -2e^{-2x+y} & e^{-2x+y} \end{bmatrix} \tag{4.37}$$

$$(J_L)_{ij} = a_{ij} \tag{4.38}$$

E 4.1.2.

Este exercício está sem resposta sugerida. Proponha uma resposta. Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

E 4.1.3. 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 4.1.4. $(\pm 0.8241323, 0.6791941)$

E 4.1.5. $x \approx 0{,}259751$, $y \approx 0{,}302736$, $z \approx 0{,}045896$

E 4.1.6.

Este exercício está sem resposta sugerida. Proponha uma resposta. Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

E 4.1.7. $y = mx + b$ 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:

$$y = f(x_1) + m(x - x_1) \tag{4.45}$$

onde o coeficiente angular m é dado por

$$m = \frac{f(x_2) - f(x_1)}{x_2 - x_1} \tag{4.46}$$

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.

$$m = f'(x_1) = f'(x_2) \tag{4.47}$$

E sabemos que:

$$f'(x) = \frac{\cos(x)}{1+x} - \frac{\operatorname{sen}(x)}{(1+x)^2}. \tag{4.48}$$

Assim, podemos reescrever o problema como

$$\frac{\cos(x_1)}{1+x_1} - \frac{\operatorname{sen}(x_1)}{(1+x_1)^2} - \frac{\cos(x_2)}{1+x_2} + \frac{\operatorname{sen}(x_2)}{(1+x_2)^2} = 0 \tag{4.49}$$

$$\frac{\cos(x_1)}{1+x_1} - \frac{\operatorname{sen}(x_1)}{(1+x_1)^2} - \frac{f(x_2) - f(x_1)}{x_2 - x_1} = 0 \tag{4.50}$$

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 4.1.8. $(0.1956550; 0.2441719)$, $(0.3694093; 0.4590564)$, $(0.9990712; 1.1865168)$ e $(1.4773606; 1.5552232)$

E 4.1.9. $(0.0449310; 0.0648872; 0.0698750)$, $(0.3981385; 0.5658310; 0.6069019)$,

$(1.1862966; 1.4348545; 1.480127)$

E 4.1.10. $(-1{,}2085435, -1{,}0216674)$ e $(2{,}7871115, 1{,}3807962)$

Exemplo de implementação:

```
function z=f(x,y)
    z=x^2/8+(y-1)^2/5-1
endfunction
function z=g(x,y)
    z=atan(x)+x-y-y^3
endfunction

contour([-3:.1:3],[-2:.1:4],f,[0 0])
contour([-3:.1:3],[-2:.1:4],g,[0 0])

function y=F(x)
    y(1)=f(x(1),x(2))
    y(2)=g(x(1),x(2))
endfunction
function y=JF(x)
    y(1,1)=x(1)/4
    y(1,2)=2*(x(2)-1)/5
    y(2,1)=1/(1+x(1)^2)+1
    y(2,2)=-1-3*x(2)^2
endfunction

//primeiro ponto
//x=[-1.2;-1.0]

//segundo ponto
//x=[2.8;1.4]

x=x-JF(x)\F(x)   // 4 vezes
```

E 4.1.11. 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 $(2{,}8441544, 6{,}9954443)$.

Uma possível implementação é

```
function z=f(x,y)
    z=(x-3)^2/16+(y-1)^2/36-1
endfunction

function z=g(x,y)
    z=atan(x)+x-sin(y)-0.01*y^3
endfunction

contour([-1:.1:7],[-5:.1:7],f,[0 0])
contour([-1:.1:7],[-5:.1:7],g,[0 0])

function y=F(x)
    y(1)=f(x(1),x(2))
    y(2)=g(x(1),x(2))
endfunction
function y=JF(x)
    y(1,1)=(x(1)-3)/8
    y(1,2)=(x(2)-1)/18
    y(2,1)=1/(1+x(1)^2)+1
    y(2,2)=-cos(x(2))-0.03*x(2)^2
endfunction


//primeiro ponto
//x=[-.5;-2.0]

//segundo ponto
//x=[3;7]

x=x-JF(x)\F(x)   // 4 vezes
```

E 4.1.12. $(x_1, x_2, x_3) \approx (453{,}62,\ 901{,}94,\ 144{,}43)$

E 4.1.13. 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 4.1.14. $f(-1{,}1579702, -1{,}2020694) \approx 2.376985$

Um exemplo de implementação no Scilab é:

```
deff('z=f(x,y)','z=-x^4-y^6+3*x*y^3-x')
contour([-2:.01:0],[-2:.01:0],f,[ 0:.2: 3])
deff('z=F(x)','z=[-4*x(1)^3+3*x(2)^3-1;-6*x(2)^5+9*x(1)*x(2)^2]')
deff('z=JF(x)','z=[-12*x(1)^2,9*x(2)^2;9*x(2)^2,-30*x(2)^4+18*x(1)*x(2)]')
x=[-1.2;-1.2]
x=x-JF(x)\F(x)
x=x-JF(x)\F(x)
x=x-JF(x)\F(x)
x=x-JF(x)\F(x)
mprintf('f(%f,%f)=%f',x(1),x(2),f(x(1),x(2)))
```

E 4.1.15.

Este exercício está sem resposta sugerida. Proponha uma resposta. Veja como em:
https://github.com/livroscolaborativos/CalculoNumerico

E 4.1.16. $x \approx 0{,}2982646$, $y \approx -0{,}2990796$, $z \approx -1{,}6620333$ e $x \approx -0{,}0691328$, $y \approx 0{,}2923039$, $z \approx -0{,}8235705$.

E 4.1.17.

$$F(x) = \begin{bmatrix} x_1 - x_2 \\ -x_1 + 5(x_2 + x_2^3) - x_3 - 10\exp(-2/3) \\ -x_2 + 5(x_3 + x_3^3) - x_4 - 10\exp(-3/3) \\ -x_3 + 5(x_4 + x_4^3) - x_5 - 10\exp(-4/3) \\ \vdots \\ -x_9 + 5(x_{10} + x_{10}^3) - x_{11} - 10\exp(-10/3) \\ x_{11} - 1 \end{bmatrix} \tag{4.74}$$

$$J_F(x) = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & \dots & 0 \\ -1 & 5(1+3x_2^2) & -1 & 0 & 0 & \dots & 0 \\ 0 & -1 & 5(1+3x_3^2) & -1 & 0 & \dots & 0 \\ 0 & 0 & -1 & 5(1+3x_4^2) & -1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \tag{4.75}$$

Exemplo de implementação no Scilab:

```
function y=F(x)
    y(1)=x(1)-x(2)
    for j=2:10
        y(j)=-x(j-1)+5*(x(j)+x(j)^3)-x(j+1)-10*exp(-j/3)
    end
    y(11)=x(11)-1
endfunction

function y=JF(x)
    y=zeros(11,11)

    y(1,1)=1
    y(1,2)=-1
    for j=2:10

        y(j,j-1)=-1
        y(j,j)=5*(1+3*x(j)^2)
        y(j,j+1)=-1
    end
    y(11,11)=1
endfunction
```

Resposta final: 0,80447, 0,80447, 0,68686, 0,57124, 0,46535, 0,37061, 0,28883, 0,22433, 0,19443, 0,28667, 1

E 4.1.18. $f(0{,}8108792, 1{,}6217584) \approx 0{,}1950369$ e $f(0{,}5527864, 1{,}1055728) \approx 0{,}1455298$

E 5.1.1. $p(x) = -3 + 2x + 5x^3$.

E 5.1.2. $p(x) = 0{,}25 + x^2$.

E 5.4.1.
$\int_0^1 P(x)dx = \frac{f(0)+f(1)}{2}$, $\frac{1}{12}\max_{x\in[0,1]}|f''(x)|$