Matemática numérica

Pedro H A Konzen

1 de setembro de 2023

Licença

Prefácio

Nestas notas de aula são abordados temas introdutórios de matemática numérica. Como ferramenta computacional de apoio didático, apresentam-se códigos em `GNU Octave` (compatíveis com `MATLAB`).

Agradeço a todos e todas que de modo assíduo ou esporádico contribuem com correções, sugestões e críticas. :)

Pedro H A Konzen

Conteúdo

Capítulo 1

Aritmética de Máquina

No `GNU Octave`, temos

```
>> 0.1+0.2==0.3
ans = 0
```

1.1 Sistema de numeração posicional

Cotidianamente, usamos o sistema de numeração posicional na base decimal. Por exemplo, temos

$$123{,}5 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 5 \times 10^{-1}, \tag{1.1}$$

onde o algarismo/dígito 1 está na posição 2 (posição das centenas), o dígito 2 está na posição 1 (posição das dezenas) e o dígito 3 está na posição 0 (posição das unidades). Mais geralmente, temos a representação decimal

$$\pm d_n \dots d_2 d_1 d_0{,}d_{-1}d_{-2}d_{-3}\dots := \tag{1.2}$$

$$\pm \left(d_n \times 10^n + \dots + d_2 \times 10^2 + d_1 \times 10^1 + d_0 \times 10^0 \right. \tag{1.3}$$

$$\left. + d_{-1} \times 10^{-1} + d_{-2} \times 10^{-2} + d_{-3} \times 10^{-3} + \cdots \right), \tag{1.4}$$

cujos os dígitos $d_i \in \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, $i = n, \dots, 2, 1, 0, -1, -2, -3, \dots$. Observamos que esta representação posicional pode ser imediatamente generalizada para outras bases numéricas.

Definição 1.1.1. (Representação posicional) Dada uma base $b \in \mathbb{N} \setminus \{0\}$, definimos a representação

$$\pm(d_n \dots d_2 d_1 d_0, d_{-1} d_{-2} d_{-3} \dots)_b := \tag{1.5}$$
$$\pm \left(d_n \times b^n + \dots + d_2 \times b^2 + d_1 \times b^1 + d_0 \times b^0 \right. \tag{1.6}$$
$$\left. + d_{-1} \times b^{-1} + d_{-2} \times b^{-2} + d_{-3} \times b^{-3} + \dots \right), \tag{1.7}$$

onde os dígitos $d_i \in \{0, 1, \dots, b-1\}$[1], $i = n, \dots, 2, 1, 0, -1, -2, -3, \dots$.

Exemplo 1.1.1. (Representação binária) O número $(11010{,}101)_2$ está escrito na representação binária (base $b = 2$). Da Definição 1.1.1, temos

$$(\overset{4}{1}\ \overset{3}{1}\ \overset{2}{0}\ \overset{1}{1}\ \overset{0}{0}\ ,\ \overset{-1}{1}\ \overset{-2}{0}\ \overset{-3}{1})_2 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 \tag{1.8}$$
$$+ 1 \times 2^{-1} + 0 \times 2^{-2} + 1 \times 2^{-3} \tag{1.9}$$
$$= 26{,}625. \tag{1.10}$$

Podemos fazer estas contas no `GNU Octave` da seguinte forma

```
>> 1*2^4+1*2^3+0*2^2+1*2^1+0*2^0+1*2^-1+0*2^-2+1*2^-3
ans =  26.625
```

1.1.1 Mudança de base

Um mesmo número pode ser representado em diferentes bases e, aqui, estudaremos como obter a representação de uma número em diferentes bases. A mudança de base de representação de um dado número pode ser feita de várias formas. De forma geral, se temos um número x representado na base b_1 e queremos obter sua representação na base b_2, fazemos

1. Calculamos a representação do número x na base decimal.
2. Da calculada representação decimal, calculamos a representação de x na base b_2.

Observamos que o passo 1. ($b \to 10$) segue imediatamente da Definição 1.1.1. Agora, o passo 2. ($10 \to b$), podemos usar o seguinte procedimento.

[1] Para bases $b \geq 11$, usamos a representação dos dígitos maiores ou iguais a 10 por letras maiúsculas do alfabeto latino, i.e. $A = 10$, $B = 11$, $C = 12$ e assim por diante.

Suponhamos que x tenha a seguinte representação decimal

$$d_n d_{n-1} d_{n-2} \ldots d_0, d_{-1} d_{-2} d_{-3} \ldots \tag{1.11}$$

Então, separamos sua parte inteira $I = d_n d_{n-1} d_{n-2} \ldots d_0$ e sua parte fracionária $F = 0,d_{-1} d_{-2} d_{-3} \ldots$ ($x = I + F$). Então, usando de sucessivas divisões de I pela base b desejada, obtemos sua representação nesta mesma base. Analogamente, usando de sucessivas multiplicações de F pela base b, obtemos sua representação nesta base. Por fim, basta somar as representações calculadas.

Exemplo 1.1.2. Obtenha a representação em base quartenária ($b = 4$) do número $(11010{,}101)_2$.

1. $b = 2 \rightarrow b = 10$. A representação de $(11010{,}101)_2$ segue direto da Definição 1.1.1 (veja, o Exemplo 1.1.1). Ou seja, temos

$$(\overset{4}{1}\overset{3}{1}\overset{2}{0}\overset{1}{1}\overset{0}{0}, \overset{-1}{1}\overset{-2}{0}\overset{-3}{1})_2 = 2^4 + 2^3 + 2^1 + 2^{-1} + 2^{-3} \tag{1.12}$$

$$= 26{,}625. \tag{1.13}$$

No `GNU Octave` podemos fazer a mudança para a base decimal com a função `base2dec`:

```
>> I = base2dec("11010",2)
I =  26
>> F = base2dec("101",2)*2^-3
F =  0.62500
>> I+F
ans =  26.625
```

2. $b = 10 \rightarrow b = 4$.

Primeiramente, decompomos 26,625 em sua parte inteira $I = 26$ e em sua parte fracionária 0,625. Então, ao fazermos sucessivas divisões de I por $b = 4$, obtemos:

$$I = 26 \tag{1.14}$$

$$= 6 \times 4 + 2 \times 4^0 \tag{1.15}$$

$$= (1 \times 4 + 2) \times 4 + 2 \times 4^0 \tag{1.16}$$

$$= 1 \times 4^2 + 2 \times 4 + 2 \times 4^0 \tag{1.17}$$

$$= (122)_4. \tag{1.18}$$

Agora, para a parte fracionária, usamos sucessivas multiplicações de F por $b = 4$, obtendo:

$$\begin{aligned} F &= 0{,}625 && (1.19)\\ &= 2{,}5 \times 4^{-1} = 2 \times 4^{-1} + 0{,}5 \times 4^{-1} && (1.20)\\ &= 2 \times 4^{-1} + 2 \times 4^{-1} \times 4^{-1} && (1.21)\\ &= 2 \times 4^{-1} + 2 \times 4^{-2} && (1.22)\\ &= (0{,}22)_4. && (1.23) \end{aligned}$$

No `GNU Octave`, podemos computar a representação de F na base $b = 4$ da seguinte forma:

```
>> F=0.625
F =  0.62500
>> d=fix(F*4),F=F*4-d
d =  2
F =  0.50000
>> d=fix(F*4),F=F*4-d
d =  2
F = 0
```

Por fim, dos passos 1. e 2., temos $(11010{,}101)_2 = (122{,}22)_4$.

Exercícios

Exercício 1.1.1. Obtenha a representação decimal dos seguinte números:

a) $(101101{,}00101)_2$

b) $(23{,}1)_4$

c) $(DAAD)_{16}$

d) $(0{,}1)_3$

e) $(0{,}\overline{1})_4$

Exercício 1.1.2. Obtenha a representação dos seguintes números na base indicada:

a) 45,5 na base $b = 2$.

b) 0,3 na base $b = 4$.

Exercício 1.1.3. Obtenha a representação dos seguintes números na base indicada:

a) $(101101{,}00101)_2$ na base $b = 4$.

b) $(23{,}1)_4$ na base $b = 2$.

1.2 Representação de números em máquina

Usualmente, números são manipulados em máquina através de suas representações em registros com *n-bits*. Ao longo desta seção, vamos representar um tal registro por

$$[b_0\ b_1\ b_2\ \cdots\ b_{n-1}], \tag{1.24}$$

onde cada *bit* é $b_i = 0, 1$, $i = 0, 1, 2, \ldots, n-1$.

Na sequência, fazemos uma breve discussão sobre as formas comumente usadas para a manipulação de números em computadores.

1.2.1 Números inteiros

A representação de complemento de 2 é usualmente utilizada em computadores para a manipulação de números inteiros. Nesta representação, um registro de *n-bits*

$$[b_0\ b_1\ b_2\ \cdots\ b_{n-1}], \tag{1.25}$$

representa o número inteiro

$$x = -d_{n-1}2^{n-1} + (d_{n-2}\ldots d_2 d_1 d_0)_2. \tag{1.26}$$

Exemplo 1.2.1. O registro de 8 *bits*

$$[1\ 1\ 0\ 0\ 0\ 0\ 0\ 0] \tag{1.27}$$

representa o número

$$x = -0\cdots 2^7 + (0000011)_2 \tag{1.28}$$
$$= 2^1 + 2^0 = 3. \tag{1.29}$$

No `GNU Octave`, podemos usar:

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

   1   1   0   0   0   0   0   0

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

Nesta representação de complemento de 2, o maior e o menor números inteiros que podem ser representados em um registro com n-*bits* são

$$[1\ 1\ 1\ \cdots\ 1\ 0] \text{ e } [1\ 0\ 0\ 0\ \cdots\ 0], \tag{1.30}$$

respectivamente. Já o zero é obtido com o registro

$$[0\ 0\ 0\ 0\ 0\ 0\ 0\ 0]. \tag{1.31}$$

Exemplo 1.2.2. Com um registro de 8-*bits*, temos que o maior e o menor números inteiros representados em complemento de 2 são

$$[1\ 1\ 1\ 1\ 1\ 1\ 1\ 0] \sim x = -0\cdot 2^7 + (1111111)_2 = 127, \tag{1.32}$$
$$[0\ 0\ 0\ 0\ 0\ 0\ 0\ 1] \sim x = -1\cdot 2^7 + (1111111)_2 = -128, \tag{1.33}$$
$$\tag{1.34}$$

respectivamente. Confirmamos isso no `GNU Octave` com

```
>> intmax('int8')
ans = 127
>> intmin('int8')
ans = -128
```

A adição de números inteiros na representação de complemento de 2 pode ser feita de maneira simples. Por exemplo, consideremos a soma $3+9$ usando registros de 8 *bits*. Temos

$$3 \sim [1\ 1\ 0\ 0\ 0\ 0\ 0\ 0] \tag{1.35}$$

$$9 \sim [1\ 0\ 0\ 1\ 0\ 0\ 0\ 0] + \tag{1.36}$$
$$- - - - - - - - - - \tag{1.37}$$
$$12 \sim [0\ 0\ 1\ 1\ 0\ 0\ 0\ 0] \tag{1.38}$$

Em representação de complemento de 2, a representação de um número negativo $-x$ pode ser obtida da representação de x, invertendo seus *bits* e somando 1. Por exemplo, a representação de -3 pode ser obtida da representação de 3, como segue

$$3 \sim [1\ 1\ 0\ 0\ 0\ 0\ 0\ 0]. \tag{1.39}$$

Invertendo seus *bits* e somando 1, obtemos

$$-3 \sim [1\ 0\ 1\ 1\ 1\ 1\ 1\ 1]. \tag{1.40}$$

A subtração de números inteiros usando a representação de complemento de 2 fica, então, tanto simples quanto a adição. Por exemplo:

$$3 \sim [1\ 1\ 0\ 0\ 0\ 0\ 0\ 0] \tag{1.41}$$
$$-9 \sim [1\ 1\ 1\ 0\ 1\ 1\ 1\ 1] + \tag{1.42}$$
$$- - - - - - - - - - \tag{1.43}$$
$$-6 \sim [0\ 1\ 0\ 1\ 1\ 1\ 1\ 1] \tag{1.44}$$

Observação 1.2.1. Por padrão, o `GNU Octave` usa a representação de complemento de 2 com 32 *bits* para números inteiros. Com isso, temos

```
>> intmin()
ans = -2147483648
>> intmax()
ans = 2147483647
```

1.2.2 Ponto flutuante

A manipulação de números decimais em computadores é comumente realizada usando a representação de ponto flutuante de 64-*bits*. Nesta, um dado registro de 64-*bits*

$$[m_{52}\ m_{51}\ m_{50}\ \cdots\ m_1 \mid c_0\ c_1\ c_2\ \cdots\ c_{10} \mid s] \tag{1.45}$$

representa o número

$$x = (-1)^s M \cdot 2^{c-1023}, \tag{1.46}$$

onde M é chamada de mantissa e c da característica, as quais são definidas por

$$M := (1{,}m_1 m_2 m_3 \dots m_{52})_2, \tag{1.47}$$
$$c := (c_{10} \dots c_2 c_1 c_0)_2. \tag{1.48}$$

Exemplo 1.2.3. Por exemplo, na representação em ponto flutuante de 64-*bits*, temos que o registro

$$[0\ 0\ 0\ \cdots\ 0\ 1\ 0\ 1 \mid 0\ 0\ 0\ \cdots\ 0\ 1 \mid 1] \tag{1.49}$$

representa o número $-3{,}25$.

1.2.3 Erro de arredondamento

Dado um número real x, sua representação $fl(x)$ em ponto flutuante é dada pelo registro que representa o número mais próximo de x. Este procedimento é chamado de arredondamento por proximidade. Por exemplo, $x = 1{,}1$ é representado pelo registro

$$[0101100110011001100110011001100110011001100110011001100110001111111111100]$$

o qual é o número

$$fl(x) = 1{,}1000000000000000088817841970012523233890533447265625,$$

sendo o erro de arredondamento $|x - fl(x)| \approx 8{,}9 \times 10^{-17}$.

Observemos que o erro de arredondamento varia conforme o número dado, podendo ser zero no caso de $x = fl(x)$. Comumente, utiliza-se o **épsilon de máquina** como uma aproximação deste erro. O épsilon de máquina é definido como a distância entre o número 1 e seu primeiro sucessor em ponto flutuante. Notemos que

$$1 = fl(1) \sim [0\ 0\ 0\ \cdots\ 0 \mid 1\ 1\ 1\ \cdots\ 1\ 0 \mid 0]. \tag{1.50}$$

Assim sendo, o primeiro sucessor de $fl(1)$ é

$$fl(1) + \text{eps} \sim [1\ 0\ 0\ \cdots\ 0 \mid 1\ 1\ 1\ \cdots\ 1\ 0 \mid 0] \tag{1.51}$$

onde eps é o épsilon de máquina e

$$\text{eps} = 2^{-52} \approx 2{,}22 \times 10^{-16}. \tag{1.52}$$

No `GNU Octave`, o eps está definido como constante:

```
>> eps
ans =    2.2204e-16
```

A aritmética em ponto flutuante requer arredondamentos sucessivos de números. Por exemplo, a computação da soma de dois números dados x e y é feita a partir de suas representações em ponto flutuante $fl(x)$ e $fl(y)$. Então, computa-se $z = fl(x)+fl(y)$ e o resultado é $fl(z)$. Observe, inclusive que $fl(x+y)$ pode ser diferente de $fl(fl(x)+fl(y))$. Por exemplo

```
>> 0.1+0.2 == 0.3
ans = 0
```

Exercício

Exercício 1.2.1. Considerando a representação de complemento de 2 de números inteiros, obtenha os registros de 8-*bits* dos seguintes números:

a) 15

b) -15

c) 32

d) -32

Exercício 1.2.2. Considerando a representação de complemento de 2 de números inteiros, obtenha os registros de 16-*bits* dos seguintes números:

a) 1024

b) -1024

Exercício 1.2.3. Considerando a representação de complemento de 2 de números inteiros, qual é o maior número que pode ser representado por um registro de 32-*bits* da forma

$$[1\ 0\ b_2\ b_3\ b_4\ \cdots\ b_{30}\ 1], \tag{1.53}$$

onde $b_i \in \{0,1\}$, $i = 2,3,4,\cdots,15$.

Exercício 1.2.4. Obtenha os registros em ponto flutuante de 64-*bits* dos seguintes números:

a) $-1{,}25$

b) 3

Exercício 1.2.5. Considerando a representação em ponto flutuante de 64-*bits*, encontre o número que é representado pelos seguintes registros:

a) $[000\ldots011|1000\ldots01|0]$

b) $[111\ldots1|111\ldots1|1]$

1.3 Notação científica

Enquanto que a manipulação de números decimais é comumente feita usando-se da aritmética em ponto flutuante, a interpretação dos parâmetros dos problemas de interesse e seus resultados é normalmente feita com poucos dígitos. Nesta seção, introduziremos algumas notações que serão utilizadas ao longo deste material.

A **notação científica** é a representação de um dado número na forma

$$d_n \ldots d_2 d_1 d_0{,}d_{-1}d_{-2}d_{-3}\ldots \times 10^E, \tag{1.54}$$

onde d_i, $i = n,\ldots,1,0,-1,\ldots$, são algarismos da base 10. A parte à esquerda do sinal $\times$ é chamada de mantissa do número e E é chamado de expoente (ou ordem de grandeza).

Exemplo 1.3.1. O número 31,515 pode ser representado em notação científica das seguintes formas

$$\begin{aligned} 31{,}515 \times 10^0 &= 3{,}1515 \times 10^1 && (1.55)\\ &= 315{,}15 \times 10^{-2} && (1.56)\\ &= 0{,}031515 \times 10^3, && (1.57) \end{aligned}$$

entre outras tantas possibilidades.

No exemplo anterior (Exemplo 1.3.1), podemos observar que a representação em notação científica de um dado número não é única. Para contornar isto, introduzimos a **notação científica normalizada**, a qual tem a forma

$$d_0{,}d_{-1}d_{-2}d_{-3}\ldots \times 10^E, \tag{1.58}$$

com $d_0 \neq 0$[2].

Exemplo 1.3.2. O número 31,515 representado em notação científica normalizada é $3{,}1515 \times 10^1$.

Como vimos na seção anterior, costumeiramente usamos da aritmética de ponto flutuante nas computações, com a qual os números são representados com muito mais dígitos dos quais estamos interessados na interpretação dos resultados. Isto nos leva de volta a questão do arredondamento.

Dizemos que um número está representado com n **dígitos significativo** (na notação científica normalizada) quando está escrito na forma

$$d_0{,}d_1d_2\ldots d_{n-1} \times 10^E, \tag{1.59}$$

com $d_0 \neq 0$.

1.3.1 Arredondamento

Observamos que pode ocorrer a necessidade de se arredondar um número para obter sua representação com um número finito de dígitos significativos. Por exemplo, para representarmos o número $x = 3{,}1415 \times 10^1$ com 3 dígitos significativos, precisamos determinar de que forma vamos considerar a contribuição de seus demais dígitos a direita. Isto, por sua vez, é determinado pelo tipo de arredondamento que iremos utilizar.

O tipo de arredondamento mais comumente utilizado é o chamado **arredondamento por proximidade com desempate par**. Neste, a representação escolhida é aquela mais próxima do número dado. Por exemplo, a representação de

$$x = 3{,}1515 \times 10^1 \tag{1.60}$$

[2]No caso do número zero, temos $d_0 = 0$.

com três dígitos significativos é

$$x = 3{,}15 \times 10^1. \tag{1.61}$$

Agora, sua representação com apenas dois dígitos significativos é

$$x = 3{,}2 \times 10^1. \tag{1.62}$$

No caso de empate, usa-se a seguinte regra: 1) no caso de o último dígito significativo ser par, este é mantido; 2) no caso de o último dígito significativo ser ímpar, este é acrescido de uma unidade. Por exemplo, no caso do número $x = 3{,}1515 \times 10^1$, sua representação com 4 dígitos significativos é

$$x = 3{,}152 \times 10^1. \tag{1.63}$$

No `GNU Octave`, o arredondamento por proximidade com desempate par é o padrão para números em ponto flutuante. Vejamos os seguintes casos:

```
>> printf("%1.1\E\n",0.625)
6.2E-01
>> printf("%1.1\E\n",0.635)
6.4E-01
```

No restante deste material estaremos assumindo a notação científica normalizada com arredondamento por proximidade.

Exercícios

Exercício 1.3.1. Obtenha a representação de 2718,2818 em notação científica normalizada com:

a) 3 dígitos significativos.

b) 4 dígitos significativos.

Por padrão no `GNU Octave`, números em ponto flutuante são arredondados por proximidade com desempate par. Então, explique o que está ocorrendo nos seguintes casos:

```
>> printf("%1.3\E\n",3.1515)
3.151E+00
>> printf("%1.3\E\n",3.1525)
3.152E+00
```

1.4 Tipos e medidas de erros

Ao utilizarmos computadores na resolução de problemas matemáticos, acabamos obtendo soluções aproximadas aproximadas. A diferença entre a solução exata e a computada solução aproximada é chamada de erro. O erro é comumente classificado nas seguintes duas categorias:

- **Erro de arredondamento**. Este é o erro que ocorre na representação aproximada de números na máquina.
- **Erro de truncamento**. Este é o erro que ocorre na interrupção (truncamento) de um procedimento com infinitos passos.

Exemplo 1.4.1. O erro de arredondamento em aproximar π por $3{,}1415\times10^{0}$ é de aproximadamente $9{,}3\times10^{-5}$.

Exemplo 1.4.2. Consideremos a seguinte série numérica $\sum_{k=0}^{\infty}1/k! = e \approx 2{,}7183\times10^{0}$. Ao computarmos esta série no computador, precisamos truncá-la em algum k suficientemente grande. Por exemplo, trunca a série em seu décimo termo, temos

$$\sum_{k=1}^{\infty}\frac{1}{k!} \approx 1+1+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{9} \tag{1.64}$$

$$\approx 2{,}7182815\times10^{0} =: \tilde{e}. \tag{1.65}$$

A diferença $e-\tilde{e}\approx 3\times10^{-7}$ é o erro de truncamento associado.

Suponhamos, agora, que x seja o valor exato de uma quantidade de interesse e $\tilde{x}$ a quantidade computada (i.e., uma aproximação de x). Em matemática numérica, utilizamos frequentemente as seguintes medidas de erro:

- Erro absoluto:

$$e_{abs} := |x-\tilde{x}|. \tag{1.66}$$

- Erro relativo:

$$e_{rel} := \frac{|x-\tilde{x}|}{|x|}\,(\times100\%)\,. \tag{1.67}$$

A vantagem do erro relativo é em levar em conta a ordem de grandeza das quantidades x e $\tilde{x}$. Vejamos o seguinte exemplo (Exemplo 1.4.3).

Exemplo 1.4.3. Observemos os seguintes casos:

a) $x = 1{,}0$ e $\tilde{x} = 1{,}1$:

$$e_{abs} = |x - \tilde{x}| = 1 \times 10^{-1}. \tag{1.68}$$

$$e_{rel} = \frac{|x - \tilde{x}|}{|x|} = 1 \times 10^{-1} = 10\%. \tag{1.69}$$

b) $x = 10000{,}0$ e $\tilde{x} = 11000{,}0$:

$$e_{abs} = |x - \tilde{x}| = 1 \times 10^{3}. \tag{1.70}$$

$$e_{rel} = \frac{|x - \tilde{x}|}{|x|} = 1 \times 10^{-1} = 10\%. \tag{1.71}$$

Outra medida de erro comumente empregada é o **número de dígitos significativos corretos**. Dizemos que $\tilde{x}$ aproxima x com n dígitos significativos corretos, quando

$$\frac{|x - \tilde{x}|}{|x|} < 5 \times 10^{-n}. \tag{1.72}$$

Exemplo 1.4.4. Vejamos o seguintes casos:

- $x = 2$ e $\tilde{x} = 2{,}5$:

$$\frac{|x - \tilde{x}|}{|x|} = 0{,}25 < 5 \times 10^{-1}, \tag{1.73}$$

  donde concluímos que $\tilde{x} = 2{,}5$ é uma aproximação com 1 dígito significativo correto de $x = 2$.

- $x = 1$ e $\tilde{x} = 1{,}5$:

$$\frac{|x - \tilde{x}|}{|x|} = 0{,}5 < 5 \times 10^{-0}, \tag{1.74}$$

  donde concluímos que $\tilde{x} = 1{,}5$ é uma aproximação com zero dígito significativo correto de $x = 1$.

Exercícios

Exercício 1.4.1. Calcule o erro absoluto na aproximação de

a) π por 3,14.

b) $10e$ por 27,18.

Forneça as respostas com 4 dígitos significativos.

Exercício 1.4.2. Calcule o erro relativo na aproximação de

a) π por 3,14.

b) $10e$ por 27,18.

Forneça as respostas em porcentagem.

Exercício 1.4.3. Com quantos dígitos significativos corretos

a) 3,13 aproxima π?

b) 27,21 aproxima $10e$?

Exercício 1.4.4. Obtenha uma estimativa do erro de truncamento em se aproximar o valor de sen(1) usando-se $p_5(1)$, onde $p_5(x)$ é o polinômio de Taylor de grau 5 da função sen(x) em torno de $x = 0$.

1.5 Propagação de erros

Nesta seção, vamos introduzir uma estimativa para a propagação de erros (de arredondamento) na computação de um problema. Para tando, vamos considerar o caso de se calcular o valor de uma dada função f em um dado ponto x, i.e. queremos calcular y com

$$y = f(x). \tag{1.75}$$

Agora, assumindo que x seja conhecido com um erro $e(x)$, este se propaga no cálculo da f, levando a um erro $e(y)$ no valor calculado de y. Ou seja, temos

$$y + e(y) = f(x + e(x)). \tag{1.76}$$

Notemos que $e_{abs}(x) = |e(x)|$ é o erro absoluto associado a x e $e_{abs}(y) = |e(y)|$ é o erro absoluto associado a y.

Nosso objetivo é de estimar $e_{abs}(y)$ com base em $e_{abs}(x)$. Para tanto, podemos tomar a aproximação de $f(x+e(x))$ dada pelo polinômio de Taylor de grau 1 de f em torno de x, i.e.

$$f(x + e(x)) = f(x) + f'(x)e(x) + O\left(e^2(x)\right). \tag{1.77}$$

Então, de (1.75) e (1.76), temos

$$e(y) = f'(x)e(x) + O(e^2(x)). \tag{1.78}$$

Daí, passando ao valor absoluto e usando a desigualdade triangular, obtemos

$$e_{abs}(y) \leq |f'(x)|e_{abs}(x) + O\left(e^2_{abs}(x)\right). \tag{1.79}$$

Deste resultado, consideraremos a seguinte estimativa de propagação de erro

$$e_{abs}(y) \approx |f'(x)|e_{abs}(x). \tag{1.80}$$

Exemplo 1.5.1. Consideremos o problema em se calcular $y = f(x) = x^2\,\mathrm{sen}(x)$ com $x = \pi/3 \pm 0{,}1$. Usando (1.80) para estimarmos o erro absoluto $e_{abs}(y)$ no cálculo de y com base no erro absoluto $e_{abs}(x) = 0{,}1$, calculamos

$$\begin{aligned} e_{abs}(y) &= |f'(x)|e_{abs}(x) && (1.81)\\ &= |2x\,\mathrm{sen}(x) + x^2\cos(x)|e_{abs}(x) && (1.82)\\ &= 2{,}3621 \times 10^{-1}. && (1.83)\end{aligned}$$

Com isso, podemos concluir que um erro em x de tamanho 0,1 é propagado no cálculo de $f(x)$, causando um erro pelo menos duas vezes maior em y. Também, podemos interpretar este resultado do ponto de vista do erro relativo. O erro relativo associado a x é

$$e_{rel}(x) = \frac{e_{abs}(x)}{|x|} = \frac{0{,}1}{\pi/3} = 9{,}5493 \times 10^{-2} \approx 1\%, \tag{1.84}$$

acarretando um erro relativo em y de

$$e_{rel}(y) = \frac{e_{abs}(y)}{|y|} = \frac{e_{abs}(y)}{|f(x)|} = 2{,}4872 \times 10^{-1} \approx 2\%. \tag{1.85}$$

Podemos fazer estas contas com o seguinte código `GNU Octave`:

```
format short e

f = @(x) x^2*sin(x);
fl = @(x) 2*x*sin(x) + x^2*cos(x);

x=pi/3;
eax=0.1
erx = eax/abs(x)

eay = abs(fl(x))*eax
ery = eay/abs(f(x))
```

Associada à estimativa (1.5.1), temos

$$\begin{aligned} e_{rel}(y) = \frac{e_{abs}(y)}{|y|} &= \frac{|f'(x)|}{|y|} e_{abs}(x) \\ &= \frac{|x| \cdot |f'(x)|}{|f(x)|} \frac{e_{abs}(x)}{|x|} \\ &= \left| \frac{xf'(x)}{f(x)} \right| e_{rel}(x). \end{aligned}$$

Desta última equação, definimos o **número de condicionamento** de f, denotado por

$$\kappa_f(x) := \left| \frac{xf'(x)}{f(x)} \right|. \tag{1.86}$$

Observamos que $\kappa_f(x)$ é a escala com que erros em x são propagados no cálculo de $y = f(x)$.

Exemplo 1.5.2. O número de condicionamento da função $f(x) = x^2 \operatorname{sen}(x)$ no ponto $x = \pi/3$ pode ser calculado de

$$\kappa_f(x) = \left| \frac{xf'(x)}{f(x)} \right| \tag{1.87}$$

$$= \left| \frac{x(2x \operatorname{sen}(x) + x^2 \cos(x))}{x^2 \operatorname{sen}(x)} \right|. \tag{1.88}$$

Substituindo x por $\pi/3$ temos

$$\kappa_f(\pi/3) = 2{,}6046. \tag{1.89}$$

Notamos que este resultado é compatível com as observações feitas no Exemplo 1.5.1.

Podemos computar $\kappa_f(x)$ com o seguinte código `GNU Octave`:

```
format short e

f = @(x) x^2*sin(x);
fl = @(x) 2*x*sin(x) + x^2*cos(x);

x=pi/3;

kf = abs(x*fl(x)/f(x))
```

A estimativa (1.80) pode ser generalizada para uma função de várias variáveis. No caso de uma função $y = f(x_1,x_2,\ldots,x_n)$, temos

$$e_{abs}(y) = \sum_{k=1}^{n} \left|\frac{\partial f}{\partial x_k}\right| e_{abs}(x_k). \tag{1.90}$$

Exemplo 1.5.3. Consideremos o problema em se calcular $z = f(x,y) = x^2\,\mathrm{sen}(x)\cos(y)$ com $x = \pi/3 \pm 0{,}1$ e $y = \pi/4 \pm 0{,}02$. Usando (1.90) para estimarmos o erro absoluto $e_{abs}(z)$ no cálculo de z com base nos erros absolutos $e_{abs}(x) = 0{,}1$ e $e_{abs}(y) = 0{,}02$, calculamos

$$\begin{aligned} e_{abs}(z) &= \left|\frac{\partial f}{\partial x}\right| e_{abs}(x) + \left|\frac{\partial f}{\partial y}\right| e_{abs}(y) && (1.91)\\ &= |(2x\,\mathrm{sen}(x) + x^2\cos(x))\cos(y)|e_{abs}(x) \\ &+ \left|-x^2\,\mathrm{sen}(x)\,\mathrm{sen}(y)\right| e_{abs}(y) && (1.92)\\ &= 1{,}8046 \times 10^{-1}. && (1.93) \end{aligned}$$

Podemos fazer estas contas com o seguinte código `GNU Octave`:

```
format short e
```

```
f = @(x,y) x^2*sin(x)*cos(y);
fx = @(x,y) (2*x*sin(x) + x^2*cos(x))*cos(y);
fy = @(x,y) -x^2*sin(x)*sin(y);

x=pi/3;
eax=0.1

y=pi/4;
eay=0.02

eaz = abs(fx(x,y))*eax + abs(fy(x,y))*eay
```

1.5.1 Cancelamento catastrófico

No computador (com aritmética de ponto flutuante de 64-*bits*), as operações e funções elementares são computadas, usualmente, com um erro próximo do épsilon de máquina (eps $\approx 10^{-16}$). Entretanto, em algumas situações estas operações fundamentais acarretam erros maiores, causando uma perda de precisão.

O chamado cancelamento catastrófico ocorre quando ao computarmos a diferença entre dois números próximos. Para ilustrá-lo, consideremos os seguintes números

$$x = 314150000001549, \tag{1.94}$$
$$y = 314150000002356. \tag{1.95}$$

Suponhamos, ainda, os arredondamentos de x e y com 12 dígitos significativos

$$\tilde{x} = 314150000002000, \tag{1.96}$$
$$\tilde{y} = 314150000002000. \tag{1.97}$$

Notemos que os erros relativos associados às aproximações de x e y por $\tilde{x}$ e $\tilde{y}$ são

$$e_{rel}(x) = \frac{|x - \tilde{x}|}{|x|} \approx 10^{-10}\%, \tag{1.98}$$

$$e_{rel}(y) = \frac{|y - \tilde{y}|}{|y|} \approx 10^{-10}\%, \tag{1.99}$$

respectivamente. Agora, temos

$$y - x = 807 \quad \text{e} \quad \tilde{y} - \tilde{x} = 0. \tag{1.100}$$

Ou seja, o erro relativa na aproximação de $y - x$ por $\tilde{y} - \tilde{x}$ é

$$e_{rel}(y - x) = \frac{|(y - x) - (\tilde{y} - \tilde{x})|}{(y - x)} = \frac{807}{807} = 100\%! \tag{1.101}$$

Vejamos outros exemplos.

Exemplo 1.5.4. Na Tabela 1.1 temos os erros em se computar

$$\frac{(1 + x^4) - 1}{x^4} \tag{1.102}$$

para diferentes valores de x. Observamos que, para o valor de $x = 0{,}001$ o erro na computação já é da ordem de 10^{-5} e para valores de x menores ou iguais a 0,0001 o erro é catastrófico. Isto ocorre, pois se $x \leq 10^{-4}$, então $x^4 \leq 10^{-16} < \text{eps}$ e, portanto, $(1 + x^4) - 1 = 0$.

x	erro
1	0
10^{-1}	$1{,}1 \times 10^{-13}$
10^{-2}	$6{,}1 \times 10^{-9}$
10^{-3}	$8{,}9 \times 10^{-5}$
10^{-4}	$1{,}0 \times 10^{0}$
10^{-5}	$1{,}0 \times 10^{0}$

Tabela 1.1: Resultados referentes ao Exemplo 1.5.4.

Exemplo 1.5.5. Uma equação de segundo grau $ax^2 + bx + c = 0$ tem raízes

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}, \tag{1.103}$$

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}. \tag{1.104}$$

Entretanto, no caso de b e $\sqrt{b^2-4ac}$ serem ambos positivos, a fórmula (1.103) não é adequada para a computação da raiz x_1, pois pode ocorrer cancelamento catastrófico. Podemos contornar este problema reescrevendo (1.103) da seguinte forma

$$x_1 = \frac{-b+\sqrt{b^2-4ac}}{2a}\frac{-b-\sqrt{b^2-4ac}}{-b-\sqrt{b^2-4ac}} \tag{1.105}$$

$$= \frac{-b^2+b^2-4ac}{2a(-b-\sqrt{b^2-4ac})} \tag{1.106}$$

$$= \frac{2c}{b+\sqrt{b^2-4ac}}, \tag{1.107}$$

a qual não sofre mais de cancelamento catastrófico. Observemos que também pode ocorrer cancelamento catastrófico no cálculo de x_2 pela fórmula (1.104), no caso de b e $\sqrt{b^2-4ac}$ serem ambos negativos. No `GNU Octave`, podemos escrever a seguinte função para computar as raízes de um polinômio quadrático ax^2+bx+c (código):

```
function [r1,r2] = ex_solpq(a,b,c)
  aux = -0.5 * (b + sign(b)*sqrt(b^2-4*a*c));
  r1 = aux/a;
  r2 = c/aux;
endfunction
```

Exercícios

Exercício 1.5.1. Considerando que $x = 2\pm0{,}1$, estime o erro absoluto em se calcular $y = e^{-x^2}\cos(\pi x/3)$. Forneça a estimativa com 7 dígitos significativos por arredondamento.

Exercício 1.5.2. Considerando que $x = 2 \pm 2\%$ e $y = 1{,}5 \pm 0{,}3$, estime o erro absoluto em se calcular $y = e^{-x^2}\cos(\pi y/3)$. Forneça a estimativa com 6 dígitos significativos por arredondamento.

Exercício 1.5.3. Considere a computação de

$$y = \frac{1-\cos(h)}{h} \tag{1.108}$$

para $h = 10^{-9}$. Compute o valor de y reescrevendo esta expressão de forma a mitigar o cancelamento catastrófico. Forneça o valor computado de y com 2 dígitos significativos por arredondamento.

Capítulo 2

Equação com uma incógnita

Neste capítulo, discutiremos sobre métodos numéricos para resolver equações com uma incógnita real. Para tanto, notemos que toda equação pode ser reescrita na seguinte forma equivalente

$$f(x) = 0, \tag{2.1}$$

onde f é uma função adequada. Isto é, o problema de se encontrar a incógnita de uma dada equação pode ser reescrito como um problema de encontrar os zeros (ou raízes) de uma função de uma variável real.

Os métodos numéricos que abordaremos ao longo deste capítulo são descritos para problemas da forma (2.1).

2.1 Método da bisseção

O método da bisseção explora o fato de que toda função contínua f com $f(a) \cdot f(b) < 0$ (i.e., $f(a)$ e $f(b)$ tem sinais diferentes) tem pelo menos um zero no intervalo (a, b)[1].

Exemplo 2.1.1. Consideremos o problema de resolver

$$\operatorname{sen}^2\left(x + \frac{\pi}{4}\right) = x^3 - \frac{\pi}{4}x^2 - \frac{5\pi^2}{16}x - \frac{3\pi^3}{64}. \tag{2.2}$$

[1]Esta é uma consequência imediata do teorema do valor intermediário.

Este problema é equivalente a encontrar os zeros da seguinte função

$$f(x) = \operatorname{sen}^2\left(x + \frac{\pi}{4}\right) - x^3 + \frac{\pi}{4}x^2 + \frac{5\pi^2}{16}x + \frac{3\pi^3}{64}. \tag{2.3}$$

Os zeros exatos[2] desta função são $x_1 = 3\pi/4 \approx 2{,}3562$ e $x_2 = x_3 = -\pi/4 \approx -0{,}78540$ (veja a Figura 2.1).

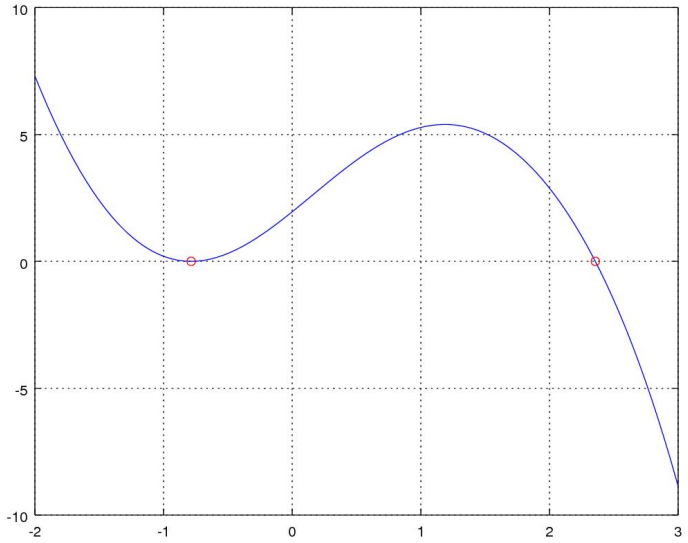

Figura 2.1: Esboço da função f do Exemplo 2.1.1.

Observamos que esta função é contínua e que, por exemplo, $f(-2) > 0$ e $f(3) < 0$, logo $f(-2) \cdot f(3) < 0$ e, de fato, f tem pelo menos um zero[3] no intervalo $(-2, 3)$.

O esboço do gráfico da função f pode ser feito no `GNU Octave` com o seguinte código:

```
f = @(x) sin(x+pi/4).^2-x.^3+pi*x.^2/4+5*pi^2*x/16+3*pi^3/64;
xx=linspace(-2,3);
plot(xx,f(xx));grid
```

[2] O problema foi construído para que tivesse estas soluções.
[3] De fato, f tem três zeros no intervalo $(-2, 3)$.

Consideremos, então, uma função f contínua tal que $f(a) \cdot f(b) < 0$. O método da bisseção é iterativo, sendo que a primeira aproximação para uma solução de $f(x) = 0$ tomada como o ponto médio do intervalo (a, b), i.e.

$$x^{(1)} = \frac{a^{(1)} + b^{(1)}}{2}, \tag{2.4}$$

onde $a^{(1)} = a$ e $b^{(1)} = b$. Daí, se ocorrer $f(x^{(1)}) = 0$ o problema está resolvido. Caso contrário, f tem pelo menos um zero num dos subintervalos $(a^{(1)}, x^{(1)})$ ou $(x^{(1)}, b^{(1)})$, pois $f(a^{(1)}) \cdot f(x^{(1)}) < 0$ ou $f(x^{(1)}) \cdot f(b^{(1)}) < 0$, respectivamente e exclusivamente. No primeiro caso, escolhemos $(a^{(2)}, b^{(2)}) = (a^{(1)}, x^{(1)})$ ou, no segundo caso, tomamos $(a^{(2)}, b^{(2)}) = (x^{(1)}, b^{(1)})$. Então, a segunda aproximação para uma solução é computada como

$$x^{(2)} = \frac{a^{(2)} + b^{(2)}}{2}. \tag{2.5}$$

Daí, o procedimento se repete até obtermos uma aproximação com a precisão desejada.

Exemplo 2.1.2. Consideremos o problema de encontrar um zero da função

$$f(x) = \operatorname{sen}^2\left(x + \frac{\pi}{4}\right) - x^3 + \frac{\pi}{4}x^2 + \frac{5\pi^2}{16}x + \frac{3\pi^3}{64}. \tag{2.6}$$

Do esboço de seu gráfico (Figura 2.1) vemos que $f(2) \cdot f(3) \neq 0$ sendo que o zero $x = 3\pi/4 \approx 2{,}3562$ de f está no intervalo $(2, 3)$. Então, aplicando o método da bisseção com intervalo inicial $(a^{(1)}, b^{(1)}) = (2, 3)$ e aproximação inicial $x^{(1)} = (a^{(1)} + b^{(1)})/2$, obtemos as aproximações apresentadas na Tabela 2.1.

A tabela 2.1 pode ser obtida no `GNU Octave` com o seguinte código:

```
f = @(x) sin(x+pi/4).^2-x.^3+pi*x.^2/4+5*pi^2*x/16+3*pi^3/64;

a=2; b=3;
x=(a+b)/2
printf("%d %1.4E %1.4E %1.4E %d\n",...
       1,a,b,x,sign(f(a))*sign(f(x)))
for k=2:10
```

Tabela 2.1: Resultados referentes ao Exemplo 2.1.2.

k	$a^{(k)}$	$b^{(k)}$	$x^{(k)}$	$f(a^{(k)}) \cdot f(x^{(k)})$
1	2,0000	3,0000	2,5000	-1
2	2,0000	2,5000	2,2500	1
3	2,2500	2,5000	2,3750	-1
4	2,2500	2,3750	2,3125	1
5	2,3125	2,3750	2,3438	1
6	2,3438	2,3750	2,3594	-1
7	2,3438	2,3594	2,3516	1
8	2,3516	2,3594	2,3555	1
9	2,3555	2,3594	2,3574	-1
10	2,3555	2,3574	2,3564	-1

```
  if (f(x) == 0)
    disp("sol. encontrada")
  elseif (sign(f(a))*sign(f(x)) == -1)
    b=x;
  else
    a=x;
  end

  x=(a+b)/2;
  printf("%d %1.4E %1.4E %1.4E %d\n",...
         k,a,b,x,sign(f(a))*sign(f(x)))

end
```

2.1.1 Análise de convergência

Dada uma função estritamente monótona[4] e contínua $f : [a, b] \to \mathbb{R}$ com $f(a) \cdot f(b) < 0$, temos que o método da bisseção converge para o zero de f no intervalo (a, b).

De fato, como consequência imediata do teorema do valor intermediário, temos que f tem pelo menos um zero no intervalo (a, b). Agora, da hipótese

[4]Estritamente crescente ou estritamente decrescente.

de monotonicidade estrita, temos que f tem um único zero neste intervalo, o qual denotaremos por x^*.

Da construção das iteradas do método, temos

$$|x^{(k)} - x^*| \leq \frac{b^{(k)} - a^{(k)}}{2} \tag{2.7}$$

$$\leq \frac{b^{(k-1)} - a^{(k-1)}}{2^2} \tag{2.8}$$

$$\vdots \tag{2.9}$$

$$\leq \frac{b^{(1)} - a^{(1)}}{2^k}, \tag{2.10}$$

donde, temos a seguinte estimativa do erro de truncamento

$$|x^{(k)} - x^*| \leq \frac{b^{(1)} - a^{(1)}}{2^k}. \tag{2.11}$$

E, daí também, segue a convergência do método da bisseção, pois

$$\lim_{k\to\infty} |x^{(k)} - x^*| = \lim_{k\to\infty} \frac{b^{(1)} - a^{(1)}}{2^k} = 0. \tag{2.12}$$

Observação 2.1.1. No caso de f não ser estritamente monótona no intervalo (a, b), ainda podemos garantir a convergência do método da bisseção. Isto segue do fato de que após algumas iteradas, digamos k iteradas, a função f terá apenas um zero no intervalo $(a^{(k)}, b^{(k)})$. A partir daí, as estimativas acima podem ser aplicadas.

Exemplo 2.1.3. No Exemplo 2.1.2 aplicamos o método da bisseção para a função

$$f(x) = \operatorname{sen}^2\left(x + \frac{\pi}{4}\right) - x^3 + \frac{\pi}{4}x^2 + \frac{5\pi^2}{16}x + \frac{3\pi^3}{64}. \tag{2.13}$$

no intervalo $(2, 3)$. Observando os resultados mostrados na Tabela 2.1, vemos que

$$|x^{(10)} - x^*| = 2{,}5\text{e}{-4}, \tag{2.14}$$

com $x^* = x_1 = 3\pi/4$. Observamos que este resultado é consistente com a estimativa do erro de truncamento (2.11), da qual temos

$$|x^{(10)} - x^*| \leq \frac{b^{(1)} - a^{(1)}}{2^{10}} \tag{2.15}$$

$$= \frac{1}{2^{10}} = 9{,}8\text{e}{-4}. \tag{2.16}$$

Observação 2.1.2. (Taxa de convergência.) A estimativa de convergência (2.11) também pode ser usada para mostrarmos que, assintoticamente, o método da bisseção tem a seguinte taxa de convergência linear

$$\left|x^{(k+1)} - x^{(k)}\right| \lesssim \frac{1}{2}\left|x^{(k)} - x^{(k-1)}\right|^{\mathbf{1}}. \tag{2.17}$$

2.1.2 Zeros de multiplicidade par

Sejam f uma função suave e x^* um zero de multiplicidade par de f. Observamos que o método da bisseção não é diretamente aplicável para aproximar x^*. Isto ocorre, pois, neste caso, x^* será um ponto de mínimo ou de máximo local de f, não havendo pontos a e b próximos de x^* tal que $f(a) \cdot f(b) < 0$.

Agora, sendo x^* é um zero de multiplicidade $2m$ de f, temos que ela admite a seguinte decomposição

$$f(x) = (x - x^*)^{2m} g(x), \tag{2.18}$$

onde g é uma função suave e $g(x^*) \neq 0$. Daí, a derivada de f

$$f'(x) = 2m(x - x^*)^{2m-1} g(x) + (x - x^*)^{2m} g'(x), \tag{2.19}$$

tem x^* como um zero de multiplicidade $2m - 1$ (ímpar) e, desta forma, podemos aplicar o método da bisseção em f' para aproximar x^*.

Exemplo 2.1.4. A função

$$f(x) = \operatorname{sen}^2\left(x + \frac{\pi}{4}\right) - x^3 + \frac{\pi}{4}x^2 + \frac{5\pi^2}{16}x + \frac{3\pi^3}{64}. \tag{2.20}$$

tem $x = -\pi/4 \approx -0{,}7854$ como um zero de multiplicidade par (veja Figura 2.1). Para aplicarmos o método da bisseção para aproximarmos este zero, primeiramente, derivamos f

$$f'(x) = 2\sin(x + \pi/4)\cos(x + \pi/4) - 3x^2 + \frac{\pi}{2}x + \frac{5\pi^2}{16}. \tag{2.21}$$

O esboço do gráfico de f' (Figura 2.2) mostra que $f'(-1) \cdot f'(0) < 0$ sendo que no intervalo $(-1, 0)$ f' tem um zero de multiplicidade ímpar. Então, aplicando o método da bisseção a f' no intervalo inicial $(a^{(1)}, b^{(1)}) = (-1,\ 0)$, obtemos os resultados apresentados na Tabela 2.2. Nesta tabela são apresentados as iteradas até a convergência da solução com precisão de 10^{-3}.

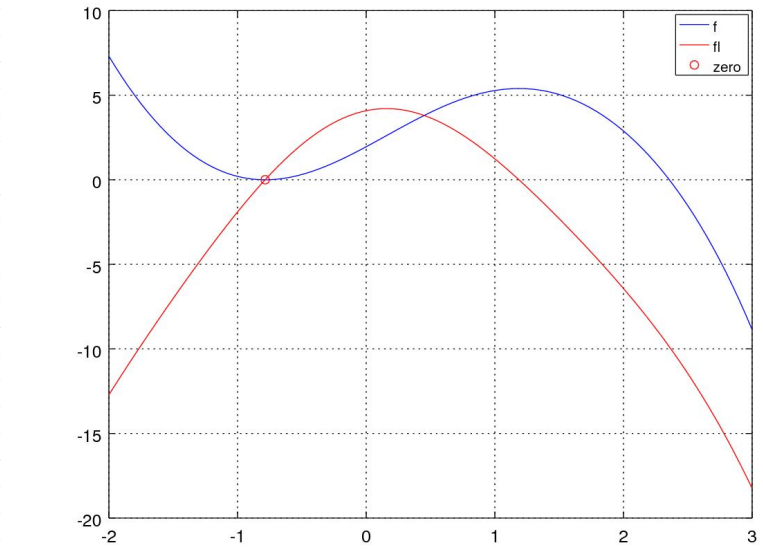

Figura 2.2: Esboço do gráfico da f e de sua derivada f' dada no Exemplo 2.1.4.

A tabela 2.2 pode ser obtida no `GNU Octave` com o seguinte código:

```
pkg load symbolic
syms x
f = @(x) sin(x+pi/4).^2-x.^3+pi*x.^2/4+5*pi^2*x/16+3*pi^3/64;
fl = function_handle(diff(f(x)));

TOL=1e-3;
a=-1;
b=0;
for k=1:15
  x=(a+b)/2;
  printf("%d %1.4E %1.4E %1.4E %d\n",...
```

Tabela 2.2: Resultados referentes ao Exemplo 2.1.2.

k	$a^{(k)}$	$b^{(k)}$	$x^{(k)}$	$f'(a^{(k)}) \cdot f'(x^{(k)})$
1	−1,0000e+0	0,0000e+0	−5,0000e−1	-1
2	−1,0000e+0	−5,0000e−1	−7,5000e−1	-1
3	−1,0000e+0	−7,5000e−1	−8,7500e−1	1
4	−8,7500e−1	−7,5000e−1	−8,1250e−1	1
5	−8,1250e−1	−7,5000e−1	−7,8125e−1	-1
6	−8,1250e−1	−7,8125e−1	−7,9688e−1	1
7	−7,9688e−1	−7,8125e−1	−7,8906e−1	1
8	−7,8906e−1	−7,8125e−1	−7,8516e−1	-1
9	−7,8906e−1	−7,8516e−1	−7,8711e−1	1
10	−7,8711e−1	−7,8516e−1	−7,8613e−1	1

```
      k,a,b,x,sign(fl(a))*sign(fl(x)));
  if ((fl(x)==0) | ((b-a)/2<TOL))
    disp('convergiu')
    break;
  elseif (sign(fl(a))*sign(fl(x))==-1)
    b=x;
  else
    a=x;
  end
endfor
```

Exercícios

Exercício 2.1.1. Use o método da bisseção para aproximar um zero de $f(x) = x^3 \operatorname{sen}(x) - \cos(x)$, aplicando como intervalo inicial $(a^{(1)}, b^{(1)}) = (0{,}5,\ 1)$ e aproximação inicial $x^{(1)} = (a^{(1)} + b^{(1)})/2$. Faça, então, 6 iterações de forma a obter a aproximação $x^{(7)}$ e forneça-a com 7 dígitos significativos por arredondamento.

Exercício 2.1.2. Considere que o método da bisseção para aproximar um zero de $f(x) = x^3 \operatorname{sen}(x) - \cos(x)$, aplicando como intervalo inicial $(a^{(1)}, b^{(1)}) = (0{,}5,\ 1)$ e aproximação inicial $x^{(1)} = (a^{(1)} + b^{(1)})/2$. Use a

estimativa de convergência (2.11)

$$\left|x^{(k)} - x^*\right| \leq \frac{b^{(1)} - a^{(1)}}{2^k}, \tag{2.22}$$

para estimar o número mínimo de iterações k_{conv} necessárias para se obter a solução com exatidão de 10^{-4}. Então, compute $x^{(k_{conv})}$ e forneça-o com 6 dígitos significativos por arredondamento.

Exercício 2.1.3. Use o método da bisseção para encontrar uma aproximação com precisão de 10^{-4} do zero de

$$f(x) = (-x^2 + 1{,}154x - 0{,}332929)\cos(x) + x^2 - 1{,}154x + 0{,}332929 \tag{2.23}$$

no intervalo $(0{,}55,\ 0{,}65)$. Forneça a aproximação computada com 7 dígitos significativos por arredondamento.

2.2 Método da falsa posição

O método da falsa posição é uma variação do método da bisseção. Dada uma função f contínua, escolhemos um intervalo inicial (a, b) tal que $f(a) \cdot f(b) < 0$ (i.e. f tem sinais trocados nos pontos a e b). Então, uma aproximação para o zero de f neste intervalo é computada como o ponto de interseção da reta secante a f pelos pontos $(a, f(a))$ e $(b, f(b))$, i.e.

$$x = a - \frac{b - a}{f(b) - f(a)} f(a). \tag{2.24}$$

Veja a Figura 2.3.

Mais explicitamente, o método da falsa posição consiste no seguinte procedimento iterativo:

1. Determinar um intervalo $(a^{(1)}, b^{(1)})$ tal que $f(a^{(1)}) \cdot f(b^{(1)}) < 0$.
2. Para $k = 1, 2, 3, \cdots, N$:

   2.1 $x^{(k)} = a^{(k)} - \dfrac{b^{(k)} - a^{(k)}}{f(b^{(k)}) - f(a^{(k)})} f(a^{(k)})$

   2.2 Verificar critério de parada.

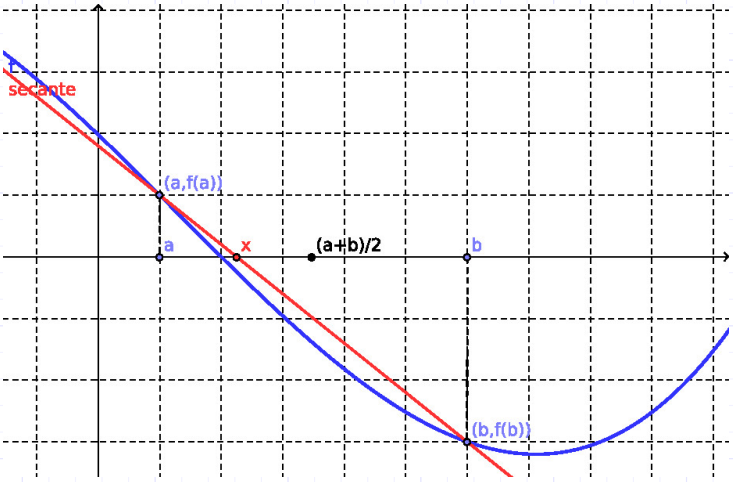

Figura 2.3: Ilustração do método da falsa posição (veja no Geogebra).

2.3 Se $f(a^{(k)}) \cdot f(x^{(k)}) < 0$, então $a^{(k+1)} = a^{(k)}$ e $b^{(k+1)} = x^{(k)}$.

2.4 Se $f(x^{(k)}) \cdot f(b^{(k)}) > 0$, então $a^{(k+1)} = x^{(k)}$ e $b^{(k+1)} = b^{(k)}$.

Exemplo 2.2.1. Consideremos o problema de aproximar o zero de

$$f(x) = \operatorname{sen}^2\left(x + \frac{\pi}{4}\right) - x^3 + \frac{\pi}{4}x^2 + \frac{5\pi^2}{16}x + \frac{3\pi^3}{64}. \tag{2.25}$$

no intervalo $(0, 3)$. A Tabela 2.3 mostra os resultados obtidos da aplicação do método da falsa posição com intervalo inicial $(a^{(1)}, b^{(1)}) = (2, 3)$. Aqui, o método foi iterado até a convergência com cinco dígitos significativos.

A tabela 2.3 pode ser obtida no `GNU Octave` com o seguinte código:

```
f = @(x) sin(x+pi/4).^2-x.^3+pi*x.^2/4+5*pi^2*x/16+3*pi^3/64;

TOL=1e-3;
a=2; b=3;
for k=1:10
  x=a-(b-a)/(f(b)-f(a))*f(a);
  printf("%d %1.4E %1.4E %1.4E %d\n",...
       k,a,b,x,sign(f(a))*sign(f(x)))

  if (f(x) == 0)
```

Tabela 2.3: Resultados referentes ao Exemplo 2.2.1.

k	$a^{(k)}$	$b^{(k)}$	$x^{(k)}$	$f'(a^{(k)}) \cdot f'(x^{(k)})$
1	2,0000	3,0000	2,2455	1
2	2,2455	3,0000	2,3240	1
3	2,3240	3,0000	2,3470	1
4	2,3470	3,0000	2,3536	1
5	2,3536	3,0000	2,3555	1
6	2,3555	3,0000	2,3560	1
7	2,3560	3,0000	2,3561	1
8	2,3561	3,0000	2,3562	1
9	2,3562	3,0000	2,3562	1
10	2,3562	3,0000	2,3562	1

```
    disp("convergiu")
    break;
  elseif (sign(f(a))*sign(f(x)) == -1)
    b=x;
  else
    a=x;
  end
end
```

Exercícios

Exercício 2.2.1. Use o método da falsa posição para aproximar um zero de $f(x) = x^3 \operatorname{sen}(x) - \cos(x)$, aplicando como intervalo inicial $(a^{(1)}, b^{(1)}) = (0{,}5,\ 1)$ e aproximação inicial

$$x^{(1)} = a^{(1)} - \frac{b^{(1)} - a^{(1)}}{f(b^{(1)}) - f(a^{(1)})} f(a^{(1)}). \tag{2.26}$$

Faça, então, 4 iterações deste método de forma a obter a aproximação $x^{(5)}$ e forneça-a com 7 dígitos significativos por arredondamento.

Exercício 2.2.2. Use o método da bisseção para encontrar uma aproximação com precisão de 10^{-4} do zero de

$$f(x) = (-x^2 + 1{,}154x - 0{,}332929)\cos(x) + x^2 - 1{,}154x + 0{,}332929 \tag{2.27}$$

no intervalo (0,55, 0,65). Forneça a aproximação computada com 7 dígitos significativos por arredondamento.

2.3 Iteração de ponto fixo

O ponto fixo de uma função dada g é o ponto x tal que

$$g(x) = x. \tag{2.28}$$

Geometricamente, pontos fixos são interseções do gráfico da g com a reta $y = x$, veja a Figura 2.4.

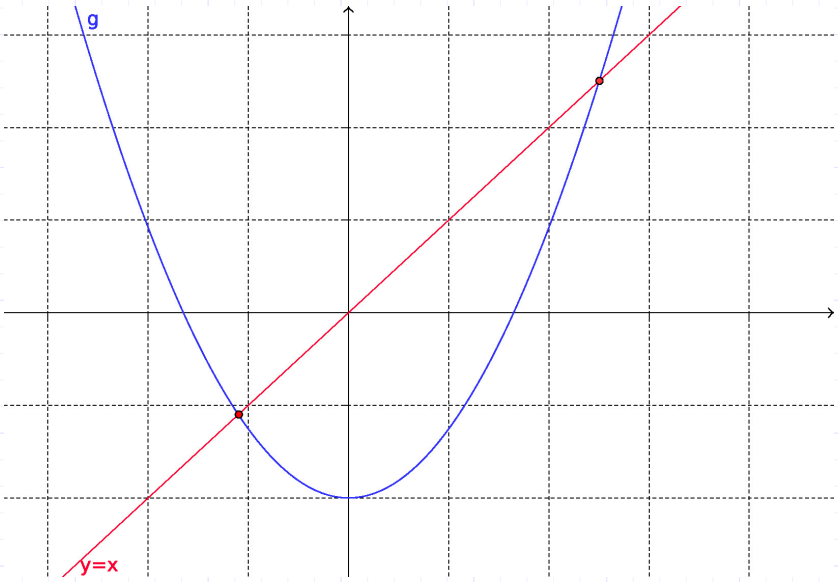

Figura 2.4: Exemplos de pontos fixos (veja no Geogebra).

Observamos que toda equação de uma incógnita pode ser reescrita de forma equivalente como um problema de ponto fixo.

Exemplo 2.3.1. Consideremos o problema de resolver

$$\operatorname{sen}^2\left(x + \frac{\pi}{4}\right) = x^3 - \frac{\pi}{4}x^2 - \frac{5\pi^2}{16}x - \frac{3\pi^3}{64}. \tag{2.29}$$

Podemos reescrevê-la como o problema de se obter os zeros da seguinte função

$$f(x) = \operatorname{sen}^2\left(x + \frac{\pi}{4}\right) - x^3 + \frac{\pi}{4}x^2 + \frac{5\pi^2}{16}x + \frac{3\pi^3}{64}. \tag{2.30}$$

Por sua vez, este problema é equivalente aos seguintes problemas de ponto fixo (entre outros):

$$a)\ g_1(x) = \frac{16}{5\pi^2}\left[-\operatorname{sen}^2\left(x+\frac{\pi}{4}\right) + x^3 - \frac{\pi}{4}x^2 - \frac{3\pi^3}{64}\right] = x. \tag{2.31}$$

$$b)\ g_2(x) = \sqrt[3]{\operatorname{sen}^2\left(x+\frac{\pi}{4}\right) + \frac{\pi}{4}x^2 + \frac{5\pi^2}{16}x + \frac{3\pi^3}{64}} \tag{2.32}$$

Na Figura 2.5 podemos observar que os zeros da f (a saber, $x_1 = 3\pi/4 \approx 2{,}3562$ e $x_2 = x_3 = -\pi/4 \approx -0{,}78540$) coincidem com os pontos fixos das funções g_1 e g_2.

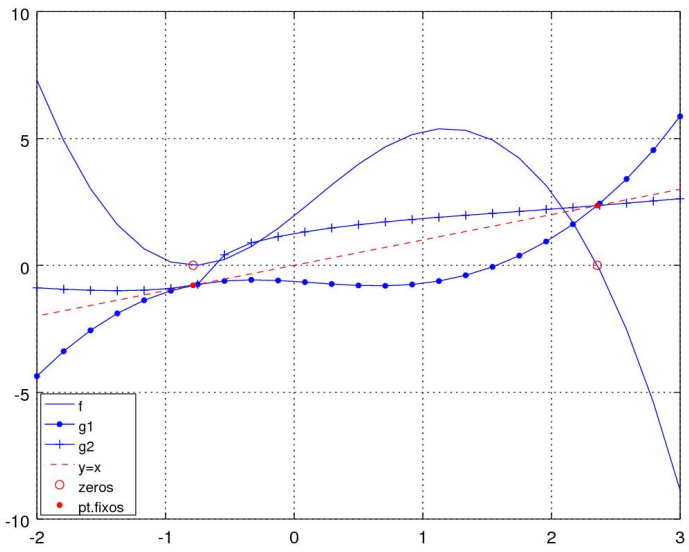

Figura 2.5: Esboço da função f do Exemplo 2.3.1.

Em muitos casos, é possível obter aproximações de um ponto fixo de uma dada função g pela chamada **iteração de ponto fixo**:

$$x^{(1)} = \text{aprox. inicial} \tag{2.33}$$

$$x^{(k+1)} = g(x^{(k)}), \quad k = 1, 2, 3, \ldots \tag{2.34}$$

Exemplo 2.3.2. Observemos que para a função g_1 dada no Exemplo 2.3.1, temos as seguintes iterações:

$$x^{(1)} = -0{,}70000, \tag{2.35}$$
$$x^{(2)} = -0{,}70959, \tag{2.36}$$
$$x^{(3)} = -0{,}71716, \tag{2.37}$$
$$\vdots \tag{2.38}$$
$$x^{(100)} = -0{,}77862, \tag{2.39}$$
$$\vdots \tag{2.40}$$
$$x^{(1000)} = -0{,}78466, \tag{2.41}$$
$$\vdots \tag{2.42}$$
$$x^{(20000)} = -0{,}78536. \tag{2.43}$$

Ou seja, neste caso as iterações de ponto fixo convergem (lentamente) para o ponto fixo $x = -\pi/4 \approx -0{,}78540$. Veja o código no `GNU Octave`.

Agora, se usarmos a iteração de ponto fixo com esta mesma função para aproximar o ponto fixo $x = 3\pi/4 \approx 2{,}3562$, obtemos

$$x^{(1)} = 2{,}50000, \tag{2.44}$$
$$x^{(2)} = 2{,}9966, \tag{2.45}$$
$$x^{(3)} = 5{,}8509, \tag{2.46}$$
$$\vdots \tag{2.47}$$
$$x^{(8)} = 4{,}8921e \times 10^{121}. \tag{2.48}$$

Donde observamos que as iterações divergem rapidamente.

Entretanto, se usarmos a iteração de ponto fixo com a função f_2 dada no Exemplo 2.3.1, obtemos

$$x^{(1)} = 2{,}50000, \tag{2.49}$$
$$x^{(2)} = 2{,}4155, \tag{2.50}$$
$$x^{(3)} = 2{,}3805, \tag{2.51}$$
$$\vdots \tag{2.52}$$

$$x^{(10)} = 2{,}3562. \tag{2.53}$$

A qual, portanto, converge para o ponto fixo esperado.

Este último exemplo (Exemplo 2.3.2) mostra que a iteração do ponto fixo nem sempre é convergente. Antes de vermos condições suficientes para a convergência, vejamos sua interpretação geométrica.

2.3.1 Interpretação geométrica

A Figura 2.6 apresenta o caso de uma iteração de ponto fixo convergente. As iterações iniciam-se no ponto $x^{(1)}$ e seguem para $x^{(2)} = g(x^{(1)})$ e $x^{(3)} = g(x^{(2)})$.

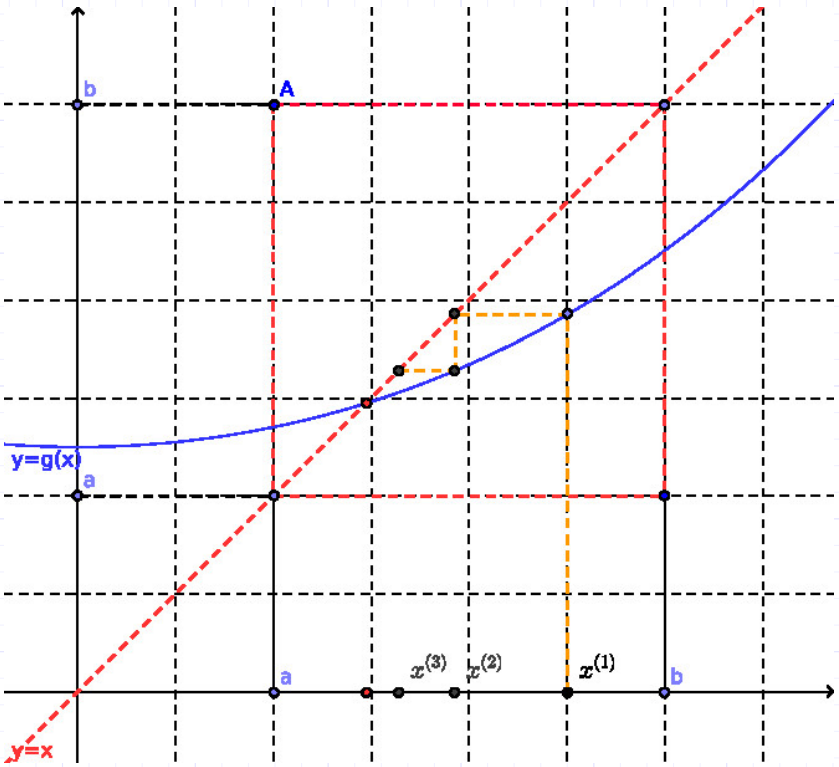

Figura 2.6: Interpretação geométrica da iteração de ponto fixo.

2.3.2 Análise de convergência

O seguinte teorema nos fornece condições suficientes para a convergência das iterações de ponto fixo.

Teorema 2.3.1. *(*Teorema do ponto fixo) Seja uma dada função g continuamente diferenciável satisfazendo

a) $g([a,b]) \subset [a,b]$,

b) $|g'(x)| < K < 1$ para todo $x \in [a,b]$.

Então, g tem um único ponto fixo $x^* \in [a,b]$ e as iterações $x^{(k+1)} = x^{(k)}$, $k = 1,2,3,\ldots$, convergem para x^*, para qualquer escolha de $x^{(1)} \in [a,b]$.

Demonstração. Da hipótese b), temos que g é uma contração com

$$|g(x) - g(y)| < K \cdot |x - y|, \ \forall x,y \in [a,b]. \tag{2.54}$$

Com isso, da hipótese a) e tomando $x^{(1)} \in [a,b]$, temos

$$\begin{aligned} |x^{(k+1)} - x^{(k)}| &= |g(x^{(k)}) - g(x^{(k-1)})| && (2.55)\\ &\leq K|x^{(k)} - x^{(k-1)}| && (2.56)\\ &\vdots && (2.57)\\ &\leq K^{k-1}|x^{(2)} - x^{(1)}|, && (2.58) \end{aligned}$$

para todo $k = 2,3,\ldots$. Como $K < 1$, temos $|x^{(k+1)} - x^{(k)}| \to 0$ quando $k \to \infty$ e, portanto, $x^{(k)}$ converge para algum $x^* \in [a,b]$.

De fato, x^* é ponto fixo de g, pois da continuidade da g, temos

$$x^* = \lim_{k\to\infty} x^{(k+1)} = \lim_{k\to\infty} g(x^{(k)}) = g(x^*). \tag{2.59}$$

Por fim, x^* é único, pois assumindo a existência de outro ponto fixo $x^{**} \neq x^*$ teríamos

$$|x^* - x^{**}| = |g(x^*) - g(x^{**})| < K|x^* - x^{**}|. \tag{2.60}$$

□

Agora, dado um problema de encontrar um zero de uma dada função f, i.e. $f(x) = 0$, podemos construir uma função g para a iteração de ponto fixo associada da seguinte forma:

$$f(x) = 0 \Leftrightarrow \underbrace{x - \alpha f(x)}_{g(x)} = x, \tag{2.61}$$

com $\alpha \in \mathbb{R}$ escolhido de forma a satisfazer as hipóteses do teorema do ponto fixo (Teorema 2.3.1).

Exemplo 2.3.3. Retornamos ao problema de encontrar o zero da função

$$f(x) = \text{sen}^2\left(x + \frac{\pi}{4}\right) - x^3 + \frac{\pi}{4}x^2 + \frac{5\pi^2}{16}x + \frac{3\pi^3}{64}. \tag{2.62}$$

no intervalo [2,3]. Para construir uma função g para a iteração de ponto fixo neste intervalo, podemos tomar

$$g(x) = x - \alpha f(x), \tag{2.63}$$

com $\alpha = -0{,}1$. A Figura 2.7 mostra esboços dos gráficos de g e $|g'|$ no intervalos $[2, 3]$ e podemos observar que esta escolha de α faz com que a g satisfaça o teorema do ponto fixo.

Então, fazendo as iterações de ponto fixo com aproximação inicial $x^{(1)} = 2{,}6$, obtemos os resultados apresentados na Tabela 2.4.

Tabela 2.4: Resultados referentes ao Exemplo 2.3.3.

k	$x^{(k)}$	$\lvert x^{(k)} - x^{(k-1)}\rvert$
1	2,6000	-x-
2	2,3264	2,7e−1
3	2,3553	2,9e−2
4	2,3562	8,4e−4
5	2,3562	1,1e−5

Os resultados apresentados na Tabela 2.4 podem ser computados no `GNU Octave` com o seguinte código:

```
f = @(x) sin(x+pi/4).^2-x.^3+pi/4*x.^2+5*pi^2/16*x+3*pi^3/64;
g = @(x,alpha) x - alpha*f(x);
```

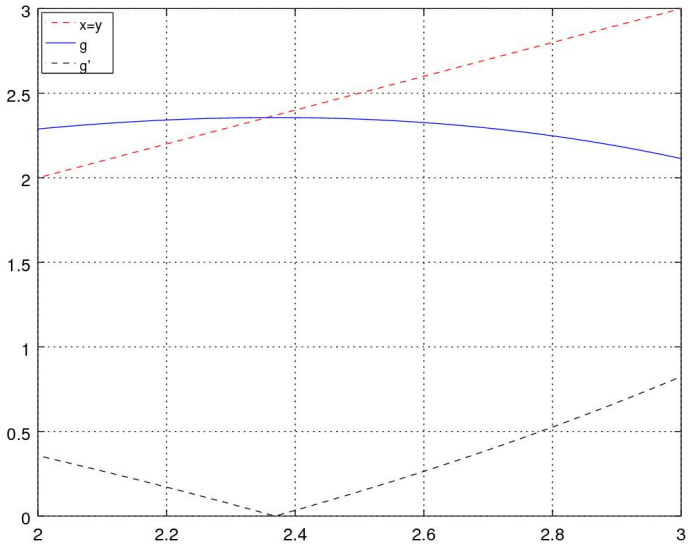

Figura 2.7: Esboço dos gráficos de g e $|g'|$ discutidas no Exemplo 2.3.3.

```
alpha=-0.1;

x=2.6
x0=x;
for k=2:5
  x=g(x0,alpha);
  printf("%d %1.4E %1.1E\n",...
          k,x,abs(x-x0))
  x0=x;
endfor
```

Observação 2.3.1. (Taxa de convergência) A iteração de ponto fixo tem taxa de convergência linear

$$|x^{(k+1)} - x^{(k)}| < K|x^{(k)} - x^{(k-1)}|^{\mathbf{1}}, \tag{2.64}$$

onde $K > 0$ é a constante dada na hipótese b) do teorema do ponto fixo (Teorema 2.3.1). Além disso, isso mostra que quanto menor o valor da constante K, mais rápida será a convergência das iterações de ponto fixo.

Exercícios

Exercício 2.3.1. Considere o problema de computar uma aproximação do zero de $f(x) = x - \cos(x)$. Resolva-o aplicando a iteração de ponto fixo para a função auxiliar

$$g(x) = x - \alpha f(x), \tag{2.65}$$

restrita ao intervalo $[a, b] = [0.5, 1]$ com aproximação inicial $x^{(1)} = (a + b)/2$. Escolha o melhor valor de α entre os seguintes:

1. $\alpha = 1$
2. $\alpha = 0{,}5$
3. $\alpha = -0{,}5$
4. $\alpha = 0{,}6$

Então, compute uma aproximação do zero de f com 5 dígitos significativos de precisão.

2.4 Método de Steffensen

O método de Steffensen[5] é uma aplicação do método de aceleração de convergência Δ^2 de Aitken[6] à iteração de ponto fixo.

2.4.1 Acelerador Δ^2 de Aitken

Seja dada uma sequência $(x^{(k)})_{k=1}^{\infty}$ monotonicamente convergente para x^*. Assumamos que k seja suficientemente grande tal que

$$\frac{x^{(k+1)} - x^*}{x^{(k)} - x^*} \approx \frac{x^{(k+2)} - x^*}{x^{(k+1)} - x^*}. \tag{2.66}$$

Então, isolando x^* obtemos

$$x^* \approx \frac{x^{(k)}x^{(k+2)} - (x^{(k+1)})^2}{x^{(k)} - 2x^{(k+1)} + x^{(k+2)}}. \tag{2.67}$$

[5] Johan Frederik Steffensen, matemático e estatístico dinamarquês, 1873 - 1961. Fonte: Wikipedia.

[6] Alexander Aitken, matemático neozelandês, 1895 - 1967. Fonte: Wikipedia.

Ainda, somando e subtraindo $(x^{(k)})^2$ e $2x^{(k)}x^{(k+1)}$ no numerador acima e rearranjando os termos, obtemos

$$x^* \approx x^{(k)} - \frac{(x^{(k+1)} - x^{(k)})^2}{x^{(k+2)} - 2x^{(k+1)} + x^{(k)}}. \tag{2.68}$$

O observado acima, nos motiva a introduzir o acelerador Δ^2 de Aitken

$$\Delta^2\{x^{(k)},x^{(k+1)},x^{(k+2)}\} := x^{(k)} - \frac{(x^{(k+1)} - x^{(k)})^2}{x^{(k+2)} - 2x^{(k+1)} + x^{(k)}}. \tag{2.69}$$

Exemplo 2.4.1. Consideremos o problema de encontrar o zero da função

$$f(x) = \text{sen}^2\left(x + \frac{\pi}{4}\right) - x^3 + \frac{\pi}{4}x^2 + \frac{5\pi^2}{16}x + \frac{3\pi^3}{64}. \tag{2.70}$$

no intervalo [2,3]. Para tanto, podemos aplicar a iteração de ponto fixo dada por

$$x^{(k+1)} = g(x^{(k)}) := x^{(k)} - \alpha f(x^{(k)}), \quad k = 1,2,\ldots, \tag{2.71}$$

com $\alpha = -0{,}05$ e $x^{(1)} = 2{,}6$. Na Tabela 2.5 temos os valores das iteradas $x^{(k)}$ e das correções $\Delta^2 = \Delta^2\{x^{(k)},x^{(k+1)},x^{(k+2)}\}$ de Aitken. Neste caso, a aceleração de convergência é notável.

Tabela 2.5: Resultados referentes ao Exemplo 2.4.1.

k	$x^{(k)}$	Δ^2
1	2,6000	-x-
2	2,4632	-x-
3	2,4073	2,3687
4	2,3814	2,3590
5	2,3688	2,3569
6	2,3625	2,3564
7	2,3594	2,3562
8	2,3578	2,3562

Os resultados apresentados na Tabela 2.5 podem ser computados no `GNU Octave` com o seguinte código:

```
f = @(x) sin(x+pi/4).^2-x.^3+pi/4*x.^2+5*pi^2/16*x+3*pi^3/64;

alpha=-0.05;
g = @(x) x - alpha*f(x);

x=2.6
printf("%d %1.4E\n",1,x)
x1=g(x);
printf("%d %1.4E\n",2,x1)
for k=3:8
  x2=g(x1);
  d=x-(x1-x)^2/(x2-2*x1+x);
  printf("%d %1.4E %1.4E\n",k,x2,d)

  x=x1;
  x1=x2;
endfor
```

2.4.2 Algoritmo de Steffensen

O método de Steffensen consiste em aplicar o acelerador Δ^2 de Aitken à iteração de ponto fixo. Mais especificamente, sejam uma aproximação inicial $x^{(1)}$ e uma iteração de ponto fixo

$$x^{(k+1)} = g(x^{(k)}), \quad k = 1,2,\ldots. \tag{2.72}$$

O algoritmo de Steffensen pode ser descrito como segue:

1. Fazemos $x = x^{(1)}$.
2. Para $k = 1,2,3,\ldots,N-1$:
   (a) Computamos $x_1 = g(x^{(k)})$.
   (b) Computamos $x_2 = g(x_1)$.
   (c) Computamos $x^{(k+1)} = \Delta^2\{x^{(k)},x_1,x_2\}$
   (d) Verificamos o critério de parada.

Exemplo 2.4.2. Retornemos ao exemplo anterior (Exemplo 2.4.1. Na Tabela 2.6 temos os valores das iteradas de Steffensen $x^{(k)}$ e do indicador de

convergência $|x^{(k)} - x^{(k-1)}|$. Observe que os resultados são compatíveis com aqueles obtidos no último exemplo.

Tabela 2.6: Resultados referentes ao Exemplo 2.4.2.

k	$x^{(k)}$	$\lvert x^{(k)} - x^{(k-1)}\rvert$
1	2,6000	-x-
2	2,3687	2,3e−1
3	2.3562	1,2e−2
4	2,3562	4,2e−5

Os resultados apresentados na Tabela 2.6 podem ser computados no `GNU Octave` com o seguinte código:

```
f = @(x) sin(x+pi/4).^2-x.^3+pi/4*x.^2+5*pi^2/16*x+3*pi^3/64;
alpha=-0.05;
g = @(x) x - alpha*f(x);


x0=2.6;
printf("%d %1.4E\n",1,x0)
for k=2:4
  x1=g(x0);
  x2=g(x1);
  x=x0-(x1-x0)^2/(x2-2*x1+x0);
  printf("%d %1.4E %1.1E\n",...
          k,x,abs(x-x0))
  x0=x;
endfor
```

Exercícios

Exercício 2.4.1. Use o método de Steffensen para obter uma aproximação do zero de $f(x) = x^3 \operatorname{sen}(x) - \cos(x)$ no intervalo $[0{,}5, 1]$ com precisão de 10^{-6}.

2.5 Método de Newton

Seja x^* um zero de uma dada função f, i.e. $f(x^*) = 0$. Usando de expansão em polinômio de Taylor da f em um dado ponto $\tilde{x}$, temos

$$f(x^*) = f(\tilde{x}) + f'(\tilde{x})(x^* - \tilde{x}) + O((x^* - \tilde{x})^2). \tag{2.73}$$

Como $f(x^*) = 0$, temos

$$x^* + O((x^* - \tilde{x})^2) = \tilde{x} - \frac{f(\tilde{x})}{f'(\tilde{x})}. \tag{2.74}$$

Esta última expressão nos indica que dada uma aproximação $\tilde{x}$ do zero de f a expressão

$$\tilde{x} - \frac{f(\tilde{x})}{f'(\tilde{x})}, \tag{2.75}$$

aproxima x^* com um erro da ordem de $(x^* - \tilde{x})^2$.

Estes observações nos levam a **iteração de Newton**[7]

$$x^{(1)} = \text{aprox. inicial}, \tag{2.76}$$

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}, \tag{2.77}$$

com $k = 1, 2, \ldots$.

Exemplo 2.5.1. Retornamos ao problema de encontrar o zero da função

$$f(x) = \text{sen}^2\left(x + \frac{\pi}{4}\right) - x^3 + \frac{\pi}{4}x^2 + \frac{5\pi^2}{16}x + \frac{3\pi^3}{64}. \tag{2.78}$$

no intervalo [2,3]. Então, fazendo as iterações de Newton com aproximação inicial $x^{(1)} = 2{,}6$, obtemos os resultados apresentados na Tabela 2.7.

Os resultados apresentados na Tabela 2.7 podem ser computados no `GNU Octave` com o seguinte código:

[7] Sir Isaac Newton, matemático e físico inglês, 1642 - 1726/27. Fonte: Wikipedia.

Tabela 2.7: Resultados referentes ao Exemplo 2.5.1.

k	$x^{(k)}$	$\lvert x^{(k)} - x^{(k-1)}\rvert$
1	2,6000	-x-
2	2,3836	2,2e−1
3	2,3566	2,7e−2
4	2,3562	3,9e−4
5	2,3562	8,3e−8

```
f = @(x) sin(x+pi/4).^2-x.^3+pi/4*x.^2+5*pi^2/16*x+3*pi^3/64;
fl = @(x) 2*sin(x+pi/4).*cos(x+pi/4)-3*x.^2+pi/2*x+5*pi^2/16;

x = 2.6
x0 = x;
for k=2:5
  x = x0 - f(x0)/fl(x0);
  printf("%d %1.4E %1.1E\n",...
         k,x,abs(x-x0))
  x0=x;
endfor
```

Observação 2.5.1. O método de Newton é uma iteração de ponto fixo ótima. Do Teorema do ponto fixo (Teorema 2.3.1), uma iteração de ponto fixo

$$x^{k+1} = g(x^{(k)}) := x^{(k)} - \alpha f(x) \tag{2.79}$$

tem taxa de convergência[8]

$$|x^{(k+1)} - x^{(k)}| \leq K|x^{(k)} - x^{(k-1)}|, \tag{2.80}$$

com K tal que $|g'(x)| = |1 - \alpha f'(x)| < K < 1$. Isto nos indica que a melhor escolha para α é

$$\alpha = \frac{1}{f'(x)}, \tag{2.81}$$

de forma que (2.79) coincide com a iteração de (2.77).

[8]Supondo as demais hipóteses do Teorema 2.3.1.

2.5.1 Interpretação geométrica

Dada uma aproximação $x^{(k)}$ de um zero de uma dada função f, a iteração de Newton fornece uma nova aproximação $x^{(k+1)}$ com

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}. \tag{2.82}$$

Subtraindo $x^{(k+1)}$ e multiplicando por $-f'(x^{(k)})$, obtemos

$$0 = f'(x^{(k)})(x^{(k+1)} - x^{(k)}) + f(x^{(k)}), \tag{2.83}$$

Observemos que o lado direito desta última equação corresponde a expressão da reta tangente ao gráfico de f pelo ponto $(x^{(k)}, f(x^{(k)}))$, avaliada em $x^{(k+1)}$. Mais precisamente, a equação desta reta tangente é

$$y = f'(x^{(k)})(x - x^{(k)}) + f(x^{(k)}) \tag{2.84}$$

e a equação (2.83) nos informa que em $x = x^{(k+1)}$ a reta tangente cruza o eixo x.

Destas observações, concluímos que a iterada $x^{(k+1)}$ do método de Newton corresponde ao ponto de interseção da reta tangente ao gráfico da f pelo ponto $(x^k, f(x^k))$. Veja a Figura 2.8.

Exemplo 2.5.2. Consideremos que o método de Newton seja usado para aproximarmos o zero de

$$f(x) = (x-1)e^{-x^2}. \tag{2.85}$$

Observemos que esta função tem $x = 1$ como seu único zero. Agora, se escolhermos $x^{(1)} = 0{,}5$ as iterações de Newton convergem para este zero, mas, se escolhermos $x^{(1)} = 1{,}5$ não (veja a Tabela 2.8).

Embora ambas aproximações iniciais estão a mesma distância da solução $x = 1$, quando tomamos $x^{(1)} = 1{,}5$ as iterações irão divergir, como podemos observar da interpretação geométrica dada na Figura 2.9.

2.5.2 Análise de convergência

Seja x^* o zero de uma dada função f duas vezes continuamente diferenciável com $f'(x) \neq 0$ para todo $x \in [x^* - \varepsilon_0, x^* + \varepsilon_0]$ para algum $\varepsilon_0 > 0$. Seja,

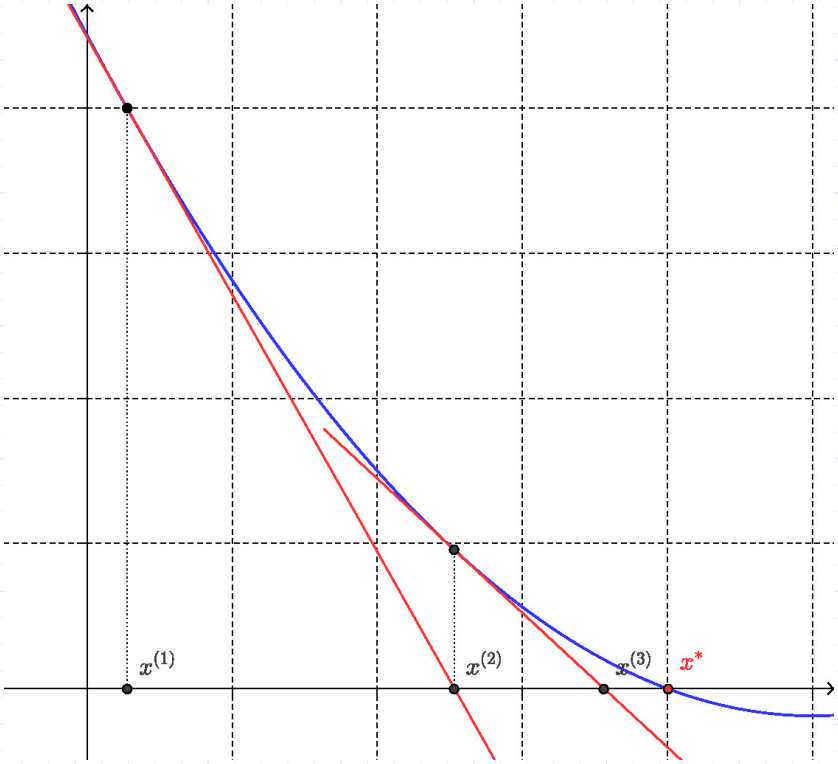

Figura 2.8: Interpretação geométrica das iterações de Newton. Veja no Geogebra.

Tabela 2.8: Resultados referentes ao Exemplo 2.5.2

k	$x^{(k)}$	$x^{(k+1)}$
1	5,0000e−1	1,5000e+0
2	8,3333e−1	2,5000e+0
3	9,6377e−1	2,7308e+0
4	9,9763e−1	2,9355e+0
5	9,9999e−1	3,1223e+0
6	1,0000e+0	3,2955e+0
7	1,0000e+0	3,4580e+0

também, $(x^{(k)})_{k=1}^{\infty}$ a sequência das iteradas de Newton

$$x^{(k+1)} = x^{(k)} - \frac{f(x^{(k)})}{f'(x^{(k)})}, \quad k = 1, 2, \ldots, \tag{2.86}$$

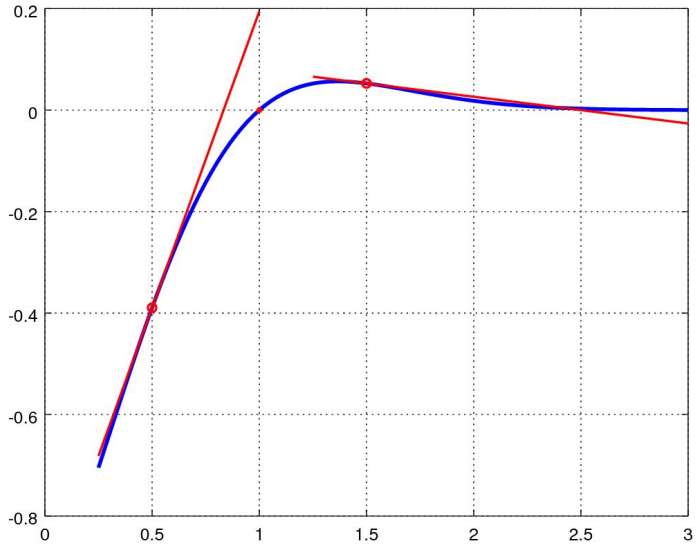

Figura 2.9: Escolha da aproximação inicial para o método de Newton.

com aproximação inicial $x^{(1)} \in (x^* - \varepsilon_0, x^* + \varepsilon_0)$. Então, do polinômio de Taylor de grau 1 de f em torno de $x^{(1)}$, temos

$$f(x^*) = f(x^{(1)}) + f'(x^{(1)})(x^* - x^{(1)}) + \frac{f''(\xi^{(1)})}{2!}(x^* - x^{(1)})^2, \tag{2.87}$$

onde $\xi^{(1)}$ está entre $x^{(1)}$ e ξ. Daí, rearranjamos os termos e notamos que $f(x^*) = 0$ para obtermos

$$x^* - \left[x^{(1)} - \frac{f(x^{(1)})}{f'(x^{(1)})}\right] = \frac{f''(\xi^{(1)})}{2f'(x^{(1)})}(x^* - x^{(1)})^2. \tag{2.88}$$

Então, da iteração de Newton (2.86), temos

$$x^* - x^{(2)} = \frac{f''(\xi^{(1)})}{2f'(x^{(1)})}(x^* - x^{(1)})^2 \tag{2.89}$$

Logo,

$$|x^* - x^{(2)}| \leq C|x^* - x^{(1)}|^2, \tag{2.90}$$

com

$$C = \sup_{x,y\in[x^*-\varepsilon,x^*+\varepsilon]} \left|\frac{f''(x)}{2f'(y)}\right|. \tag{2.91}$$

Segue, então, que se $x^{(1)} \in (x^* - \varepsilon, x^* + \varepsilon)$ para algum $\varepsilon > 0$ tal que

$$C|x^* - x|^2 < 1, \quad \forall x \in (x^* - \varepsilon, x^* + \varepsilon), \tag{2.92}$$

então $x^{(2)} \in (x^* - \varepsilon, x^* + \varepsilon)$.

Logo, por indução matemática[9], temos que o método de Newton tem taxa de convergência quadrática

$$|x^{(k+1)} - x^*| \leq C|x^{(k)} - x^*|^{\mathbf{2}}, \tag{2.93}$$

para qualquer escolha de $x^{(1)}$ suficientemente próximo de x^*, i.e. $x^{(1)} \in (x^* - \varepsilon, x^* + \varepsilon)$.

Observação 2.5.2. O intervalo $(x^* - \varepsilon, x^* + \varepsilon)$ é chamado de **bacia de atração** do método de Newton.

Exemplo 2.5.3. Retornamos ao problema de encontrar o zero da função

$$f(x) = \operatorname{sen}^2\left(x + \frac{\pi}{4}\right) - x^3 + \frac{\pi}{4}x^2 + \frac{5\pi^2}{16}x + \frac{3\pi^3}{64}. \tag{2.94}$$

no intervalo [2,3]. Este problema foi construído de forma que $x^* = 3\pi/4$ é um zero de f. Então, fazendo as iterações de Newton com aproximação inicial $x^{(1)} = 2{,}6$, obtemos os resultados apresentados na Tabela 2.9, os quais evidenciam a convergência quadrática das iterações computadas.

Os resultados apresentados na Tabela 2.9 podem ser computados no `GNU Octave` com o seguinte código:

```
f = @(x) sin(x+pi/4).^2-x.^3+pi/4*x.^2+5*pi^2/16*x+3*pi^3/64;
fl = @(x) 2*sin(x+pi/4).*cos(x+pi/4)-3*x.^2+pi/2*x+5*pi^2/16;

xstar = 3*pi/4;

x = 2.6
```

[9] Veja o exercício 2.5.3.

Tabela 2.9: Resultados referentes ao Exemplo 2.5.3.

k	$x^{(k)}$	$\lvert x^{(k)} - x^* \rvert$
1	2,6000	2,4e−01
2	2,3836	2,7e−02
3	2,3566	3,9e−04
4	2,3562	8,3e−08
5	2,3562	3,6e−15

```
printf("%d %1.4E %1.1E\n",...
          1,x,abs(x-xstar))
for k=2:5
  x = x - f(x)/fl(x);
  printf("%d %1.4E %1.1E\n",...
          k,x,abs(x-xstar))
endfor
```

2.5.3 Zeros múltiplos

Na análise de convergência acima foi necessário assumir que $f'(x) \neq 0$ para todo x em uma vizinha do zero x^* da função f. Isto não é possível no caso de x^* ser um zero duplo pois, então, $f'(x^*) = 0$. Neste caso, podemos aplicar o método de Newton a $f'(x)$, a qual tem x^* como um zero simples.

Exemplo 2.5.4. Consideremos o problema de aproximar o zero da função

$$f(x) = \operatorname{sen}^2\left(x + \frac{\pi}{4}\right) - x^3 + \frac{\pi}{4}x^2 + \frac{5\pi^2}{16}x + \frac{3\pi^3}{64}. \tag{2.95}$$

no intervalo $[-1{,}0]$. Este problema foi construído de forma que $x^* = -\pi/4$ é um zero duplo de f. Então, aplicamos o método de Newton a

$$f'(x) = 2\operatorname{sen}\left(x + \frac{\pi}{4}\right)\cos\left(x + \frac{\pi}{4}\right) - 3x\check{s} + \frac{\pi}{2}x + \frac{5\pi^2}{16}. \tag{2.96}$$

Ou seja, as iterações de Newton são

$$x^{(k+1)} = x^{(k)} - \frac{f'(x^{(k)})}{f''(x^{(k)})}, \quad k = 1{,}2,\ldots, \tag{2.97}$$

Tabela 2.10: Resultados referentes ao Exemplo 2.5.4.

k	$x^{(k)}$	$\lvert x^{(k)} - x^* \rvert$
1	−5,0000e−1	2,9e−01
2	−8,3407e−1	4,9e−02
3	−7,8619e−1	7,9e−04
4	−7,8540e−1	2,3e−07
5	−7,8540e−1	1,9e−14

sendo $x^{(1)}$ uma aproximação inicial. Na Tabela 2.10, temos os resultados obtidos da computação destas iterações com $x^{(1)} = -0{,}5$.

Os resultados apresentados na Tabela 2.10 podem ser computados no `GNU Octave` com o seguinte código:

```
f = @(x) sin(x+pi/4).^2-x.^3+pi/4*x.^2+5*pi^2/16*x+3*pi^3/64;
fl = @(x) 2*sin(x+pi/4).*cos(x+pi/4)-3*x.^2+pi/2*x+5*pi^2/16;
fll = @(x) 2*cos(x+pi/4)^2-2*sin(x+pi/4)^2-6*x+pi/2;

xstar = -pi/4;

x = -0.5;
printf("%d %1.4E %1.1E\n",...
          1,x,abs(x-xstar))
for k=2:5
  x = x - fl(x)/fll(x);
  printf("%d %1.4E %1.1E\n",...
          k,x,abs(x-xstar))
endfor
```

Observação 2.5.3. No caso de zeros de multiplicidade n de uma dada função f, podemos aplicar o método de Newton a derivada $n-1$ de f.

Exercícios

Exercício 2.5.1. Use o método de Newton para obter uma aproximação do zero de $f(x) = x^3 \operatorname{sen}(x) - \cos(x)$ no intervalo $[0{,}5, 1]$ com precisão de 10^{-6}.

Exercício 2.5.2. Use o método de Newton para obter uma aproximação do zero de

$$f(x) = (-x^2 + 1{,}154x - 0{,}332929)\cos(x) + x^2 - 1{,}154x + 0{,}332929 \quad (2.98)$$

no intervalo $(0{,}55,\ 0{,}65)$ com precisão de 10^-5.

Exercício 2.5.3. Complete a demonstração por indução matemática de que o método de Newton tem taxa de convergência quadrática.

2.6 Método da secante

O método da secante é uma variação do método de Newton. Observemos que para duas aproximações $x^{(k)}$ e $x^{(k-1)}$ suficientemente próximas, temos[10]

$$f'(x^{(k)}) \approx \frac{f(x^{(k)}) - f(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}. \quad (2.99)$$

Assim sendo, substituindo esta aproximação em (2.77), obtemos as iterações do método da secante:

$$x^{(1)}, x^{(2)} = \text{aprox. iniciais}, \quad (2.100)$$

$$x^{(k+1)} = x^{(k)} - f(x^{(k)})\frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})}, \quad (2.101)$$

para $k = 2{,}3, \ldots$.

Exemplo 2.6.1. Consideremos o problema de encontrar o zero da função

$$f(x) = \text{sen}^2\left(x + \frac{\pi}{4}\right) - x^3 + \frac{\pi}{4}x^2 + \frac{5\pi^2}{16}x + \frac{3\pi^3}{64}. \quad (2.102)$$

no intervalo [2,3]. Então, fazendo as iterações do método da secante com aproximações iniciais $x^{(1)} = 2{,}6$ e $x^{(2)} - 2{,}5$, obtemos os resultados apresentados na Tabela 2.11.

Os resultados apresentados na Tabela 2.11 podem ser computados no `GNU Octave` com o seguinte código:

[10] Razão fundamental do Cálculo.

Tabela 2.11: Resultados referentes ao Exemplo 2.6.1.

k	$x^{(k-1)}$	$x^{(k)}$	$\lvert x^{(k)} - x^{(k-1)}\rvert$
2	2,6000	2,5000	-x-
3	2,5000	2.3728	1,3e−1
4	2,3728	2,3574	1,5e−2
5	2,3574	2,3562	1,2e−3
6	2,3562	2,3562	1,1e−5
7	2,3562	2,3562	7,0e−9

```
f = @(x) sin(x+pi/4).^2-x.^3+pi/4*x.^2+5*pi^2/16*x+3*pi^3/64;

x1 = 2.6;
x2 = 2.5;
printf("%d %1.4E %1.4E\n",
      1,x1,x2)
for k=3:7
  x = x2 - f(x2)*(x2-x1)/(f(x2)-f(x1));
  printf("%d %1.4E %1.4E %1.1E\n",...
          k,x2,x,abs(x-x2))
  x1=x2;
  x2=x;
endfor
```

2.6.1 Interpretação geométrica

A iteração do método da secante é

$$x^{(k+1)} = x^{(k)} - f(x^{(k)})\frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})}, \tag{2.103}$$

donde segue que

$$0 = x^{(k+1)} - x^{(k)} + f(x^{(k)})\frac{x^{(k)} - x^{(k-1)}}{f(x^{(k)}) - f(x^{(k-1)})}, \tag{2.104}$$

bem como que

$$0 = \frac{f(x^{(k)}) - f(x^{(k-1)})}{x^{(k)} - x^{(k-1)}}(x^{(k+1)} - x^{(k)}) + f(x^{(k)}). \tag{2.105}$$

Agora, observemos que

$$y = \frac{f(x^{(k)}) - f(x^{(k-1)})}{x - x^{(k-1)}}(x^{(k+1)} - x^{(k)}) + f(x^{(k)}). \tag{2.106}$$

é a equação da reta secante ao gráfico de f pelos pontos $(x^{(k)}, f(x^{(k)}))$ e $(x^{(k-1)}, f(x^{(k-1)}))$.

Do observado acima, temos que (2.105) nos mostra que a iterada $x^{(k+1)}$ é a a interseção do eixo x com a reta secante ao gráfico de f pelos pontos $(x^{(k)}, f(x^{(k)}))$ e $(x^{(k-1)}, f(x^{(k-1)}))$. Veja a Figura 2.10.

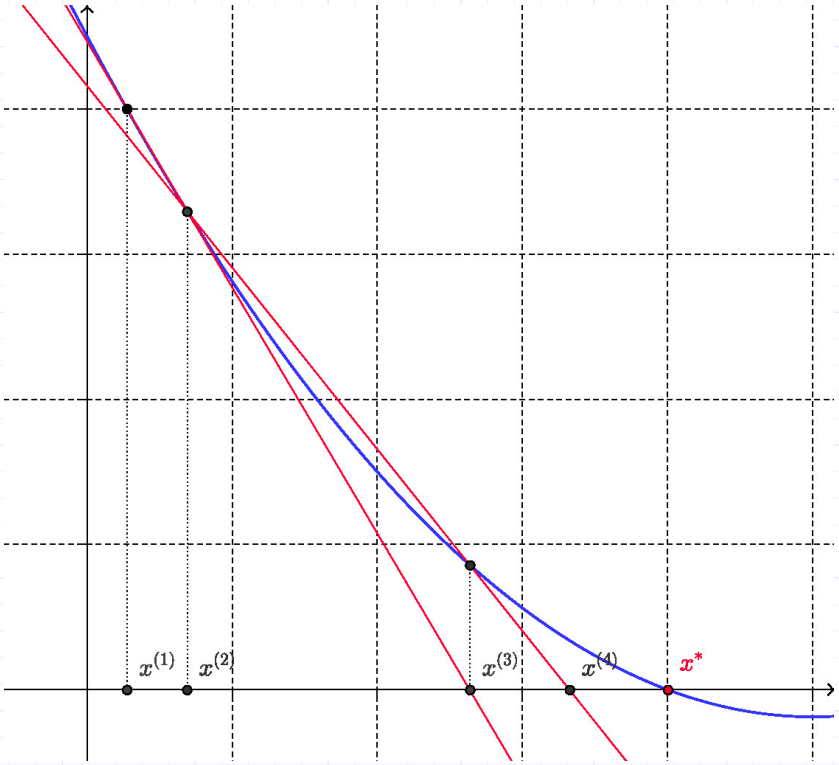

Figura 2.10: Interpretação geométrica das iterações do método da secante. Veja no Geogebra.

Observação 2.6.1. A interpretação geométrica do método da secante pode nos ajudar a escolher as aproximações iniciais $x^{(1)}$ e $x^{(2)}$. Como uma boa prática, escolhemo-las próximas do zero (por inspeção gráfica), tomando $x^{(2)}$ como uma aproximação melhor que $x^{(1)}$.

2.6.2 Análise de convergência

Pode-se mostrar[11] que a taxa de convergência do método da secante é super-linear com

$$|x^{(k+1)} - x^*| \leq C|x^{(k)} - x^*|^{\varphi}, \tag{2.107}$$

onde $\varphi = (1+\sqrt{5})/2 \approx 1{,}618$ (razão áurea) e x^* é o zero de f.

Exemplo 2.6.2. Consideremos o problema de encontrar o zero da função

$$f(x) = \operatorname{sen}^2\left(x + \frac{\pi}{4}\right) - x^3 + \frac{\pi}{4}x^2 + \frac{5\pi^2}{16}x + \frac{3\pi^3}{64}. \tag{2.108}$$

no intervalo [2,3]. Este problema foi construído de forma que $x^* = 3\pi/4$ é um zero de f. Agora, fazendo as iterações do método da secante com aproximações iniciais $x^{(1)} = 2{,}6$ e $x^{(2)} = 2{,}5$, obtemos os resultados apresentados na Tabela 2.12.

Tabela 2.12: Resultados referentes ao Exemplo 2.6.2.

k	$x^{(k)}$	$\lvert x^{(k)} - x^*\rvert$
3	2,3728	1,7e−02
4	2,3574	1,2e−03
5	2,3562	1,1e−05
6	2,3562	7,0e−09

Os resultados apresentados na Tabela 2.12 podem ser computados no `GNU Octave` com o seguinte código:

```
f = @(x) sin(x+pi/4).^2-x.^3+pi/4*x.^2+5*pi^2/16*x+3*pi^3/64;

xstar = 3*pi/4;

phi = (1+sqrt(5))/2

x1 = 2.6;
x2 = 2.5;
for k=3:10
```

[11] Veja, por exemplo, Seção 3.5.2 do livro "Cálculo Numérico" do projeto REAMAT.

```
  x = x2 - f(x2)*(x2-x1)/(f(x2)-f(x1));
  printf("%d %1.4E %1.1E\n",...
         k,x,abs(x-xstar))
  x1=x2;
  x2=x;
endfor
```

Observação 2.6.2. (Zeros de multiplicidade par.) A taxa de convergência super-linear do método da secante não se mantém para o caso de x^* ser um zero múltiplo. Para contornar este problema, pode-se aplicar o método à derivada $n-1$ de f, a fim de se aproximar um zero de multiplicidade n.

Observação 2.6.3. (Cancelamento catastrófico.) Conforme convergem as iterações do método da secante, o denominador $f(x^{(k)}) - f(x^{(k-1)})$ pode convergir rapidamente para zero, ocasionando uma divisão por zero.

Exercícios

Exercício 2.6.1. Use o método da secante para obter uma aproximação do zero de $f(x) = x^3 \operatorname{sen}(x) - \cos(x)$ no intervalo $[0{,}5, 1]$ com precisão de 10^{-5}.

Exercício 2.6.2. Use o método da secante para obter uma aproximação do zero de

$$f(x) = (-x^2 + 1{,}154x - 0{,}332929)\cos(x) + x^2 - 1{,}154x + 0{,}332929 \quad (2.109)$$

no intervalo $(0{,}55,\ 0{,}65)$ com precisão de 10^-5.

2.7 Raízes de polinômios

Nesta seção, veremos como o método de Newton pode ser aplicado de forma robusta a polinômios com o auxílio do método de Horner[12]. A questão central é que dado um polinômio de grau n escrito na forma

$$p(x) = a_1 x^n + a_2 x^{n-1} + \cdots + a_n x + a_{n+1}, \quad (2.110)$$

[12] William George Horner, matemático britânico, 1786 - 1837. Fonte: Wikipedia

o procedimento de avaliá-lo em um ponto requer $n!n$ multiplicações e n adições. Desta forma, a aplicação do método de Newton para obter as raízes de p torna-se computacionalmente custosa, por requer a avaliação de p e de sua derivada a cada iteração. Agora, o método de Horner nos permite avaliar p em um ponto qualquer usando apenas n multiplicações e n adições, reduzindo enormemente o custo computacional.

2.7.1 Método de Horner

Sejam dados um número x_0 e um polinômio de grau n

$$p(x) = p_1 x^n + p_2 x^{n-1} + \cdots + p_n x + p_{n+1} s. \tag{2.111}$$

O método de Horner consiste em computar $p(x_0)$ pela iteração

$$q_1 = p_1, \tag{2.112}$$
$$q_k = p_k + q_{k-1} x_0, \quad k = 2,3,\ldots,n+1, \tag{2.113}$$

sendo que, então, $p(x_0) = q_{n+1}$ e, além disso,

$$p(x) = (x - x_0) q(x) + q_{n+1}, \tag{2.114}$$

com

$$q(x) = q_1 x^{n-1} + q_2 x^{n-2} + \cdots + q_{n-1} x + q_n. \tag{2.115}$$

De fato, a verificação de (2.114) é direta, uma vez que

$$\begin{aligned}(x - x_0) q(x) + q_{n+1} &= (x - x_0)(q_1 x^{n-1} + q_2 x^{n-2} + \cdots + q_{n-1} x + q_n) \\ &\quad + q_{n+1} \qquad (2.116)\\ &= q_1 x^n + (q_2 - q_1 x_0)^{n-1} + \cdots + (q_{n+1} - q_n x_0). \qquad (2.117)\end{aligned}$$

E, então, igualando a $p(x)$ na forma (2.111), temos as equações (2.112)-(2.113).

Exemplo 2.7.1. Consideremos o polinômio

$$p(x) = x^3 - 3x^2 + 4. \tag{2.118}$$

Para computarmos $p(1)$ pelo método de Horner, tomamos $x_0 = 1$, $q_1 = p_1 = 1$ e

$$q_2 = p_2 + q_1 x_0 = -3 + 1 \cdot 1 = -2 \tag{2.119}$$
$$q_3 = p_3 + q_2 x_0 = 0 + (-2) \cdot 1 = -2 \tag{2.120}$$
$$q_4 = p_4 + q_3 x_0 = 4 + (-2) \cdot 1 = 2. \tag{2.121}$$

Com isso, temos $p(3) = q_4 = 4$ (verifique!).

Para este caso, podemos implementar o método de Horner no `GNU Octave` com o seguinte código:

```
p = [1 -3 0 4];
x0=1;

q = zeros(4,1);
q(1)=p(1);
for k=2:4
  q(k)=p(k)+q(k-1)*x0;
endfor
q(4)
```

Observação 2.7.1. Ao computarmos $p(x_0)$ pelo método de Horner, obtemos a decomposição

$$p(x) = (x - x_0)q(x) + b_{n+1}. \tag{2.122}$$

Desta forma, temos

$$p'(x) = q(x) + (x - x_0)q'(x), \tag{2.123}$$

donde temos que $p'(x_0) = q(x_0)$. Com isso, para computarmos $p'(x_0)$ podemos aplicar o método de Horner a $q(x)$.

A implementação do método de Horner no `GNU Octave` para computar $p(x_0)$ e $p'(x_0)$ pode ser feita com o seguinte código:

```
function [y,dy] = Horner(p,x0)
  n = length(p);
  y =p(1);
  dy=y;
```

```
  for k=2:n-1
    y=p(k)+y*x0;
    dy=y+dy*x0;
  endfor
  y=p(n)+y*x0;
endfunction
```

2.7.2 Método de Newton-Horner

A implementação do método de Newton a polinômios pode ser feita de forma robusta com o auxílio do método de Horner. Dado um polinômio p e uma aproximação inicial $x^{(1)}$ para uma de suas raízes reais, a iteração de Newton consiste em

$$x^{(k+1)} = x^{(k)} - \frac{p(x^{(k)})}{p'(x^{(k)})}, \tag{2.124}$$

na qual podemos utilizar o método de Horner para computar $p(x^{(k)})$ e $p'(x^{(k)})$.

Exemplo 2.7.2. Consideremos o caso de aplicar o método de Newton para obter uma aproximação da raiz $x = -1$ de

$$p(x) = x^3 - 3x^2 + 4, \tag{2.125}$$

com aproximação inicial $x^{(1)} = -2$. Na Tabela 2.13 temos os resultados obtidos.

Tabela 2.13: Resultados referentes ao Exemplo 2.7.2.

k	$x^{(k)}$	$\lvert x^{(k)} - x^{(k-1)}\rvert$
1	$-2{,}0000$	-x-
2	$-1{,}3333$	$6{,}7\text{e}{-1}$
3	$-1{,}0556$	$2{,}8\text{e}{-1}$
4	$-1{,}0019$	$5{,}4\text{e}{-2}$
5	$-1{,}0000$	$1{,}9\text{e}{-3}$

No `GNU Octave`, a implementação do método de Newton para polinômios (com a aplicação do método de Horner, veja Observação 2.7.1) pode ser feita com o seguinte código:

```
p = [1 -3 0 4];

x0=-2;
printf("%d %1.4E\n",1,x0)
for k=2:5
  [y,dy]=Horner(p,x0);
  x=x0-y/dy;
  printf("%d %1.4E %1.1E\n",k,x,abs(x-x0))
  x0=x;
endfor
```

Exercícios

Exercício 2.7.1. Use o método de Newton-Horner para computar a aproximação da raiz de $p(x) = x^3 - 3x^2 + 4$ no intervalo [1,3]. Observe que p tem um zero duplo deste intervalo.

Capítulo 3

Métodos diretos para sistemas lineares

Neste capítulo, discutiremos sobre métodos diretos para a resolução de sistemas lineares de n-equações com n-incógnitas. Isto é, sistemas que podem ser escritos na seguinte forma algébrica

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \tag{3.1}$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \tag{3.2}$$
$$\vdots \tag{3.3}$$
$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n. \tag{3.4}$$

3.1 Eliminação gaussiana

Um sistema linear

$$a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n = b_1 \tag{3.5}$$
$$a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n = b_2 \tag{3.6}$$
$$\vdots \tag{3.7}$$
$$a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n = b_n. \tag{3.8}$$

pode ser escrito na forma matricial

$$A\boldsymbol{x} = \boldsymbol{b}, \tag{3.9}$$

onde $A = [a_{ij}]_{i,j=1}^{n,n}$ é chamada de matriz dos coeficientes, $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ é o vetor (coluna) das incógnitas e $\boldsymbol{b} = (b_1, b_2, \ldots, b_n)$ é o vetor (coluna) dos termos constantes.

Outra forma matricial de representar o sistema (3.5)-(3.8) é pela chamada matriz estendida

$$E = [A\ \boldsymbol{b}]. \tag{3.10}$$

No caso, E é a seguinte matriz $n \times (n+1)$

$$E = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2n} & b_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & b_n \end{bmatrix} \tag{3.11}$$

O método de eliminação gaussiana consistem em realizar operações sobre as equações (sobre as linhas) do sistema (3.5)-(3.8) (da matriz estendida E) de forma a reescrevê-lo como um sistema triangular, ou diagonal. Para tanto, podemos utilizar as seguintes operações:

1. permutação entre as equações (linhas) ($E_i \leftrightarrow E_j$).
2. multiplicação de uma equação (linha) por um escalar não nulo ($E_i \leftarrow \lambda E_i$).
3. substituição de uma equação (linha) por ela somada com a multiplicação de uma outra por um escalar não nulo ($E_i \leftarrow E_i + \lambda E_j$).

Exemplo 3.1.1. O sistema linear

$$-2x_1 - 3x_2 + 2x_3 + 3x_4 = 10 \tag{3.12}$$
$$-4x_1 - 6x_2 + 6x_3 + 6x_4 = 20 \tag{3.13}$$
$$-2x_1 + 4x_3 + 6x_4 = 10 \tag{3.14}$$
$$4x_1 + 3x_2 - 8x_3 - 6x_4 = -17 \tag{3.15}$$

pode ser escrito na forma matricial $A\boldsymbol{x} = \boldsymbol{b}$, onde

$$A = \begin{bmatrix} -2 & -3 & 2 & 3 \\ -4 & -6 & 6 & 6 \\ -2 & 0 & 4 & 6 \\ 4 & 3 & -8 & -6 \end{bmatrix}, \tag{3.16}$$

$\boldsymbol{x} = (x_1, x_2, x_3, x_4)$ e $\boldsymbol{b} = (10, 20, 10, -17)$. Sua matriz estendida é

$$E = \begin{bmatrix} -2 & -3 & 2 & 3 & 10 \\ -4 & -6 & 6 & 6 & 20 \\ -2 & 0 & 4 & 6 & 10 \\ 4 & 3 & -8 & -6 & -17 \end{bmatrix} \tag{3.17}$$

Então, usando o método de eliminação gaussiana, temos

$$E = \begin{bmatrix} -2 & -3 & 2 & 3 & 10 \\ -4 & -6 & 6 & 6 & 20 \\ -2 & 0 & 4 & 6 & 10 \\ 4 & 3 & -8 & -6 & -17 \end{bmatrix} E_2 \leftarrow E_2 - (e_{21}/\boldsymbol{e_{11}})E_1 \tag{3.18}$$

$$\sim \begin{bmatrix} \mathbf{-2} & -3 & 2 & 3 & 10 \\ 0 & 0 & 2 & 0 & 0 \\ -2 & 0 & 4 & 6 & 10 \\ 4 & 3 & -8 & -6 & -17 \end{bmatrix} E_3 \leftarrow E_3 - (e_{31}/\boldsymbol{e_{11}})E_1 \tag{3.19}$$

$$\sim \begin{bmatrix} \mathbf{-2} & -3 & 2 & 3 & 10 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 3 & 2 & 3 & 0 \\ 4 & 3 & -8 & -6 & -17 \end{bmatrix} E_4 \leftarrow E_4 - (e_{41}/\boldsymbol{e_{11}})E_1 \tag{3.20}$$

$$\sim \begin{bmatrix} \mathbf{-2} & -3 & 2 & 3 & 10 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & \mathbf{3} & 2 & 3 & 0 \\ 0 & -3 & -4 & 0 & 3 \end{bmatrix} E_2 \leftrightarrow E_3 \tag{3.21}$$

$$\sim \begin{bmatrix} \mathbf{-2} & -3 & 2 & 3 & 10 \\ 0 & \mathbf{3} & 2 & 3 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & -3 & -4 & 0 & 3 \end{bmatrix} E_4 \leftarrow E_4 - (e_{42}/\boldsymbol{e_{22}})E_2 \tag{3.22}$$

$$\sim \begin{bmatrix} \mathbf{-2} & -3 & 2 & 3 & 10 \\ 0 & 3 & 2 & 3 & 0 \\ 0 & 0 & \mathbf{2} & 0 & 0 \\ 0 & 0 & -2 & 3 & 3 \end{bmatrix} E_4 \leftarrow E_4 - (e_{43}/\boldsymbol{e_{33}})E_3 \tag{3.23}$$

$$\sim \begin{bmatrix} \mathbf{-2} & -3 & 2 & 3 & 10 \\ 0 & \mathbf{3} & 2 & 3 & 0 \\ 0 & 0 & \mathbf{2} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{3} & 3 \end{bmatrix} \tag{3.24}$$

Esta última matriz estendida é chamada de **matriz escalonada** do sistema. Desta, temos que (3.12)-(3.15) é equivalente ao seguinte sistema triangular

$$-2x_1 - 3x_2 + 2x_3 + 3x_4 = 10 \tag{3.25}$$
$$3x_2 + 2x_3 + 3x_4 = 0 \tag{3.26}$$
$$2x_3 = 0 \tag{3.27}$$
$$3x_4 = 3. \tag{3.28}$$

Resolvendo da última equação para a primeira, temos

$$x_4 = 1, \tag{3.29}$$
$$x_3 = 0, \tag{3.30}$$
$$x_2 = \frac{-2x_3 - 3x_4}{3} = -1, \tag{3.31}$$
$$x_1 = \frac{10 + 3x_2 - 3x_3 - 3x_4}{-2} = -2. \tag{3.32}$$

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

```
A = [-2 -3 2 3; ...
     -4 -6 6 6; ...
     -2 0 4 6; ...
     4 3 -8 -6]
b = [10 20 10 -17]'

E = [A b]

E(2,:) = E(2,:) - E(2,1)/E(1,1)*E(1,:)
E(3,:) = E(3,:) - E(3,1)/E(1,1)*E(1,:)
E(4,:) = E(4,:) - E(4,1)/E(1,1)*E(1,:)

aux = E(2,:);
E(2,:) = E(3,:);
E(3,:) = aux

E(4,:) = E(4,:) - E(4,2)/E(2,2)*E(2,:)
```

```
E(4,:) = E(4,:) - E(4,3)/E(3,3)*E(3,:)

x = zeros(4,1);
x(4) = E(4,5)/E(4,4);
x(3) = (E(3,5) - E(3,4)*x(4))/E(3,3);
x(2) = (E(2,5) - E(2,3)*x(3) ...
        - E(2,4)*x(4))/E(2,2);
x(1) = (E(1,5) - E(1,2)*x(2) ...
        - E(1,3)*x(3) - E(1,4)*x(4))/E(1,1)
```

Observação 3.1.1. Para a resolução de um sistema linear $n \times n$, o método de eliminação gaussiana demanda

$$\frac{n^3}{3} + n^2 - \frac{n}{3} \tag{3.33}$$

multiplicações/divisões e

$$\frac{n^3}{3} + \frac{n^2}{2} - \frac{5n}{6} \tag{3.34}$$

adições/subtrações [2].

Com o mesmo custo computacional, podemos utilizar o método de eliminação gaussiana para transformar o sistema dado em um sistema diagonal.

Exemplo 3.1.2. Voltando ao exemplo anterior (Exemplo 3.1.1, vimos que a matriz estendida do sistema (3.12)-(3.15) é equivalente a

$$E \sim \begin{bmatrix} -2 & -3 & 2 & 3 & 10 \\ 0 & 3 & 2 & 3 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 3 & 3 \end{bmatrix}. \tag{3.35}$$

Então, podemos continuar aplicando o método de eliminação gaussiana, agora de baixo para cima, até obtermos um sistema diagonal equivalente. Vejamos

$$E \sim \begin{bmatrix} -2 & -3 & 2 & 3 & 10 \\ 0 & 3 & 2 & 3 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{3} & 3 \end{bmatrix} \begin{array}{l} E_1 \leftarrow E_1 - (e_{14}/e_{44})E_4 \\ E_2 \leftarrow E_2 - (e_{24}/e_{44})E_4 \end{array} \tag{3.36}$$

$$\sim \begin{bmatrix} -2 & -3 & 2 & 0 & 7 \\ 0 & 3 & 2 & 0 & -3 \\ 0 & 0 & \mathbf{2} & 0 & 0 \\ 0 & 0 & 0 & 3 & 3 \end{bmatrix} \begin{array}{l} E_1 \leftarrow E_1 - (e_{13}/e_{33})E_3 \\ E_2 \leftarrow E_2 - (e_{23}/e_{33})E_3 \end{array} \tag{3.37}$$

$$\sim \begin{bmatrix} -2 & -3 & 0 & 0 & 4 \\ 0 & \mathbf{3} & 0 & 0 & -3 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 3 & 3 \end{bmatrix} \begin{array}{l} E_1 \leftarrow E_1 - (e_{12}/e_{22})E_2 \end{array} \tag{3.38}$$

$$\sim \begin{bmatrix} \mathbf{-2} & 0 & 0 & 0 & 4 \\ 0 & \mathbf{3} & 0 & 0 & -3 \\ 0 & 0 & \mathbf{2} & 0 & 0 \\ 0 & 0 & 0 & \mathbf{3} & 3 \end{bmatrix} \begin{array}{l} E_1 \leftarrow E_1/e_{11} \\ E_2 \leftarrow E_2/e_{22} \\ E_3 \leftarrow E_3/e_{33} \\ E_4 \leftarrow E_4/e_{44} \end{array} \tag{3.39}$$

$$\sim \begin{bmatrix} 1 & 0 & 0 & 0 & -2 \\ 0 & 1 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \tag{3.40}$$

Esta última matriz é chamada de matriz escalonada reduzida (por linhas) e a solução do sistema encontra-se em sua última coluna, i.e. $\boldsymbol{x} = (-2, -1, 0, 1)$.

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

```
A = [-2 -3 2 3; ...
     -4 -6 6 6; ...
     -2 0 4 6; ...
     4 3 -8 -6]
b = [10 20 10 -17]'
E = [A b]
n=size(E,1)

%para baixo
for i=1:n-1
  if (abs(E(i,i)) < 1e-15)
    for j=i+1:n
      if (abs(E(j,i)) > 1e-15)
        break
      end
```

```
    endfor
    aux=E(i,:);
    E(i,:)=E(j,:);
    E(j,:)=aux;
  end
  E(i+1:n,:) -= E(i+1:n,i)/E(i,i)*E(i,:);
endfor
E

%para cima
for i=n:-1:2
  E(i,:) = E(i,:)/E(i,i);
  E(1:i-1,:) -= E(1:i-1,i)*E(i,:);
endfor
E(1,:) = E(1,:)/E(1,1)
```

Exercícios

Exercício 3.1.1. Use o método de eliminação gaussiana para obter a matriz escalonada reduzida do seguinte sistema

$$-3x_1 + 2x_2 - 5x_4 + x_5 = -23 \tag{3.41}$$
$$-x_2 - 3x_3 = 9 \tag{3.42}$$
$$-2x_1 - x_2 + x_3 = -1 \tag{3.43}$$
$$2x_2 - 4x_3 + 3x_4 = 8 \tag{3.44}$$
$$x_1 - 3x_3 - x_5 = 11 \tag{3.45}$$

Exercício 3.1.2. Use o método de eliminação gaussiana para obter a matriz escalonada reduzida do seguinte sistema

$$-10^{-12}x_1 + 20x_2 - 3x_3 = -1 \tag{3.46}$$
$$2{,}001x_1 + 10^{-5}x_2 - x_3 = -2 \tag{3.47}$$
$$4x_1 - 2x_2 + x_3 = 0{,}1 \tag{3.48}$$

3.2 Norma e número de condicionamento

Nesta seção, fazemos uma rápida discussão sobre normas de vetores e matrizes, bem como do número de condicionamento de uma matriz.

3.2.1 Norma L^2

A norma L^2 de um dado vetor $v = (v_1, v_2, \ldots, v_n) \in \mathbb{R}^\kappa$ é definida por

$$\|v\| := \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}. \tag{3.49}$$

Proposição 3.2.1. *Dados os vetores u,$v \in \mathbb{R}^\kappa$ e um escalar $\lambda \in \mathbb{R}$, temos:*

a) $\|v\| \geq 0$.

b) $\|v\| = 0 \Leftrightarrow v = 0$.

c) $\|\lambda v\| = |\lambda| \cdot \|v\|$.

d) $\|u + v\| \leq \|u\| + \|v\|$ *(desigualdade triangular).*

e) $u \cdot v \leq \|u\| \cdot \|v\|$ *(desigualdade de Cauchy-Schwarz).*

Exemplo 3.2.1. A norma L^2 do vetor $v = (1, -2, 3, -4)$ é

$$\|v\| = \sqrt{v_1^2 + v_2^2 + v_3^2 + v_4^2} \tag{3.50}$$

$$= \sqrt{1^2 + (-2)^2 + 3^2 + (-4)^2} \tag{3.51}$$

$$= 5{,}4772. \tag{3.52}$$

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

```
v = [1 -2 3 -4]'
normal2 = sqrt(v(1)^2+v(2)^2+v(3)^2+v(4)^2)
norm(v)
```

A norma L^2 induzida de uma dada matriz real $A = [a_{ij}]_{i,j=1}^n$ é definida por

$$\|A\| := \sup_{x \in \mathbb{R}^n, \|x\|=1} \|Ax\|. \tag{3.53}$$

Pode-se mostrar que

$$\|A\| = \sqrt{\lambda_{max}(A^T A)}, \tag{3.54}$$

onde $\lambda_{max}(A^T A) := \max\{|\lambda|;\ \lambda \text{ é autovalor de } A^T A\}$.

Proposição 3.2.2. *Dadas as matrizes reais A, B $n \times n$, um vetor $v \in \mathbb{R}^2$ e um escalar λ, temos*

a) $\|A\| \geq 0$.

b) $\|A\| = 0 \Leftrightarrow A = 0$.

c) $\|\lambda A\| = |\lambda| \cdot \|A\|$.

d) $\|A + B\| \leq \|A\| + \|B\|$ *(desigualdade triangular).*

e) $\|AB\| \leq \|A\|\|B\|$.

f) $\|Av\| \leq \|A\|\|v\|$.

Exemplo 3.2.2. A matriz

$$A = \begin{bmatrix} 1 & -1 & 2 \\ -2 & \pi & 4 \\ 7 & -5 & \sqrt{2} \end{bmatrix} \tag{3.55}$$

tem norma L^2

$$\|A\| = 9{,}3909. \tag{3.56}$$

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

```
A = [1 -1 2; ...
     -2 pi 4; ...
     7 -5 sqrt(2)]

autoval = eig(A'*A)
normal2 = sqrt(max(abs(autoval)))
norm(A)
```

3.2.2 Número de condicionamento

O número de condicionamento de uma matriz é uma medida referente a propagação de erros de ocorre da sua aplicação. Mais especificamente, assumamos que seja dada uma matriz invertível $A = [a_{ij}]_{i,j=1}^{n,n}$, um vetor $x \in \mathbb{R}^n$ e uma perturbação $\delta_x \in \mathbb{R}^n$. Além disso, sejam

$$y = Ax \tag{3.57}$$
$$y + \delta_y = A(x + \delta_x). \tag{3.58}$$

Ou seja, δ_y é a perturbação em y propagada da aplicação de A em x com perturbação δ_x.

Agora, podemos estimar a razão entre os erros relativos $e_{rel}(y) := \|\delta_y\|/\|y\|$ e $e_{rel}(x) := \|\delta_x\|/\|x\|$ da seguinte forma

$$\frac{\frac{\|y\|}{\|\delta_y\|}}{\frac{\|x\|}{\|\delta_x\|}} = \frac{\|y\|}{\|\delta_y\|}\frac{\|\delta_x\|}{\|x\|} \tag{3.59}$$
$$= \frac{\|Ax\|}{\|\delta_y\|}\frac{\|A^{-1}\delta_y\|}{\|x\|} \tag{3.60}$$
$$\leq \frac{\|A\|\|x\|\|A^{-1}\|\|\delta_y\|}{\|\delta_y\|\|x\|} \tag{3.61}$$
$$\leq \|A\|\|A^{-1}\|. \tag{3.62}$$

Logo, temos a seguinte estimativa de propagação de erro

$$e_{rel}(y) \leq \|A\|\|A^{-1}\|e_{rel}(x). \tag{3.63}$$

Isto nos motiva a definir o **número de condicionamento** da matriz A por

$$\kappa(A) := \|A\|\|A^{-1}\|. \tag{3.64}$$

Observação 3.2.1. A matriz identidade tem o menor número de condicionamento, o qual é

$$\kappa(I) = 1. \tag{3.65}$$

Exemplo 3.2.3. Um exemplo de uma matriz bem condicionada é

$$A = \begin{bmatrix} 1 & -1 & 2 \\ -2 & \pi & 4 \\ 7 & -5 & \sqrt{2} \end{bmatrix}, \tag{3.66}$$

cujo número de condicionamento é $\kappa(A) = 13{,}997$.

Já, a matriz

$$B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -2 \\ 10^5 & 10^{-4} & 10^5 \end{bmatrix}, \tag{3.67}$$

tem número de condicionamento

$$\kappa(B) = 1{,}5811 \times 10^{14}, \tag{3.68}$$

o que indica que B é uma matriz mal condicionada.

Exercícios

Exercício 3.2.1. Considere o seguinte sistema linear

$$10^{-12}x_1 + 20x_2 + 3x_3 = -1, \tag{3.69}$$
$$2{,}001x_1 + 10^{-5}x_2 + -x_3 = -2, \tag{3.70}$$
$$x_1 - 2x_2 - 0{,}1x_3 = 0{,}1. \tag{3.71}$$

a) Compute a norma L^2 do vetor dos termos constantes deste sistema.

b) Compute a norma L^2 da matriz dos coeficientes deste sistema.

c) Compute o número de condicionamento da matriz dos coeficientes deste sistema.

3.3 Método de eliminação gaussiana com pivotamento parcial com escala

O método de eliminação gaussiana é suscetível a propagação dos erros de arredondamento, em particular, quando os pivôs são números próximos de

zero. Isto pode ser mitigado com o chamado pivotamento parcial com escala. Nesta variação do método de eliminação gaussiana, o pivô é escolhido como sendo o candidato que é o maior em relação aos elementos em sua linha.

Dado um sistema $Ax = b$ com n-equações e n-incógnitas, o método pode ser descrito pelo seguinte pseudo-código:

1. $E \leftarrow [A\ b]$.
2. Para $i = 1, 2, \ldots, n$, faça $s_i \leftarrow \max_{1 \leq j \leq n} |e_{i,j}|$.
3. Para $i = 1, 2, \ldots, n-1$:
   3.1 Compute j tal que
   $$\frac{e_{j,i}}{s_j} \geq \frac{e_{k,i}}{s_k}, \quad \forall k = i, i+1, \ldots, n. \tag{3.72}$$
   3.2 Permute as linhas i e j, i.e. $E_i \leftrightarrow E_j$.
   3.3 Para $j = i+1, i+2, \ldots, n$:
      3.3.1 $E_j \leftarrow E_j - \frac{e_{ji}}{e_{ii}} E_i$.
4. Para $i = n, n-1, \ldots, 2$:
   4.1 Para $j = i-1, i-2, \ldots, 1$:
      4.1.1 $E_j \leftarrow E_j - \frac{e_{ji}}{e_{ii}} E_i$.

Exemplo 3.3.1. Vamos empregar o método de eliminação gaussiana com pivotamento parcial com escala para resolvermos o seguinte sistema linear

$$10^{-12}x_1 + 20x_2 + 3x_3 = -1, \tag{3.73}$$
$$2{,}001x_1 + 10^{-5}x_2 - x_3 = -2, \tag{3.74}$$
$$x_1 - 2x_2 - 0{,}1x_3 = 0{,}1. \tag{3.75}$$

Para tanto, tomamos a matriz estendida

$$E = \begin{bmatrix} 10^{-12} & 20 & 3 & -1 \\ 2{,}001 & 10^{-5} & -1 & -2 \\ 1 & -2 & -0{,}1 & 0{,}1 \end{bmatrix} \tag{3.76}$$

e computamos os valores máximos em módulo dos elementos de cada linha da matriz A, i.e.

$$s = (20,\ 2{,}001,\ 2). \tag{3.77}$$

Agora, observamos que $e_{2,1}$ é o maior pivô em escala, pois

$$\frac{e_{11}}{s_1} = 5 \times 10^{-14}, \frac{e_{21}}{s_2} = 1, \frac{e_{31}}{s_3} = 0{,}5. \tag{3.78}$$

Então, fazemos a permutação entre as linhas 1 e 2, de forma a obtermos

$$E = \begin{bmatrix} 2{,}001 & 10^{-5} & -1 & -2 \\ 10^{-12} & 20 & 3 & -1 \\ 1 & -2 & -0{,}1 & 0{,}1 \end{bmatrix} \tag{3.79}$$

Em seguida, eliminamos abaixo do pivô, e temos

$$E = \begin{bmatrix} 2{,}001 & 10^{-5} & -1 & -2 \\ 0 & 20 & 3 & -1 \\ 0 & -2 & 3{,}9975 \times 10^{-1} & 1{,}0995 \end{bmatrix} \tag{3.80}$$

Daí, o novo pivô é escolhido como e_{22}, pois ambos candidatos tem o mesmo valor em escala

$$\frac{e_{2,2}}{s_2} = 1, \frac{e_{3,2}}{s_3} = 1. \tag{3.81}$$

Logo, eliminamos abaixo do pivô para obtermos

$$E = \begin{bmatrix} 2{,}001 & 10^{-5} & -1 & -2 \\ 0 & 20 & 3 & -1 \\ 0 & 0 & 6{,}9975 \times 10^{-1} & 9{,}9950 \end{bmatrix} \tag{3.82}$$

Procedendo a eliminação para cima, obtemos a matriz escalonada reduzida

$$E = \begin{bmatrix} 1 & 0 & 0 & -2.8567\text{e}{-1} \\ 0 & 1 & 0 & -2.6425\text{e}{-1} \\ 0 & 0 & 1 & 1{,}4284\text{e}{+0} \end{bmatrix} \tag{3.83}$$

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

```
A = [1e-12 20 3;...
     2.001 1e-5 -1;
     1 -2 -0.1]
b=[-1 -2 0.1]'
E = [A b]

n=size(E,1)

%para baixo
[s,j] = max(abs(E(:,1:n)'));
for i=1:n-1
  [aux,j] = max(abs(E(i:n,i))./s(i:n)');
  j+=i-1;

  aux=E(i,:);
  E(i,:)=E(j,:);
  E(j,:)=aux;

  E(i+1:n,:) -= E(i+1:n,i)/E(i,i)*E(i,:)
endfor

%para cima
for i=n:-1:2
  E(i,:) = E(i,:)/E(i,i);
  E(1:i-1,:) -= E(1:i-1,i)*E(i,:);
endfor
E(1,:) = E(1,:)/E(1,1)
```

Exercícios

Exercício 3.3.1. Use o método de eliminação gaussiana com pivotamento parcial com escala para obter a matriz escalonada reduzida do seguinte sis-

tema

$$-2 \times 10^{-12} x_1 + 10 x_2 - 3 \times 10^{-4} x_3 = 2 \tag{3.84}$$
$$10^5 x_1 + 10^{-13} x_2 - x_3 = -2 \tag{3.85}$$
$$x_1 - 2x_2 + 3 \times 10^{-7} x_3 = 4 \tag{3.86}$$

3.4 Fatoração LU

A fatoração LU é uma forma eficiente de se resolver sistemas lineares. Dado um sistema $Ax = b$, a ideia é de fatorar a matriz A como o produto de uma matriz triangular inferior[1] L com uma matriz triangular superior[2] U, i.e.

$$A = LU. \tag{3.87}$$

Com isso, o sistema $Ax = b$ pode ser reescrito na forma

$$(LU)x = b \Leftrightarrow L(Ux) = b. \tag{3.88}$$

Denotando, $Ux = y$, podemos resolver o seguinte sistema triangular

$$Ly = b. \tag{3.89}$$

Tendo resolvido este sistema, a solução do sistema $Ax = b$ pode, então, ser computada como a solução do seguinte sistema triangular

$$Ux = y. \tag{3.90}$$

Ou seja, a decomposição LU nos permite resolver uma sistema pela resolução de dois sistemas triangulares.

O procedimento de decomposição LU é equivalente ao método de eliminação gaussiana. Consideremos uma matriz $A = [a_{ij}]_{i,j=1}^{n,n}$, com $a_{1,1} \neq 0$. Denotando esta matriz por $U^{(1)} = [u_{i,j}^{(1)}]_{i,j=1}^{n,n} = A$ e tomando $L^{(1)} = I_{n \times n}$,

[1] Do inglês, *low triangular matrix*.
[2] Do inglês, *upper triangular matrix*.

observamos que a eliminação abaixo do pivô $u_{1,1}^{(1)}$, pode ser computada com as seguintes operações equivalentes por linha

$$U^{(1)} = \begin{bmatrix} u_{1,1}^{(1)} & u_{1,2}^{(1)} & u_{1,3}^{(1)} & \dots & u_{1,n}^{(1)} \\ u_{2,1}^{(1)} & u_{2,2}^{(1)} & u_{2,3}^{(1)} & \dots & u_{2,n}^{(1)} \\ u_{3,1}^{(1)} & u_{3,2}^{(1)} & u_{3,3}^{(1)} & \dots & u_{3,n}^{(1)} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ u_{n,1}^{(1)} & u_{n,2}^{(1)} & a_{n,3}^{(1)} & \dots & u_{n,n}^{(1)} \end{bmatrix} \begin{array}{l} U_1^{(2)} \leftarrow U_1^{(1)} \\ U_2^{(2)} \leftarrow U_2^{(1)} - m_{2,1}U_1^{(1)} \\ U_3^{(2)} \leftarrow U_3^{(1)} - m_{3,1}U_1^{(1)} \\ \vdots \\ U_n^{(2)} \leftarrow U_n^{(1)} - m_{n,1}U_1^{(1)} \end{array} \tag{3.91}$$

onde, $m_{i,1} = u_{i,1}^{(1)}/u_{1,1}^{(1)}$, $i = 2, 3, \dots, n$.

Destas computações, obtemos uma nova matriz da forma

$$U^{(2)} = \begin{bmatrix} u_{1,1}^{(2)} & u_{1,2}^{(2)} & u_{1,3}^{(2)} & \dots & u_{1,n}^{(2)} \\ 0 & u_{2,2}^{(2)} & u_{2,3}^{(2)} & \dots & u_{2,n}^{(2)} \\ 0 & u_{3,2}^{(2)} & u_{3,3}^{(2)} & \dots & u_{3,n}^{(2)} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & u_{n,2}^{(2)} & u_{n,3}^{(2)} & \dots & u_{n,n}^{(2)} \end{bmatrix} \tag{3.92}$$

Observemos, também, que denotando

$$L^{(2)} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ m_{2,1} & 1 & 0 & \dots & 0 \\ m_{3,1} & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ m_{n,1} & 0 & 0 & \dots & 1 \end{bmatrix} \tag{3.93}$$

temos

$$A = L^{(2)}U^{(2)}. \tag{3.94}$$

No caso de $u_{2,2}^{(2)} \neq 0$, podemos continuar com o procedimento de eliminação gaussiana com as seguintes operações equivalentes por linha

$$U^{(2)} = \begin{bmatrix} u_{1,1}^{(2)} & u_{1,2}^{(2)} & u_{1,3}^{(2)} & \dots & u_{1,n}^{(2)} \\ 0 & u_{2,2}^{(2)} & u_{2,3}^{(2)} & \dots & u_{2,n}^{(2)} \\ 0 & u_{3,2}^{(2)} & u_{3,3}^{(2)} & \dots & u_{3,n}^{(2)} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & u_{n,2}^{(2)} & u_{n,3}^{(2)} & \dots & u_{n,n}^{(2)} \end{bmatrix} \begin{array}{l} U_1^{(3)} \leftarrow U_1^{(2)} \\ U_2^{(3)} \leftarrow U_2^{(2)} \\ U_3^{(3)} \leftarrow U_3^{(2)} - m_{3,2}U_2^{(2)} \\ \vdots \\ U_n^{(3)} \leftarrow U_n^{(2)} - m_{n,2}U_2^{(2)} \end{array} \tag{3.95}$$

onde, $m_{i,2} = u_{i,2}^{(2)}/u_{2,2}^{(2)}$, $i = 3, 4, \ldots, n$. Isto nos fornece o que nos fornece

$$U^{(3)} = \begin{bmatrix} u_{1,1}^{(3)} & u_{1,2}^{(3)} & u_{1,3}^{(3)} & \ldots & u_{1,n}^{(3)} \\ 0 & u_{2,2}^{(3)} & u_{2,3}^{(3)} & \ldots & u_{2,n}^{(3)} \\ 0 & 0 & u_{3,3}^{(3)} & \ldots & u_{3,n}^{(3)} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & u_{n,3}^{(3)} & \ldots & u_{n,n}^{(3)} \end{bmatrix}. \tag{3.96}$$

Além disso, denotando

$$L^{(3)} = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ m_{2,1} & 1 & 0 & \ldots & 0 \\ m_{3,1} & m_{3,2} & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ m_{n,1} & m_{n,2} & 0 & \ldots & 1 \end{bmatrix} \tag{3.97}$$

temos

$$A = L^{(3)}U^{(3)}. \tag{3.98}$$

Continuando com este procedimento, ao final de $n-1$ passos teremos obtido a decomposição

$$A = LU, \tag{3.99}$$

onde L é a matriz triangular inferior

$$L = L^{(n)} = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 \\ m_{2,1} & 1 & 0 & \ldots & 0 \\ m_{3,1} & m_{3,2} & 1 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ m_{n,1} & m_{n,2} & m_{n,3} & \ldots & 1 \end{bmatrix} \tag{3.100}$$

e U é a matriz triangular superior

$$U = U^{(n)} = \begin{bmatrix} u_{1,1}^{(n)} & u_{1,2}^{(n)} & u_{1,3}^{(n)} & \ldots & u_{1,n}^{(n)} \\ 0 & u_{2,2}^{(n)} & u_{2,3}^{(n)} & \ldots & u_{2,n}^{(n)} \\ 0 & 0 & u_{3,3}^{(n)} & \ldots & u_{3,n}^{(n)} \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ 0 & 0 & 0 & \ldots & u_{n,n}^{(n)} \end{bmatrix}. \tag{3.101}$$

Exemplo 3.4.1. Consideremos a seguinte matriz

$$A = \begin{bmatrix} -1 & 2 & -2 \\ 3 & -4 & 1 \\ 1 & -5 & 3 \end{bmatrix}. \tag{3.102}$$

Então, para obtermos sua decomposição LU começamos com

$$L^{(1)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \; U^{(1)} = \begin{bmatrix} -1 & 2 & -2 \\ 3 & -4 & 1 \\ 1 & -5 & 3 \end{bmatrix} \tag{3.103}$$

Então, observando que a eliminação abaixo do pivô $u_{1,1} = -1$ pode ser feita com as seguintes operações equivalentes por linha $U_2^{(2)} \leftarrow U_2^{(1)} - (-3)U_1^{(1)}$ e $U_3^{(2)} \leftarrow U_3^{(1)} - (-1)U_3^{(1)}$, obtemos

$$L^{(2)} = \begin{bmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}, \; U^{(1)} = \begin{bmatrix} -1 & 2 & -2 \\ 0 & 2 & -5 \\ 0 & -3 & 1 \end{bmatrix} \tag{3.104}$$

Agora, para eliminarmos abaixo do pivô $u_{2,2} = 2$, usamos a operação $U_3^{(3)} \leftarrow U_3^{(2)} - (-3/2)U_3^{(2)}$, donde temos

$$L^{(2)} = \begin{bmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ -1 & -1{,}5 & 1 \end{bmatrix}, \; U^{(1)} = \begin{bmatrix} -1 & 2 & -2 \\ 0 & 2 & -5 \\ 0 & 0 & -6{,}5 \end{bmatrix} \tag{3.105}$$

o que completa a decomposição tomando $L = L^{(3)}$ e $U = U^{(3)}$.

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

```
A = [-1 2 -2;...
     3 -4 1;...
     1 -5 3]

L = eye(size(A))
U = A

L(2:3,1)=U(2:3,1)/U(1,1)
```

```
U(2:3,:)-=U(2:3,1)/U(1,1)*U(1,:)

L(3,2)=U(3,2)/U(2,2)
U(3,:)-=U(3,2)/U(2,2)*U(2,:)
```

Exemplo 3.4.2. Vamos resolver o seguinte sistema linear

$$-x_1 + 2x_2 - 2x_3 = 6 \tag{3.106}$$
$$3x_1 - 4x_2 + x_3 = -11 \tag{3.107}$$
$$x_1 - 5x2 + 3x_3 = -10. \tag{3.108}$$

No exemplo anterior (Exemplo 3.4.2), vimos que a matriz de coeficientes A deste sistema admite a seguinte decomposição LU

$$\underbrace{\begin{bmatrix} -1 & 2 & -2 \\ 3 & -4 & 1 \\ 1 & -5 & 3 \end{bmatrix}}_{A} = \underbrace{\begin{bmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ -1 & -1{,}5 & 1 \end{bmatrix}}_{L} \underbrace{\begin{bmatrix} -1 & 2 & -2 \\ 0 & 2 & -5 \\ 0 & 0 & -6{,}5 \end{bmatrix}}_{U} \tag{3.109}$$

Daí, iniciamos resolvendo o seguinte sistema triangular inferior $Ly = b$, i.e.

$$y_1 = -10 \Rightarrow y_1 = 6 \tag{3.110}$$
$$-3y_1 + y_2 = -11 \Rightarrow y_2 = 7 \tag{3.111}$$
$$-y_1 - 1{,}5y_2 + y_3 = -10 \Rightarrow y_3 = 6{,}5. \tag{3.112}$$

Por fim, computamos a solução x resolvendo o sistema triangular superior $Ux = y$, i.e.

$$-6{,}5x_3 = 6{,}5 \Rightarrow x_3 = -1, \tag{3.113}$$
$$2x_2 - 5x_3 = 7 \Rightarrow x_2 = 1 \tag{3.114}$$
$$-x_1 + 2x_2 - 2x_3 = 6 \Rightarrow x_1 = -2. \tag{3.115}$$

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

```
A = [-1 2 -2;...
     3 -4 1;...
     1 -5 3]
b = [6 -11 -10]'
```

```
#decomposicao LU
L = [1 0 0; ...\\
     -3 1 0; ...
     -1 -1.5 1]
U = [-1 2 -2;...
     0 2 -5;...
     0 0 -6.5]

#Ly = b
n=length(b);
y=zeros(n,1);
for i=1:n
  y(i) = (b(i)-L(i,1:i-1)*y(1:i-1));
endfor
y

#Ux = y
x=zeros(n,1);
for i=n:-1:1
  x(i) = (y(i) - U(i,i+1:n)*x(i+1:n))/U(i,i);
endfor
x
```

3.4.1 Fatoração LU com pivotamento parcial

O algoritmo discutido acima não prevê a necessidade de permutações de linhas no processo de eliminação gaussiana. Isto pode ser corrigido com a utilização de matrizes de permutação.

Assim como na eliminação gaussiana com pivotamento parcial, na fatoração LU com pivotamento parcial o pivô fazemos permutações de linha na matriz de forma que o pivô seja sempre aquele de maior valor em módulo. Por exemplo, suponha que o elemento $a_{3,1}$ seja o maior valor em módulo na

primeira coluna da matriz $A = U^{(1)}$ com

$$U^{(1)} = \begin{bmatrix} u_{1,1}^{(1)} & u_{1,2}^{(1)} & u_{1,3}^{(1)} & \dots & u_{1,n}^{(1)} \\ u_{2,1}^{(1)} & u_{2,2}^{(1)} & u_{2,3}^{(1)} & \dots & u_{2,n}^{(1)} \\ \boldsymbol{u_{3,1}^{(1)}} & u_{3,2}^{(1)} & u_{3,3}^{(1)} & \dots & u_{3,n}^{(1)} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ u_{n,1}^{(1)} & u_{n,2}^{(1)} & a_{n,3}^{(1)} & \dots & u_{n,n}^{(1)} \end{bmatrix}. \tag{3.116}$$

Neste caso, o procedimento de eliminação na primeira coluna deve usar $u_{3,1}^{(1)}$ como pivô, o que requer a permutação entre as linhas 1 e 3 ($U_1^{(1)} \leftrightarrow U_3^{(1)}$). Isto pode ser feito utilizando-se da seguinte matriz de permutação

$$P = \begin{bmatrix} 0 & 0 & 1 & \dots & 0 \\ 0 & 1 & 0 & \dots & 0 \\ 1 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \dots & 1 \end{bmatrix}. \tag{3.117}$$

Com essa, iniciamos o procedimento de decomposição LU com $PA = L^{(1)}U^{(1)}$, onde $L^{(1)} = I_{n\times n}$ e $U^{(1)} = PA$. Caso sejam necessárias outras mudanças de linhas no decorrer do procedimento de decomposição, a matriz de permutação P deve ser atualizada apropriadamente.

Exemplo 3.4.3. Vamos fazer a decomposição LU com pivotamento parcial da seguinte matriz

$$A = \begin{bmatrix} -1 & 2 & -2 \\ 3 & -4 & 1 \\ 1 & -5 & 3 \end{bmatrix} \tag{3.118}$$

Começamos, tomando

$$P^{(1)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad L^{(1)} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad U^{(1)} = \begin{bmatrix} -1 & 2 & -2 \\ 3 & -4 & 1 \\ 1 & -5 & 3 \end{bmatrix} \tag{3.119}$$

O candidato a pivô é o elemento $u_{2,1}$. Então, fazemos as permutações de linhas $P_1 \leftrightarrow P_2$ e $U_1 \leftrightarrow U_2$ e, na sequência, as operações elementares por

linhas $U_{2:3} \leftarrow U_{2:3} - m_{2:3,1}U_1$, donde obtemos

$$P^{(2)} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.120}$$

$$L^{(2)} = \begin{bmatrix} 1 & 0 & 0 \\ -0{,}\overline{3} & 1 & 0 \\ 0{,}\overline{3} & 0 & 1 \end{bmatrix}, U^{(2)} = \begin{bmatrix} 3 & -4 & 1 \\ 0 & 0{,}\overline{6} & -1{,}\overline{6} \\ 0 & -3{,}\overline{6} & 2{,}\overline{6} \end{bmatrix} \tag{3.121}$$

Agora, o candidato a pivô é o elemento $u_{3,2}$. Assim, fazemos as permutações de linhas $P_2 \leftrightarrow P_3$, $U_2 \leftrightarrow U_3$ (análogo para os elementos da coluna 1 de L) e, então, a operação elementar por linha $U_3 \leftarrow U_3 - m_{3,2}U_2$. Com isso, obtemos

$$P^{(2)} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}, \tag{3.122}$$

$$L^{(2)} = \begin{bmatrix} 1 & 0 & 0 \\ 0{,}\overline{3} & 1 & 0 \\ -0{,}\overline{3} & -0{,}\overline{18} & 1 \end{bmatrix}, U^{(2)} = \begin{bmatrix} 3 & -4 & 1 \\ 0 & -3{,}\overline{6} & 2{,}\overline{6} \\ 0 & 0 & -1{,}\overline{18} \end{bmatrix} \tag{3.123}$$

Com isso, temos obtido a decomposição LU de A na forma

$$PA = LU, \tag{3.124}$$

com $P = P^{(3)}$, $L = L^{(3)}$ e $U = U^{(3)}$.

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

```
A = [-1 2 -2;...
     3 -4 1;...
     1 -5 3]

disp("passo 1")
P = eye(size(A))
U = A
L = eye(size(A))

disp("passo 2")
aux = P(1,:);
```

```
P(1,:)=P(2,:);
P(2,:)=aux

aux = U(1,:);
U(1,:)=U(2,:);
U(2,:)=aux;

L(2:3,1)=U(2:3,1)/U(1,1)

U(2:3,:)-=U(2:3,1)/U(1,1)*U(1,:)

disp("passo 3")
aux = P(2,:);
P(2,:)=P(3,:);
P(3,:)=aux

aux = L(2,1);
L(2,1) = L(3,1);
L(3,1) = aux;

aux = U(2,:);
U(2,:)=U(3,:);
U(3,:)=aux;

L(3,2)=U(3,2)/U(2,2)

U(3,:)-=U(3,2)/U(2,2)*U(2,:)
```

Exercícios

Exercício 3.4.1. Resolva o sistema

$$-x_1 + 2x_2 - 2x_3 = -1 \tag{3.125}$$
$$3x_1 - 4x_2 + x_3 = -4 \tag{3.126}$$
$$-4x_1 - 5x_2 + 3x_3 = 20 \tag{3.127}$$

usando fatoração LU com pivotamento parcial.

Capítulo 4

Métodos iterativos para sistemas lineares

4.1 Métodos de Jacobi e de Gauss-Seidel

Nesta seção, discutiremos os métodos de Jacobi[1] e de Gauss-Seidel[2] para a aproximação da solução de sistemas lineares.

4.1.1 Método de Jacobi

Dado um sistema $A\boldsymbol{x} = \boldsymbol{b}$ com n equações e n incógnitas, consideramos a seguinte decomposição da matriz $A = L + D + U$:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \dots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \dots & a_{2n} \\ a_{31} & a_{32} & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & a_{nn} \end{bmatrix} \tag{4.1}$$

[1] Carl Gustav Jacob Jacobi, 1804 - 1851, matemático alemão. Fonte: Wikipedia.

[2] Johann Carl Friedrich Gauss, 1777 - 1855, matemático alemão. Philipp Ludwig von Seidel, 1821 - 1896, matemático alemão. Fonte: Wikipedia.

$$= \underbrace{\begin{bmatrix} 0 & 0 & 0 & \dots & 0 \\ a_{21} & 0 & 0 & \dots & 0 \\ a_{31} & a_{32} & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ a_{n1} & a_{n2} & a_{n3} & \dots & 0 \end{bmatrix}}_{L} \tag{4.2}$$

$$+ \underbrace{\begin{bmatrix} a_{11} & 0 & 0 & \dots & 0 \\ 0 & a_{22} & 0 & \dots & 0 \\ 0 & 0 & a_{33} & \dots & 0 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \dots & a_{nn} \end{bmatrix}}_{D} \tag{4.3}$$

$$+ \underbrace{\begin{bmatrix} 0 & a_{12} & a_{13} & \dots & a_{1n} \\ 0 & 0 & a_{23} & \dots & a_{2n} \\ 0 & 0 & a_{33} & \dots & a_{3n} \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \dots & a_{nn} \end{bmatrix}}_{U}. \tag{4.4}$$

Isto é, a matriz A decomposta como a soma de sua parte triangular inferior L, de sua diagonal D e de sua parte triangular superior U.

Desta forma, podemos reescrever o sistema $A\boldsymbol{x} = b$ da seguinte forma:

$$A\boldsymbol{x} = \boldsymbol{b} \Leftrightarrow (L + D + U)\boldsymbol{x} = \boldsymbol{b} \tag{4.5}$$

$$\Leftrightarrow D\boldsymbol{x} = -(L + U)\boldsymbol{x} + \boldsymbol{b} \tag{4.6}$$

$$\Leftrightarrow \boldsymbol{x} = -D^{-1}(L + U)\boldsymbol{x} + D^{-1}\boldsymbol{b}. \tag{4.7}$$

Ou seja, resolver o sistema $A\boldsymbol{x} = \boldsymbol{b}$ é equivalente a resolver o problema de ponto fixo

$$\boldsymbol{x} = T_J\boldsymbol{x} + \boldsymbol{c}_J, \tag{4.8}$$

onde $T_J = -D^{-1}(L + U)$ é chamada de **matriz de Jacobi** e $\boldsymbol{c}_J = D^{-1}\boldsymbol{b}$ é chamado de **vetor de Jacobi**.

Exemplo 4.1.1. Consideremos o sistema linear $A\boldsymbol{x} = \boldsymbol{b}$ com

$$A = \begin{bmatrix} -4 & 2 & -1 \\ -2 & 5 & 2 \\ 1 & -1 & -3 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} -11 \\ -7 \\ 0 \end{bmatrix}. \tag{4.9}$$

Este sistema tem solução $\boldsymbol{x} = (2, -1, 1)$. Neste caso, temos a decomposição $A = L + D + U$ com

$$L = \begin{bmatrix} 0 & 0 & 0 \\ -2 & 0 & 0 \\ 1 & -1 & 0 \end{bmatrix}, \quad D = \begin{bmatrix} -4 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & -3 \end{bmatrix} \tag{4.10}$$

e

$$U = \begin{bmatrix} 0 & 2 & -1 \\ 0 & 0 & 2 \\ 0 & 0 & 0 \end{bmatrix}. \tag{4.11}$$

Ainda, observamos que

$$T_J\boldsymbol{x} + \boldsymbol{c}_J = -D^{-1}(L+U)\boldsymbol{x} + D^{-1}\boldsymbol{b} \tag{4.12}$$

$$= \underbrace{\begin{bmatrix} 0 & 1/2 & 1/4 \\ 2/5 & 0 & -2/5 \\ 1/3 & -1/3 & 0 \end{bmatrix}}_{T_J} \underbrace{\begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}}_{\boldsymbol{x}} + \underbrace{\begin{bmatrix} 11/4 \\ -7/5 \\ 0 \end{bmatrix}}_{\boldsymbol{c}_J} \tag{4.13}$$

$$= \underbrace{\begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}}_{\boldsymbol{x}}. \tag{4.14}$$

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

```
A = [-4 2 -1;...
     -2 5 2;...
     1 -1 -3]
b = [-11 -7 0]';

x = [2 -1 1]';

L = tril(A,-1)
D = diag(diag(A))
U = triu(A,1)

TJ = -inv(D)*(L+U)
cJ = inv(D)*b
```

```
TJ*x + cJ
```

Com o exposto acima, o **método de Jacobi** consiste na seguinte iteração de ponto fixo

$$\boldsymbol{x}^{(1)} = \text{aprox. inicial}, \tag{4.15}$$

$$\boldsymbol{x}^{(k+1)} = T_J \boldsymbol{x}^{(k)} + \boldsymbol{c}_J, \tag{4.16}$$

onde $\boldsymbol{x}^{(k)} = (x_1^{(k)}, x_2^{(k)}, \ldots, x_n^{(k)})$ é a k-ésima aproximação (ou iterada) de Jacobi.

A iteração (4.16) pode ser equivalentemente escrita na seguinte forma algébrica

$$x_i^{(k+1)} = \frac{b_i - \sum\limits_{\substack{j=1 \\ j\neq i}}^{n} a_{ij} x^{(k)}}{a_{ii}}, \; i = 1, 2, \ldots, n, \tag{4.17}$$

a qual não requer a computação da matriz T_J e $\boldsymbol{c}_J$.

Exemplo 4.1.2. Consideremos o sistema $A\boldsymbol{x} = \boldsymbol{b}$ com

$$A = \begin{bmatrix} -4 & 2 & -1 \\ -2 & 5 & 2 \\ 1 & -1 & -3 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} -11 \\ -7 \\ 0 \end{bmatrix}. \tag{4.18}$$

Aplicando o método de Jacobi com aproximação inicial $\boldsymbol{x}^{(1)} = (0, 0, 0)$ obtemos os resultados da Tabela 4.1.

No `GNU Octave`, podemos obter os resultados reportados na Tabela 4.1 com o seguinte código:

```
A = [-4 2 -1;...
     -2 5 2;...
     1 -1 -3];
b = [-11 -7 0]';
n=size(A,1);

x = [0 0 0]';
disp([x',norm(A*x-b)])
```

k	$\boldsymbol{x}^{(k)}$	$\|A\boldsymbol{x}^{(k)} - \boldsymbol{b}\|$
1	(0,0, 0,0, 0,0)	1,3e+1
2	(2,8, −1,4, 0,0)	7,4e+0
3	(2,0, −0,3, 1,4)	4,6e+0
4	(2,3, −1,1, 0,8)	2,2e+0
5	(2,0, −0,8, 1,1)	1,4e+0
6	(2,1, −1,1, 0,9)	6,9e−1
7	(2,0, −0,9, 1,0)	4,2e−1
8	(2,0, −1,0, 1,0)	2,2e−1
9	(2,0, −1,0, 1,0)	1,3e−1
10	(2,0, −1,0, 1,0)	6,9e−2

Tabela 4.1: Resultados referentes ao Exemplo 4.1.4.

```
for k=2:10
  x0=x;
  for i=1:n
    x(i) = (b(i) - A(i,[1:i-1,i+1:n])*x0([1:i-1,i+1:n]))/A(i,i);
  endfor
  disp([x',norm(A*x-b)])
endfor
```

4.1.2 Método de Gauss-Seidel

Como acima, começamos considerando um sistema linear $A\boldsymbol{x} = \boldsymbol{b}$ e a decomposição $A = L + D + U$, onde L é a parte triangular inferior de A, D é sua parte diagonal e U sua parte triangular superior. Então, observamos que

$$A\boldsymbol{x} = \boldsymbol{b} \Leftrightarrow (L + D + U)\boldsymbol{x} = \boldsymbol{b} \tag{4.19}$$
$$\Leftrightarrow (L + D)\boldsymbol{x} = -U\boldsymbol{x} + \boldsymbol{b} \tag{4.20}$$
$$\Leftrightarrow \boldsymbol{x} = -(L + D)^{-1}U\boldsymbol{x} + (L + D)^{-1}\boldsymbol{b}. \tag{4.21}$$

Isto nos leva a iteração de Gauss-Seidel

$$\boldsymbol{x}^{(1)} = \text{aprox. inicial}, \tag{4.22}$$
$$\boldsymbol{x}^{(k+1)} = T_G\boldsymbol{x}^{(k)} + \boldsymbol{c}_G, \tag{4.23}$$

onde $T_G = -(L + D)^{-1}U$ é a chamada **matriz de Gauss-Seidel** e $\boldsymbol{c}_G = (L + D)^{-1}\boldsymbol{b}$ é o chamado **vetor de Gauss-Seidel**.

Observamos, também, que a iteração (4.23) pode ser reescrita na seguinte forma algébrica

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_{ij}x^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x^{(k)}}{a_{ii}}, \; i = 1, 2, \ldots, n. \tag{4.24}$$

Exemplo 4.1.3. Consideremos o sistema $A\boldsymbol{x} = \boldsymbol{b}$ com

$$A = \begin{bmatrix} -4 & 2 & -1 \\ -2 & 5 & 2 \\ 1 & -1 & -3 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} -11 \\ -7 \\ 0 \end{bmatrix}. \tag{4.25}$$

Aplicando o método de Gauss-Seidel com aproximação inicial $\boldsymbol{x}^{(1)} = (0, 0, 0)$ obtemos os resultados da Tabela 4.2.

k	$\boldsymbol{x}^{(k)}$	$\|A\boldsymbol{x}^{(k)} - \boldsymbol{b}\|$
1	(0,0, 0,0, 0,0)	1,3e+1
2	(2,8, −0,3, 1,0)	2,6e+0
3	(2,3, −0,9, 1,1)	1,2e+0
4	(2,0, −1,0, 1,0)	2,5e−1
5	(2,0, −1,0, 1,0)	4,0e−2

Tabela 4.2: Resultados referentes ao Exemplo 4.1.3.

No `GNU Octave`, podemos obter os resultados reportados na Tabela 4.2 com o seguinte código:

```
A = [-4 2 -1;...
     -2 5 2;...
     1 -1 -3];
b = [-11 -7 0]';
n=size(A,1);

x = [0 0 0]';
disp([x',norm(A*x-b)])
for k=2:5
  for i=1:n
    x(i) = (b(i) - A(i,[1:i-1,i+1:n])*x([1:i-1,i+1:n]))/A(i,i);
```

```
  endfor
  disp([x',norm(A*x-b)])
endfor
```

4.1.3 Análise de convergência

Observamos que ambos os métodos de Jacobi e de Gauss-Seidel consistem de iterações da forma

$$\boldsymbol{x}^{(k+1)} = T\boldsymbol{x}^{(k)} + \boldsymbol{c},\ k = 1, 2, \ldots, \tag{4.26}$$

com $x^{(1)}$ uma aproximação inicial dada, T e c a matriz e o vetor de iteração, respectivamente. O seguinte teorema nos fornece uma condição suficiente e necessária para a convergência de tais métodos.

Teorema 4.1.1. *Para qualquer $\boldsymbol{x}^{(1)} \in \mathbb{R}^n$, temos que a sequência $\{\boldsymbol{x}^{(k+1)}\}_{k=1}^{\infty}$ dada por*

$$\boldsymbol{x}^{(k+1)} = T\boldsymbol{x}^{(k)} + \boldsymbol{c}, \tag{4.27}$$

converge para a solução única de $\boldsymbol{x} = T\boldsymbol{x} + \boldsymbol{c}$ se, e somente se, $\rho(T) < 1$[3]*.*

Demonstração. Veja [2, Cap. 7, Sec. 7.3]. □

Observação 4.1.1. (Taxa de convergência) Para uma iteração da forma (4.26), vale

$$\|\boldsymbol{x}^{(k)} - \boldsymbol{x}\| \approx \rho(T)^{k-1}\|\boldsymbol{x}^{(1)} - \boldsymbol{x}\|, \tag{4.28}$$

onde $\boldsymbol{x}$ é a solução de $\boldsymbol{x} = T\boldsymbol{x} + \boldsymbol{c}$.

Exemplo 4.1.4. Consideremos o sistema $A\boldsymbol{x} = \boldsymbol{b}$ com

$$A = \begin{bmatrix} -4 & 2 & -1 \\ -2 & 5 & 2 \\ 1 & -1 & -3 \end{bmatrix}, \quad \boldsymbol{b} = \begin{bmatrix} -11 \\ -7 \\ 0 \end{bmatrix}. \tag{4.29}$$

Nos Exemplos 4.1.4 e 4.1.3 vimos que ambos os métodos de Jacobi e de Gauss-Seidel eram convergentes, sendo que este convergiu aproximadamente

[3] $\rho(T)$ é o raio espectral da matriz T, i.e. o máximo dos módulos dos autovalores de T.

duas vezes mais rápido que esse. Isto é confirmado pelos raios espectrais das respectivas matrizes de iteração

$$\rho(T_J) \approx 0{,}56, \quad \rho(T_G) \approx 0{,}26. \tag{4.30}$$

No `GNU Octave`, podemos obter os raios espectrais das matrizes de iteração de Jacobi e Gauss-Seidel com o seguinte código:

```
A = [-4 2 -1;...
     -2 5 2;...
     1 -1 -3];
b = [-11 -7 0]';

L = tril(A,-1);
D = diag(diag(A));
U = triu(A,1);

TJ = -inv(D)*(L+U);
rhoTJ = max(abs(eig(TJ)))

TG = -inv(L+D)*U;
rhoTG = max(abs(eig(TG)))
```

Observação 4.1.2. Matriz estritamente diagonal dominante Pode-se mostrar que se A é uma matriz estritamente diagonal dominante, i.e. se

$$|a_{ii}| > \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{ij}|, \forall i = 1, 2, \ldots, n, \tag{4.31}$$

então ambos os métodos de Jacobi e de Gauss-Seidel são convergentes.

Exercícios

Exercício 4.1.1. Considere o seguinte sistema linear

$$-4x_1 + x_2 + x_3 - x_4 = -1 \tag{4.32}$$
$$5x_2 - x_3 + 2x_4 = 3 \tag{4.33}$$
$$-x_1 + 4x_3 - 2x_4 = -2 \tag{4.34}$$

$$x_1 - x_2 - 5x_4 = 1 \tag{4.35}$$

Compute a quinta iterada $x^{(5)}$ do método de Jacobi aplicado a este sistema com aproximação inicial $x^{(1)} = (1, 1, -1, -1)$. Também, compute $\|Ax^{(5)} - b\|$.

Exercício 4.1.2. Considere o seguinte sistema linear

$$-4x_1 + x_2 + x_3 - x_4 = -1 \tag{4.36}$$
$$5x_2 - x_3 + 2x_4 = 3 \tag{4.37}$$
$$-x_1 + 4x_3 - 2x_4 = -2 \tag{4.38}$$
$$x_1 - x_2 - 5x_4 = 1 \tag{4.39}$$

Compute a quinta iterada $x^{(5)}$ do método de Gauss-Seidel aplicado a este sistema com aproximação inicial $x^{(1)} = (1, 1, -1, -1)$. Também, compute $\|Ax^{(5)} - b\|$.

4.2 Método do gradiente

Começamos observando que se A é uma matriz $n \times n$ positiva definida[4], temos que $\boldsymbol{x} \in \mathbb{R}^n$ é solução de

$$A\boldsymbol{x} = \boldsymbol{b} \tag{4.40}$$

se, e somente se, $\boldsymbol{x}$ é solução do seguinte problema de minimização

$$\min_{\boldsymbol{x} \in \mathbb{R}^n} f(\boldsymbol{x}) := \frac{1}{2}\boldsymbol{x}^T A\boldsymbol{x} - \boldsymbol{x}^T\boldsymbol{b}. \tag{4.41}$$

O método do gradiente é um algoritmo da forma: dada uma aproximação inicial $\boldsymbol{x}^{(1)}$ da solução de (4.41) (ou, equivalentemente, de (4.40)), computamos novas aproximações da forma iterativa

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha^{(k)}\boldsymbol{d}^{(k)}, \quad k = 1, 2, \ldots, \tag{4.42}$$

onde $\alpha^{(k)}$ é o tamanho do passo (um escalar) e $\boldsymbol{d}^{(k)} \in \mathbb{R}^n$ é a direção de busca.

[4] A é simétrica e $x^T Ax > 0$ para todo $x \neq 0$.

Para escolhermos a direção $\boldsymbol{d}^{(k)}$, tomamos a fórmula de Taylor de f em torno da aproximação $\boldsymbol{x}^{(k)}$

$$f(\boldsymbol{x}^{(k+1)}) = f(\boldsymbol{x}^{(k)}) + \alpha^{(k)} \nabla f(\boldsymbol{x}^{(k)}) \cdot \boldsymbol{d}^{(k)} + O\left((\alpha^{(k)})^2\right), \tag{4.43}$$

com $\alpha^{(k)} \to 0$, onde ∇f denota o gradiente de f, i.e.

$$\nabla f(\boldsymbol{x}^{(k)}) = \left(\frac{\partial f}{\partial x_1}(\boldsymbol{x}^{(k)}), \frac{\partial f}{\partial x_2}(\boldsymbol{x}^{(k)}), \ldots, \frac{\partial f}{\partial x_n}(\boldsymbol{x}^{(k)}) \right) \tag{4.44}$$

$$= A\boldsymbol{x}^{(k)} - \boldsymbol{b}. \tag{4.45}$$

De (4.43), segue que se

$$\nabla f(\boldsymbol{x}^{(k)}) \cdot \boldsymbol{d}^{(k)} < 0, \tag{4.46}$$

então $f(\boldsymbol{x}^{(k+1)}) < f(\boldsymbol{x}^{(k)})$ se $\alpha^{(k)}$ é suficientemente pequeno. Em particular, podemos escolher

$$\boldsymbol{d}^{(k)} = -\nabla f(\boldsymbol{x}^{(k)}), \tag{4.47}$$

se $\nabla f(\boldsymbol{x}^{(k)}) \neq 0$.

Do exposto acima, temos a **iteração do método do gradiente**

$$\boldsymbol{x}^{(1)} = \text{aprox. inicial} \tag{4.48}$$

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - \alpha^{(k)} \boldsymbol{r}^{(k)},\ k = 1, 2, \ldots, \tag{4.49}$$

onde $\boldsymbol{r}^{(k)}$ é o resíduo da iterada k dado por

$$\boldsymbol{r}^{(k)} = A\boldsymbol{x}^{(\boldsymbol{k})} - \boldsymbol{b}. \tag{4.50}$$

Exemplo 4.2.1. Consideremos o sistema $Ax = b$ com

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} -3 \\ 2 \\ 2 \\ -3 \end{bmatrix}. \tag{4.51}$$

Na Tabela 4.3 temos os resultados do emprego do método do gradiente com $\boldsymbol{x}^{(1)} = (0, 0, 0, 0)$ e com passo constante $\alpha^{(k)} \equiv 0{,}5$.

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

Tabela 4.3: Resultados referentes ao Exemplo 4.2.1.

k	$\boldsymbol{x}^{(k)}$	$\|A\boldsymbol{x}^{(k)} - \boldsymbol{b}\|$
1	(0,0, 0,0, 0,0, 0,0)	5,1e+0
2	(−1,5, 1,0, 1,0, −1,5)	1,6e+0
3	(−1,0, 0,8, 0,8, −1,0)	5,0e−1
4	(−1,1, 0,9, 0,9, −1,1)	1,8e−1
5	(−1,1, 0,9, 0,9, −1,1)	8,8e−2
6	(−1,1, 0,9, 0,9, −1,1)	6,2e−2
7	(−1,0, 0,9, 0,9, −1,0)	4,9e−2
8	(−1,0, 0,9, 0,9, −1,0)	4,0e−2
9	(−1,0, 0,9, 0,9, −1,0)	3,2e−2
10	(−1,0, 1,0, 1,0, −1,0)	2,6e−2
11	(−1,0, 1,0, 1,0, −1,0)	2,1e−2

```
A = [2 -1 0 0;...
     -1 2 -1 0;...
     0 -1 2 -1;...
     0 0 -1 2]
b = [-3 2 2 -3]'

alpha=5e-1;
x = [0 0 0 0]'
r = A*x - b;
disp([x',norm(r)])

for k=2:11
  x = x - alpha*r;
  r = A*x - b;
  disp([x',norm(r)])
endfor
```

4.2.1 Escolha do passo

Da iteração do método do gradiente, temos que a melhor escolha do passo $\alpha^{(k)}$ é tal que

$$f(\boldsymbol{x}^{(k)} + \alpha^{(k)}\boldsymbol{r}^{(k)}) = \min_{\alpha>0} f(\boldsymbol{x}^{(k)} + \alpha\boldsymbol{r}^{(k)}). \tag{4.52}$$

Desta forma,

$$\frac{\mathrm{d}}{\mathrm{d}\alpha} f(\boldsymbol{x}^{(k)} + \alpha \boldsymbol{r}^{(k)}) = 0 \Rightarrow \nabla f(\boldsymbol{x}^{(k+1)}) \cdot \boldsymbol{r}^{(k)} = 0, \tag{4.53}$$

$$\Rightarrow \left(A(\boldsymbol{x}^{(k)} + \alpha^{(k)} \boldsymbol{r}^{(k)}) - b\right) \cdot \boldsymbol{r}^{(k)} = 0, \tag{4.54}$$

$$\Rightarrow (A\boldsymbol{x}^{(k)} - \boldsymbol{b}) \cdot \boldsymbol{r}^{(k)} + \alpha^{(k)} \boldsymbol{r}^{(k)} \cdot A\boldsymbol{r}^{(k)} = 0, \tag{4.55}$$

donde

$$\alpha^{(k)} = -\frac{\boldsymbol{r}^{(k)} \cdot \boldsymbol{r}^{(k)}}{\boldsymbol{r}^{(k)} \cdot A\boldsymbol{r}^{(k)}}. \tag{4.56}$$

Exemplo 4.2.2. Consideremos o sistema $Ax = b$ com

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} -3 \\ 2 \\ 2 \\ -3 \end{bmatrix}. \tag{4.57}$$

Na Tabela 4.4 temos os resultados do emprego do método do gradiente com $\boldsymbol{x}^{(1)} = (0, 0, 0, 0)$ e com passo

$$\alpha^{(k)} = -\frac{\boldsymbol{r}^{(k)} \cdot \boldsymbol{r}^{(k)}}{\boldsymbol{r}^{(k)} \cdot A\boldsymbol{r}^{(k)}}. \tag{4.58}$$

Tabela 4.4: Resultados referentes ao Exemplo 4.2.2.

k	$\boldsymbol{x}^{(k)}$	$\|A\boldsymbol{x}^{(k)} - \boldsymbol{b}\|$
1	(0,0, 0,0, 0,0, 0,0)	5,1e+0
2	(−1,1, 0,8, 0,8, −1,1)	1,5e−1
3	(−1,0, 1,0, 1,0, −1,0)	3,0e−2
4	(−1,0, 1,0, 1,0, −1,0)	8,8e−4
5	(−1,0, 1,0, 1,0, −1,0)	1,8e−4

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

```
A = [2 -1 0 0;...
     -1 2 -1 0;...
```

```
     0 -1 2 -1;...
     0 0 -1 2]
x = [-1 1 1 -1]';
b = [-3 2 2 -3]'

x = [0 0 0 0]'
r = A*x - b;
disp([x',norm(r)])

for k=2:5
  alpha = r'*r/(r'*A*r);
  x = x - alpha*r;
  r = A*x - b;
  disp([x',norm(r)])
endfor
```

Exercícios

Em construção ...

4.3 Método do gradiente conjugado

O método do gradiente conjugado é uma variação do método do gradiente (veja Seção 4.2). Aqui, a solução de um dado sistema $A\boldsymbol{x} = \boldsymbol{b}$, com A uma matriz positiva definida, é computada de forma iterativa por

$$\boldsymbol{x}^{(1)} = \text{aprox. inicial}, \tag{4.59}$$

$$\boldsymbol{d}^{(1)} = \boldsymbol{r}^{(1)}, \tag{4.60}$$

$$\tag{4.61}$$

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} + \alpha_k \boldsymbol{d}^{(k)}, \tag{4.62}$$

$$\alpha^{(k)} = -\frac{\boldsymbol{r}^{(k)} \cdot \boldsymbol{d}^{(k)}}{\boldsymbol{d}^{(k)} \cdot A\boldsymbol{d}^{(k)}}, \tag{4.63}$$

$$\boldsymbol{d}^{(k+1)} = -\boldsymbol{r}^{(k+1)} + \beta_k \boldsymbol{d}^{(k)}, \tag{4.64}$$

$$\beta^{(k)} = \frac{\boldsymbol{r}^{(k+1)} \cdot A\boldsymbol{d}^{(k)}}{\boldsymbol{d}^{(k)} \cdot A\boldsymbol{d}^{(k)}}, \tag{4.65}$$

para $k = 1, 2, \ldots$, e $\boldsymbol{r}^{(k)} = A\boldsymbol{x}^{(k)} - \boldsymbol{b}$.

Exemplo 4.3.1. Consideremos o sistema $Ax = b$ com

$$A = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} -3 \\ 2 \\ 2 \\ -3 \end{bmatrix}. \tag{4.66}$$

Na Tabela 4.5 temos os resultados do emprego do método do gradiente conjugado com $\boldsymbol{x}^{(1)} = (0, 0, 0, 0)$.

Tabela 4.5: Resultados referentes ao Exemplo 4.3.1.

k	$\boldsymbol{x}^{(k)}$	$\|A\boldsymbol{x}^{(k)} - \boldsymbol{b}\|$
1	(0, 0, 0, 0)	5.1e+0
2	(−1,1, 0,8, 0,8, − 1,1)	1,5e−1
3	(−1,0, 1,0, 1,0, − 1,0)	0,0e+0

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

```
A = [2 -1 0 0;...
     -1 2 -1 0;...
     0 -1 2 -1;...
     0 0 -1 2]
x = [-1 1 1 -1]';
b = [-3 2 2 -3]'

x = [0 0 0 0]'
r = A*x - b;
d = r;
disp([x',norm(r)])

for k=2:3
  alpha = r'*d/(d'*A*d);
```

```
  x = x - alpha*d;
  r = A*x - b;
  disp([x',norm(r)])
  beta=r'*A*d/(d'*A*d);
  d=-r+beta*d;
endfor
```

Exercícios

Em construção ...

Capítulo 5

Sistema de equações não lineares

Neste capítulo, discutiremos sobre métodos para a resolução de sistema de equações não lineares. Vamos tratar o caso de problemas da forma: encontrar $\boldsymbol{x} \in \mathbb{R}^n$ tal que

$$F(\boldsymbol{x}) = \mathbf{0}, \tag{5.1}$$

onde $F : \mathbb{R}^n \to \mathbb{R}^n$ é uma função vetorial dada.

5.1 Método de Newton

Consideremos o problema de encontrar $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ tal que

$$F(\boldsymbol{x}) = \mathbf{0}, \tag{5.2}$$

onde $F : \mathbb{R}^n \to \mathbb{R}^n$ é uma função vetorial dada com $F(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x}))$. Suponhamos, então, que $\boldsymbol{x}^*$ seja a solução exata deste problema e que $\boldsymbol{x}^{(1)}$ seja uma aproximação de $\boldsymbol{x}^*$. Assim sendo, tomemos a seguinte expansão de F em polinômio de Taylor:

$$F(\boldsymbol{x}^*) = F(\boldsymbol{x}^{(1)}) + J_F(\boldsymbol{x}^{(1)})(\boldsymbol{x}^* - \boldsymbol{x}^{(1)}) + R, \tag{5.3}$$

onde J_F é a matriz jacobiana[1] de F

$$J_F := \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix} \tag{5.4}$$

e $\|R\|^2 \to 0$ com $\|\boldsymbol{x}^{(1)} - \boldsymbol{x}^*\| \to 0$.

Daí, como $F(\boldsymbol{x}^*) = \mathbf{0}$, temos

$$J_F(\boldsymbol{x}^{(1)})(\boldsymbol{x}^* - \boldsymbol{x}^{(1)}) \approx -F(\boldsymbol{x}^{(1)}). \tag{5.5}$$

Então, multiplicando a inversa da jacobiana à esquerda, obtemos

$$\boldsymbol{x}^* - \boldsymbol{x}^{(1)} \approx -J_F^{-1}(\boldsymbol{x}^{(1)})F(\boldsymbol{x}^{(1)}) \tag{5.6}$$

e também

$$\boldsymbol{x}^* \approx \boldsymbol{x}^{(1)} - J_F^{-1}(\boldsymbol{x}^{(1)})F(\boldsymbol{x}^{(1)}). \tag{5.7}$$

O exposto acima nos motiva a **iteração de Newton**[2]:

$$\boldsymbol{x}^{(1)} = \text{aprox. inicial}, \tag{5.8}$$
$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - J_F^{-1}(\boldsymbol{x}^{(k)})F(\boldsymbol{x}^{(k)}), \tag{5.9}$$

com $k = 1, 2, \ldots$.

Exemplo 5.1.1. Consideremos o seguinte sistema de equações não lineares

$$x_1 x_2^2 = x_1^2 x_2 - 6, \tag{5.10}$$
$$x_1^2 x_2^3 - 7 = -x_1. \tag{5.11}$$

Para usarmos o método de Newton, reescrevemos o sistema na seguinte forma

$$x_1 x_2^2 - x_1^2 x_2 + 6 = 0, \tag{5.12}$$
$$x_1 + x_1^2 x_2^3 - 7 = 0. \tag{5.13}$$

[1] Carl Gustav Jacob Jacobi, 1804 - 1851, matemático alemão. Fonte: Wikipedia.
[2] Sir Isaac Newton, 1642 - 1726/27, matemático e físico inglês. Fonte: Wikipedia.

Com isso, identificamos a função objetivo

$$F(\boldsymbol{x}) = \begin{bmatrix} f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}) \end{bmatrix} = \begin{bmatrix} x_1 x_2^2 - x_1^2 x_2 + 6 \\ x_1 + x_1^2 x_2^3 - 7 \end{bmatrix} \tag{5.14}$$

e sua matriz jacobiana

$$J_F(\boldsymbol{x}) = \frac{\partial(f_1, f_2)}{\partial(x_1, x_2)} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix} \tag{5.15}$$

$$= \begin{bmatrix} x_2^2 - 2x_1x_2 & 2x_1x_2 - x_1^2 \\ 1 + 2x_1x_2^3 & 3x_1^2x_2^2 \end{bmatrix} \tag{5.16}$$

Definidas F e J_F e tomando $\boldsymbol{x}^{(1)} = (1{,}5,\ 1{,}5)$ como aproximação inicial, computamos as iterações de Newton de forma a obtermos os resultados apresentados na Tabela 5.1.

k	$\boldsymbol{x}^{(k)}$	$\|F(\boldsymbol{x}^{(k)})\|$
1	(−1,50, 1,50)	1,2e+0
2	(−1,07, 1,82)	1,2e+0
3	(−9,95e−1, 2,00)	7,6e−2
4	(−1,00, 2,00)	1,2e−4
5	(−1,00, 2,00)	2,1e−9

Tabela 5.1: Resultados referentes ao Exemplo 5.1.1.

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

```
disp('Newton')

function y = F(x)
  y = zeros(size(x));
  y(1) = x(1)*x(2)^2 - x(1)^2*x(2)+6;
  y(2) = x(1) + x(1)^2*x(2)^3-7;
endfunction

function y = J(x)
  y=zeros(2,2);
  y(1,1) = x(2)^2-2*x(1)*x(2);
```

```
  y(1,2) = 2*x(1)*x(2)-x(1)^2;
  y(2,1) = 1+2*x(1)*x(2)^3;
  y(2,2) = 3*x(1)^2*x(2)^2;
endfunction

x=[-1.5 1.5]';
printf("%d %1.2E %1.2E %1.1E\n",1,x,norm(F(x)))
for k=2:5
  x = x - inv(J(x))*F(x);
  printf("%d %1.2E %1.2E %1.1E\n",k,x,norm(F(x)))
endfor
```

5.1.1 Considerações sobre convergência

Para uma função F suficientemente suave e com uma escolha apropriada da aproximação inicial $\boldsymbol{x}^{(1)}$, temos que as iterações de Newton

$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - J_F^{-1}(\boldsymbol{x}^{(1)})F(\boldsymbol{x}^{(1)}), \tag{5.17}$$

são quadraticamente convergentes, i.e.

$$\|\boldsymbol{x}^{(k+1)} - \boldsymbol{x}^*\| \leq C\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|^2, \tag{5.18}$$

onde $\boldsymbol{x}^*$ é a solução exata, i.e. $F(\boldsymbol{x}^*) = \mathbf{0}$.

Exemplo 5.1.2. Consideremos o seguinte sistema de equações não lineares

$$x_1x_2^2 - x_1^2x_2 + 6 = 0, \tag{5.19}$$
$$x_1 + x_1^2x_2^3 - 7 = 0. \tag{5.20}$$

A Figura 5.1 é um esboço do gráfico da $\|F(\cdot)\|$. Este problema foi confeccionado de forma que $\boldsymbol{x}^* = (-1, 2)$. Então, tomando $\boldsymbol{x}^{(1)} = (1{,}5,\ 1{,}5)$ como aproximação inicial, computamos as iterações de Newton para este problema, donde obtemos os resultados reportados na Tabela 5.2.

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

```
disp('Newton')

function y = F(x)
```

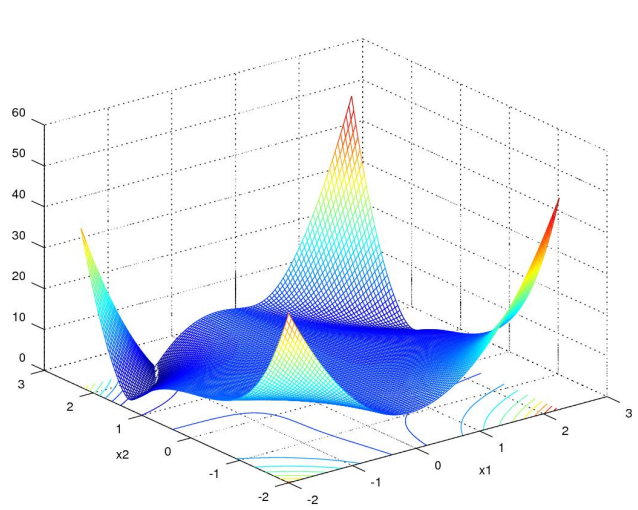

Figura 5.1: Esboço do gráfico de $\|F(\cdot)\|$ referente ao Exemplo 5.1.2.

k	$\boldsymbol{x}^{(k)}$	$\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|$
1	(−1,50, 1,50)	7,1e−01
2	(−1,07, 1,82)	2,0e−01
3	(−9,95e−1, 2,00)	5,1e−03
4	(−1,00, 2,00)	2,6e−05
5	(−1,00, 2,00)	2,0e−10

Tabela 5.2: Resultados referentes ao Exemplo 5.1.2.

```
  y = zeros(size(x));
  y(1) = x(1)*x(2)^2 - x(1)^2*x(2)+6;
  y(2) = x(1) + x(1)^2*x(2)^3-7;
endfunction

function y = J(x)
  y=zeros(2,2);
  y(1,1) = x(2)^2-2*x(1)*x(2);
  y(1,2) = 2*x(1)*x(2)-x(1)^2;
  y(2,1) = 1+2*x(1)*x(2)^3;
  y(2,2) = 3*x(1)^2*x(2)^2;
endfunction
```

```
xx1 = linspace(-2,2.25);
xx2 = xx1;
zz = zeros(length(xx1),length(xx2));
for i = 1:length(xx1)
  for j = 1:length(xx2)
    zz(j,i) = norm(F([xx1(i),xx2(j)]));
  endfor
endfor

meshc(xx1,xx2,zz)
xlabel("x1")
ylabel("x2")

xstar = [-1 2]';
x=[-1.5 1.5]';
printf("%d %1.2E %1.2E %1.1E\n",1,x,norm(x-xstar))
for k=2:5
  x = x - inv(J(x))*F(x);
  printf("%d %1.2E %1.2E %1.1E\n",k,x,norm(x-xstar))
endfor
```

Exercícios

Exercício 5.1.1. Use o método de Newton para obter uma aproximação de uma solução de

$$x_2 \operatorname{sen}(x_3) + x_1 - 2 = 0, \tag{5.21}$$
$$x_1 x_2 - \operatorname{sen}(x_2) + 0{,}2 = 0, \tag{5.22}$$
$$x_3^2 + \cos(x_1 x_2) - 4{,}5 = 0. \tag{5.23}$$

Para tanto, use $\boldsymbol{x}^{(1)} = (1,\ -1,\ -1)$.

5.2 Métodos *quasi*-Newton

5.2.1 Método do acorde

O método do acorde consiste na seguinte iteração

$$\boldsymbol{x}^{(1)} = \text{aprox. inicial}, \tag{5.24}$$
$$\boldsymbol{x}^{(k+1)} = \boldsymbol{x}^{(k)} - J_F^{-1}(\boldsymbol{x}^{(1)})F(\boldsymbol{x}^{(k)}). \tag{5.25}$$

Ou seja, é a iteração de Newton com jacobina constante.

Exemplo 5.2.1. Consideremos o seguinte sistema de equações não lineares

$$x_1x_2^2 - x_1^2x_2 + 6 = 0, \tag{5.26}$$
$$x_1 + x_1^2x_2^3 - 7 = 0. \tag{5.27}$$

Definidas F e J_F e tomando $\boldsymbol{x}^{(1)} = (1{,}5,\ 1{,}5)$ como aproximação inicial, computamos as iterações do método do acorde de forma a obtermos os resultados apresentados na Tabela 5.3.

k	$\boldsymbol{x}^{(k)}$	$\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|$
1	$(-1{,}50,\ 1{,}50)$	-x-
2	$(-1{,}07,\ 1.82)$	$5{,}3\text{e}{-1}$
3	$(-1{,}02,\ 1{,}93)$	$1{,}2\text{e}{-1}$
4	$(-1{,}00,\ 1{,}98)$	$5{,}2\text{e}{-2}$
5	$(-9{,}98\text{e}{-1},\ 2{,}00)$	$1{,}8\text{e}{-2}$
6	$(-9{,}98\text{e}{-1},\ 2{,}00)$	$4{,}7\text{e}{-3}$
7	$(-9{,}99\text{e}{-1},\ 2{,}00)$	$9{,}0\text{e}{-4}$
8	$(-1{,}00,\ 2{,}00)$	$7{,}4\text{e}{-4}$
9	$(-1{,}00,\ 2{,}00)$	$4{,}3\text{e}{-4}$

Tabela 5.3: Resultados referentes ao Exemplo 5.2.1.

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

```
disp('Newton')

function y = F(x)
  y = zeros(size(x));
```

```
  y(1) = x(1)*x(2)^2 - x(1)^2*x(2)+6;
  y(2) = x(1) + x(1)^2*x(2)^3-7;
endfunction

function y = J(x)
  y=zeros(2,2);
  y(1,1) = x(2)^2-2*x(1)*x(2);
  y(1,2) = 2*x(1)*x(2)-x(1)^2;
  y(2,1) = 1+2*x(1)*x(2)^3;
  y(2,2) = 3*x(1)^2*x(2)^2;
endfunction

x=[-1.5 1.5]';
x0=x;
printf("%d %1.2E %1.2E\n",1,x)
A = inv(J(x));
for k=2:9
  x = x - A*F(x);
  printf("%d %1.2E %1.2E %1.1E\n",k,x,norm(x-x0))
  x0=x;
endfor
```

5.2.2 Jacobiana aproximada

A jacobiana $J_F(\boldsymbol{x})$ de uma dada função $F(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_n(\boldsymbol{x}))$ é a matriz cujo elemento da i-ésima linha e j-ésima coluna é

$$\frac{\partial f_i}{\partial x_j} = \lim_{h \to 0} \frac{f_i(\boldsymbol{x} + \boldsymbol{e}_j h) - f_i(\boldsymbol{x})}{h}, \tag{5.28}$$

onde $\boldsymbol{e}_j$ é o j-ésimo vetor da base canônica de $\mathbb{R}^n$, i.e. $\boldsymbol{e}_j = (0, \ldots, 0, 1, 0, \ldots, 0)$ com 1 na j-ésima posição.

Com isso, podemos computar uma jacobiana aproximada tomando

$$\frac{\partial f_i}{\partial x_j} \approx \frac{f_i(\boldsymbol{x} + \boldsymbol{e}_j h) - f_i(\boldsymbol{x})}{h}, \tag{5.29}$$

com h suficientemente pequeno.

Exemplo 5.2.2. Consideremos o seguinte sistema de equações não lineares

$$x_1 x_2^2 - x_1^2 x_2 + 6 = 0, \tag{5.30}$$
$$x_1 + x_1^2 x_2^3 - 7 = 0. \tag{5.31}$$

Definida F, sua jacobina aproximada $\tilde{J}_F$ e tomando $\boldsymbol{x}^{(1)} = (1{,}5,\ 1{,}5)$ como aproximação inicial, computamos as iterações do *quasi*-método de forma a obtermos os resultados apresentados na Tabela 5.4.

k	$\boldsymbol{x}^{(k)}$	$\|\boldsymbol{x}^{(k)} - \boldsymbol{x}^*\|$
1	$(-1{,}50,\ 1{,}50)$	-x-
2	$(-1{,}07,\ 1{,}82)$	$5{,}3\text{e}{-1}$
3	$(-9{,}95\text{e}{-1},\ 2{,}00)$	$2{,}0\text{e}{-1}$
4	$(-1{,}00,\ 2{,}00)$	$5{,}1\text{e}{-3}$
5	$(-1{,}00,\ 2{,}00)$	$2{,}6\text{e}{-5}$

Tabela 5.4: Resultados referentes ao Exemplo 5.2.2.

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

```
disp('Newton')

function y = F(x)
  y = zeros(size(x));
  y(1) = x(1)*x(2)^2 - x(1)^2*x(2)+6;
  y(2) = x(1) + x(1)^2*x(2)^3-7;
endfunction

function y = Ja(x)
  h = 1e-7;
  y=zeros(2,2);
  fx1 = F(x + [h 0]');
  fx0 = F(x);
  y(:,1) = (fx1 - fx0)/h;
  fx1 = F(x + [0 h]');
  fx0 = F(x);
  y(:,2) = (fx1 - fx0)/h;
endfunction
```

```
x=[-1.5 1.5]';
x0=x;
printf("%d %1.2E %1.2E\n",1,x)
for k=2:5
  x = x - inv(Ja(x))*F(x);
  printf("%d %1.2E %1.2E %1.1E\n",k,x,norm(x-x0))
  x0=x;
endfor
```

Exercícios

Em construção ...

Capítulo 6

Interpolação

Neste capítulo, discutiremos sobre a resolução de problemas de interpolação da forma: dados uma família de n funções reais $\mathcal{F} = \{f_1(x), f_2(x), \ldots, f_n(x)\}$ e um conjunto de n pontos $\{(x_i, y_i)\}_{i=1}^n$, com $x_i \neq x_j$ se $i \neq j$, encontrar

$$f(x) = c_1 f_1(x) + c_2 f_2(x) + \cdots + c_n f_n(x), \tag{6.1}$$

tal que

$$y_i = f(x_i), \quad i = 1, 2, \ldots, n. \tag{6.2}$$

6.1 Interpolação polinomial

Dado um conjunto de n pontos $\{(x_i, y_i)\}_{i=1}^n$, o problema de interpolação consiste em encontrar o polinômio de grau $n-1$

$$p(x) = p_1 x^{n-1} + p_2 x^{n-2} + \cdots + p_{n-1} x + p_n \tag{6.3}$$

tal que

$$y_i = p(x_i), \quad i = 1, 2, \ldots, n. \tag{6.4}$$

Das condições (6.3), temos

$$p_1 x_1^{n-1} + p_2 x_1^{n-2} + \cdots + p_{n-1} x_1 + p_n = y_1 \tag{6.5}$$

$$p_1 x_2^{n-1} + p_2 x_2^{n-2} + \cdots + p_{n-1} x_2 + p_n = y_2 \tag{6.6}$$

$$\vdots \tag{6.7}$$

$$p_1 x_n^{n-1} + p_2 x_n^{n-2} + \cdots + p_{n-1} x_n + p_n = y_n. \tag{6.8}$$

Isto é, os coeficientes do polinômio interpolador (6.3) satisfazem o seguinte sistema linear

$$A\boldsymbol{p} = \boldsymbol{y}, \tag{6.9}$$

onde A é a matriz de Vandermonde[1]

$$A = \begin{bmatrix} x_1^{n-1} & x_1^{n-2} & \dots & x_1 & 1 \\ x_2^{n-1} & x_2^{n-2} & \dots & x_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^{n-1} & x_n^{n-2} & \dots & x_n & 1 \end{bmatrix}, \tag{6.10}$$

$\boldsymbol{p} = (p_1, p_2, \dots, p_n)$ é o vetor das incógnitas e $\boldsymbol{y} = (y_1, y_2, \dots, y_n)$ é o vetor dos termos constantes.

Exemplo 6.1.1. Consideremos o problema de encontrar o polinômio interpolador do conjunto de pontos $\{(-1, -1), (0, 1), (1, 1/2)\}$. Como temos 3 pontos, o polinômio tem grau 2 e pode ser escrito na forma

$$p(x) = p_1 x^2 + p_2 x + p_3. \tag{6.11}$$

Seguindo a abordagem acima, temos $\boldsymbol{p} = (p_1, p_2, p_3)$, $\boldsymbol{x} = (-1, 0, 1)$, $\boldsymbol{y} = (-1, 1, 1/2)$ e

$$A = \begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{bmatrix}. \tag{6.12}$$

Então, resolvendo $A\boldsymbol{p} = \boldsymbol{y}$, obtemos o polinômio interpolador

$$p(x) = -1{,}25x^2 + 0{,}75x + 1. \tag{6.13}$$

A Figura 6.1 mostra os esboços do polinômio interpolador $p(x)$ e dos pontos dados.

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

[1] Alexandre-Théophile Vandermonde, 1735 - 1796, matemático francês. Fonte: Wikipedia.

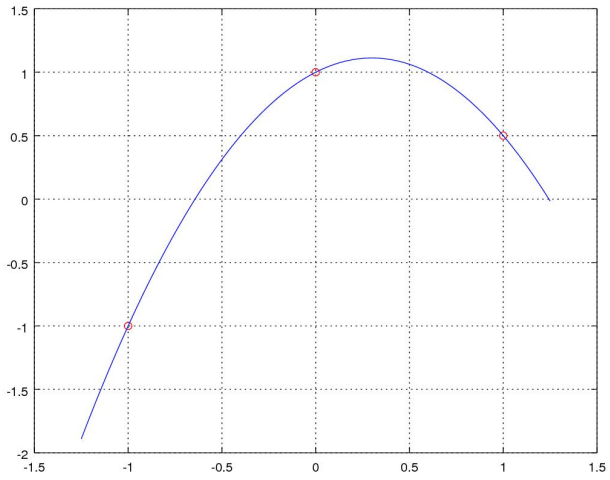

Figura 6.1: Esboços dos pontos e do polinômio interpolador referente ao Exemplo 6.1.1.

```
x = [-1 0 1]';
y = [-1 1 0.5]';
A = [x.^2 x.^1 x.^0]
p = inv(A)*y

xx=linspace(-1.25,1.25);
plot(x,y,'ro',...
     xx,polyval(p,xx),'b-')
```

Exercícios

Exercício 6.1.1. Obtenha o polinômio interpolador do conjunto de pontos $\{(-1, -1), (0, 1), (1, 1/2), (2, 1)\}$.

Em construção ...

6.2 Interpolação de Lagrange

Interpolação de Lagrange[2] é uma técnica para a computação do polinômio interpolador $p(x)$ de um conjunto de pontos $\{(x_i, y_i)\}_{i=1}^{n}$ dados. A ideia consiste em escrever o polinômio interpolador na forma

$$p(x) = y_1 L_1(x) + y_2 L_2(x) + \cdots + y_n L_n(x), \tag{6.14}$$

onde $L_i(x)$ é chamado de i-ésimo polinômio de Lagrange e é definido como o polinômio de grau $n-1$ que satisfaz

$$L_i(x_j) = \begin{cases} 1 & , i = j \\ 0 & , i \neq j \end{cases} \tag{6.15}$$

Mais especificamente, temos que $L_i(x)$ tem raízes $\{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n\}$ e, portanto, pode ser decomposto na forma

$$L_i(x) = c_i(x - x_1)\cdots(x - x_{i-1})(x - x_i)\cdots(x - x_n). \tag{6.16}$$

Além disso, como $L_i(x_i) = 1$, temos

$$c_i = \frac{1}{(x_i - x_1)\cdots(x_i - x_{i-1})(x_i - x_i)\cdots(x_i - x_n)}. \tag{6.17}$$

Assim sendo, podemos concluir que

$$L_i(x) = \frac{(x - x_1)\cdots(x - x_{i-1})(x - x_{i+1})\cdots(x - x_n)}{(x_i - x_1)\cdots(x_i - x_{i-1})(x_i - x_{i+1})\cdots(x_i - x_n)}. \tag{6.18}$$

Exemplo 6.2.1. Consideremos o problema de encontrar o polinômio interpolador do conjunto de pontos $\{(-1, \ -1), (0, \ 1), (1, \ 1/2)\}$. Como temos 3 pontos, o polinômio tem grau 2 e pode ser escrito na seguinte forma de Lagrange

$$p(x) = y_1 L_1(x) + y_2 L_2(x) + y_3 L_3(x), \tag{6.19}$$

onde $y_1 = -1$, $y_2 = 1$ e $y_3 = 1/2$. Os polinômios de Lagrange são dados por

$$L_1(x) = \frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} = \frac{1}{2}x^2 - \frac{1}{2}x, \tag{6.20}$$

[2] Joseph-Louis Lagrange, 1736 - 1813, matemático italiano. Fonte: Wikipedia.

$$L_2(x) = \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)} = -x^2 + 1, \tag{6.21}$$

$$L_3(x) = \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)} = \frac{1}{2}x^2 + \frac{1}{2}x. \tag{6.22}$$

$$\tag{6.23}$$

E, então, temos o polinômio interpolador

$$p(x) = -1{,}25x^2 + 0{,}75x + 1. \tag{6.24}$$

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

```
x = [-1 0 1]';
y = [-1 1 0.5]';

paux = poly([x(2) x(3)]);
L1 = paux/polyval(paux,x(1))

paux = poly([x(1) x(3)]);
L2 = paux/polyval(paux,x(2))

paux = poly([x(1) x(2)]);
L3 = paux/polyval(paux,x(3))

p = y(1)*L1 + y(2)*L2 + y(3)*L3
```

6.2.1 Aproximação de funções

Polinômio interpoladores podem ser usados para a aproximação de funções. Podemos aproximar uma dada função f pelo polinômio interpolador de um conjunto de pontos selecionados $\{(x_i, y_i = f(x_i))\}_{i=1}^n$. De fato, o seguinte teorema nos fornece uma estimativa para o erro de uma tal interpolação.

Teorema 6.2.1. *(*Teorema de Lagrange) Sejam dados uma função f $n+1$-vezes continuamente diferenciável em um dado intervalo $[a,b]$ e n pontos $\{x_i\}_{i=1}^n \subset [a,b]$. Então, o polinômio interpolador do conjunto de pontos $\{x_i, y_i = f(x_i)\}_{i=1}^n$ satisfaz

$$f(x) = p(x) + \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-x_1)(x-x_2)\cdots(x-x_n). \tag{6.25}$$

Demonstração. Em construção ... □

Exemplo 6.2.2. Considere o problema de aproxima $f(x) = \text{sen}(x)$ pelo polinômio interpolador pelo conjunto de pontos $x_1 = 0$, $x_2 = \pi/2$ e $x_3 = \pi$. Isto, queremos determinar o polinômio $p(x)$ de grau 2 que interpola os pontos $\{(0,0), (\pi/2, 1), (\pi,0)\}$. Usando a técnica de Lagrange, obtemos

$$p(x) = -0{,}41x^2 + 1{,}3x, \tag{6.26}$$

com seus coeficientes arredondados para dois dígitos significativos. A Figura 6.2 mostra os esboços da função $f(x) = \text{sen}(x)$, dos pontos dados e do polinômio interpolador $p(x)$.

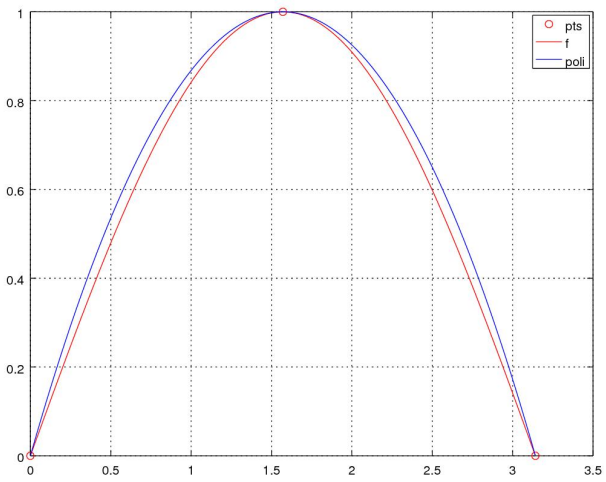

Figura 6.2: Esboços dos gráficos da função, dos pontos e do polinômio interpolador computado no Exemplo 6.2.2.

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

```
f = @(x) sin(x);
x = [0 pi/2 pi]';
y = f(x);

paux = poly([x(2) x(3)]);
```

```
L1 = paux/polyval(paux,x(1))

paux = poly([x(1) x(3)]);
L2 = paux/polyval(paux,x(2))

paux = poly([x(1) x(2)]);
L3 = paux/polyval(paux,x(3))

p = y(1)*L1 + y(2)*L2 + y(3)*L3

xx=linspace(0,pi);
plot(x,y,'ro',...
     xx,f(xx),'r-',...
     xx,polyval(p,xx),'b-')
legend("pts","f","poli")
```

Exercícios

Exercício 6.2.1. Use a técnica de interpolação de Lagrange para encontrar o polinômio interpolador que aproxima a função $f(x) = e^x$ pelos pontos $x_1 = 0$, $x_2 = 1$, $x_3 = 1{,}5$ e $x_4 = 2$.

Em construção ...

6.3 Diferenças divididas de Newton

Dado um conjunto de pontos $\{(x_i, y_i)\}_{i=1}^n$, o método das diferenças divididas de Newton[3] busca determinar o polinômio interpolar deste conjunto de pontos na forma

$$p(x) = a_1 + a_2(x - x_1) + a_3(x - x_1)(x - x_2) \tag{6.27}$$
$$+ \cdots + a_n(x - x_1) \cdot \cdots \cdot (x - x_{n-1}). \tag{6.28}$$

Por uma abordagem direta, temos que $p(x_i) = y_i$, $i = 1, 2, \ldots, n$, o que

[3] Carl Gustav Jacob Jacobi, 1804 - 1851, matemático alemão. Fonte: Wikipedia.

nos leva ao seguinte sistema triangular inferior

$$a_1 = y_1, \tag{6.29}$$
$$a_1 + a_2(x_2 - x_1) = y_2, \tag{6.30}$$
$$a_1 + a_2(x_3 - x_1) + a_3(x_3 - x_1)(x_3 - x_2) = y_3, \tag{6.31}$$
$$\vdots \tag{6.32}$$
$$a_1 + a_2(x_n - x_1) + \cdots + a_n(x_n - x_1) \cdot \cdots \cdot (x_n - x_{n-1}) = y_n. \tag{6.33}$$

Entretanto, existe uma forma mais eficiente de se determinar os coeficientes a_i, $i = 1, 2, \ldots, n$.

Denotemos por $p[x_j, x_{j+1}, \ldots, x_k](x)$ o polinômio interpolador do conjunto de pontos $\{(x_i, y_i)\}_{i=j}^{k}$. Então, temos a seguinte recursão

$$p[x_j] = y_j, \quad j = 1, 2, \ldots, n, \tag{6.34}$$

e

$$p[x_j, x_{j+1}, \ldots, x_k](x) = \frac{(x - x_j)p[x_{j+1}, \ldots, x_k](x) - (x - x_k)p[x_j, \ldots, x_{k-1}](x)}{x_k - x_j}, \tag{6.35}$$

para todo $n \geq k > j \geq 1$. De fato, (6.34) é trivial. Agora, denotando por $r(x)$ o lado direito da equação (6.35), vemos que $r(x)$ tem grau menor ou igual a $k - j$, o mesmo de $p[x_j, x_{j+1}, \ldots, x_k](x)$. Desta forma, para mostrar (6.35), basta verificarmos que $r(x)$ interpola o conjunto de pontos $\{(x_i, y_i)\}_{i=j}^{k}$. O que de fato ocorre:

$$r(x_j) = \frac{-(x_j - x_k)y_j}{x_k - x_j} = y_j, \tag{6.36}$$
$$r(x_l) = \frac{(x_l - x_j)y_l - (x_l - x_k)y_l}{x_k - x_j} = y_l,\ l = j + 1, \ldots, k - 1, \tag{6.37}$$
$$r(x_k) = \frac{(x_k - x_j)y_k}{x_k - x_j} = y_k. \tag{6.38}$$

Logo, pela unicidade do polinômio interpolador, temos demonstrado (6.35).

Observando que o polinômio interpolador $p(x)$ é igual a $p[x_1, \ldots, x_n](x)$, temos que (6.34)-(6.35) nos fornece uma forma de computar $p(x)$ recursivamente[4]. Além disso, observemos que $p[x_j, \ldots, x_{k-1}](x)$ e $p[x_j, \ldots, x_k]$ diferem

[4] De fato, o método de Neville consistem em computar $p(x)$ por esta recursão.

por um polinômio de grau $k-j$ com zeros x_j, x_{j+1}, ..., x_{k-1}. Logo, temos

$$p[x_j,\ldots,x_k](x) = p[x_j,\ldots,x_{k-1}](x) + f[x_j,\ldots,x_k](x-x_j)\cdot\cdots\cdot(x-x_{k-1}), \tag{6.39}$$

onde $f[x_j,\ldots,x_k]$ são coeficientes a determinar. Ainda, tomando $p[x_i] = f[x_i]$, temos

$$p[x_j,\ldots,x_k](x) = f[x_j] + f[x_j,x_{j+1}](x-x_j) + f[x_j,\ldots,x_k](x-x_j)\cdot\cdots\cdot(x-x_{k-1}). \tag{6.40}$$

Por fim, a recursão (6.34)-(6.35) nos mostra que as diferenças divididas Newton podem ser obtidas de

$$f[x_j] = y_j, \quad j = 1,2,\ldots,n, \tag{6.41}$$

$$f[x_j,\ldots,x_k] = \frac{f[x_{j+1},\ldots,x_k] - f[x_j,\ldots,x_{k-1}]}{x_k - x_j}, \tag{6.42}$$

para todo $n \geq k > j \geq 1$. E, temos o polinômio interpolador do conjunto de pontos $\{(x_i,y_i)\}_{i=1}^n$ dado por

$$p[x_1,\ldots,x_n](x) = f[x_1] + f[x_1,x_2](x-x_1) + \cdots + f[x_1,\ldots,x_n](x-x_1)\cdot\cdots\cdot(x-x_n). \tag{6.43}$$

Observação 6.3.1. A recursão (6.41)-(6.42) pode ser adequadamente organizada em uma matriz da forma

$$\begin{bmatrix} \boldsymbol{f[x_1]} & 0 & 0 & \ldots & 0 \\ f[x_2] & \boldsymbol{f[x_1,x_2]} & 0 & \ldots & 0 \\ f[x_3] & f[x_2,x_3] & \boldsymbol{f[x_1,x_2,x_3]} & \ldots & 0 \\ \vdots & \vdots & \vdots & \ldots & \vdots \\ f[x_n] & f[x_{n-1},x_n] & f[x_{n-2},x_{n-1},x_n] & \ldots & \boldsymbol{f[x_1,x_2,\ldots,x_n]} \end{bmatrix} \tag{6.44}$$

onde os elementos da diagonal correspondem aos coeficientes do polinômio interpolador na forma (6.43).

Exemplo 6.3.1. Consideremos o problema de encontrar o polinômio interpolador do conjunto de pontos $\{(-1,-1),(0,1),(1,1/2)\}$. Usando o método das diferenças divididas de Newton, escrevemos o polinômio na forma

$$p(x) = f[x_1] + f[x_1,x_2](x-x_1) + f[x_1,x_2,x_3](x-x_1)(x-x_2). \tag{6.45}$$

Então, computamos seus coeficientes pela recursão (6.41)-(6.42). Ou seja, temos

$$f[x_1] = -1,\ f[x_2] = 1,\ f[x_3] = 1/2. \tag{6.46}$$

Daí, segue

$$f[x_1,x_2] = \frac{f[x_2] - f[x_1]}{x_2 - x_1} = 2 \tag{6.47}$$

$$f[x_2,x_3] = \frac{f[x_3] - f[x_2]}{x_3 - x_2} = -\frac{1}{2} \tag{6.48}$$

$$\tag{6.49}$$

e, então,

$$f[x_1,x_2,x_3] = \frac{f[x_2,x_3] - f[x_1,x_2]}{x_3 - x_1} = -1{,}25. \tag{6.50}$$

Logo, o polinômio interpolador é

$$p(x) = 0{,}5 + 2(x+1) - 1{,}25(x+1)(x-1), \tag{6.51}$$

ou, equivalentemente,

$$p(x) = -1{,}25x^2 + 0{,}75x + 1. \tag{6.52}$$

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

```
x = [-1 0 1]';
y = [-1 1 0.5]';

F = zeros(3,3);

F(:,1) = y;

F(2,2) = (F(2,1)-F(1,1))/(x(2)-x(1));
F(3,2) = (F(3,1)-F(2,1))/(x(3)-x(2));

F(3,3) = (F(3,2)-F(2,2))/(x(3)-x(1));

p = F(1,1);
p = [0 p] + F(2,2)*poly(x(1));
p = [0, p] + F(3,3)*poly([x(1),x(2)])
```

Exercícios

Exercício 6.3.1. Use o método das diferenças divididas de Newton para encontrar o polinômio interpolador que aproxima a função $f(x) = e^x$ pelos pontos $x_1 = 0$, $x_2 = 1$, $x_3 = 1{,}5$ e $x_4 = 2$.

Em construção ...

6.4 Spline cúbico

Dado um conjunto de pontos $\{(x_i,y_i)\}_{i=1}^n$, um spline cúbico é uma função duas vezes continuamente diferenciável da forma

$$s(x) = \begin{cases} s_{11}(x-x_1)^3 + s_{12}(x-x_1)^2 + s_{13}(x-x_1) + s_{14} & ,x_1 \leq x < x_2, \\ s_{21}(x-x_2)^3 + s_{22}(x-x_2)^2 + s_{23}(x-x_2) + s_{24} & ,x_2 \leq x < x_3, \\ & ,\vdots \\ s_{n-1,1}(x-x_2)^3 + s_{n-1,2}(x-x_2)^2 + s_{n-1,3}(x-x_2) + s_{n-1,4} & ,x_{n-1} \leq x \leq x_n. \end{cases} \tag{6.53}$$

que satisfaz as seguintes propriedades

1. $s(x_i) = y_i$ para $i = 1, 2, \ldots, n$,
2. $s_j(x_j) = s_{j+1}(x_j)$ para todo $j = 1{,}2, \ldots, n-2$,
3. $s'_j(x_j) = s'_{j+1}(x_j)$ para todo $j = 1{,}2, \ldots, n-2$,
4. $s''_j(x_j) = s''_{j+1}(x_j)$ para todo $j = 1{,}2, \ldots, n-2$.

Observemos que o spline tem $4(n-1)$ coeficientes a determinar, enquanto que as condições acima nos fornecem $4n-6$ equações. Assim sendo, nota-se a determinação de um spline requer ainda 2 condições. Conforme a escolha destas condições, diferentes splines cúbicos são computados.

6.4.1 Spline *Not-a-knot*

A condição *not-a-knot* exige que o spline cúbico tenha derivada terceira contínua nos pontos x_2 e x_{n-1}, i.e.

$$s'''_1(x_2) = s'''_2(x_2) \quad \text{e} \quad s'''_{n-2}(x_{n-1}) = s'''_{n-1}(x_{n-1}). \tag{6.54}$$

Exemplo 6.4.1. Consideremos o problema de aproximar a função $f(x) = \text{sen}(x)$ pelo spline cúbico *not-a-knot* com pontos suporte $x_1 = 0$, $x_2 = \pi/3$, $x_3 = \pi/6$ e $x_4 = \pi/2$. Na Figura 6.3 temos os esboços de f e do spline cúbico computado.

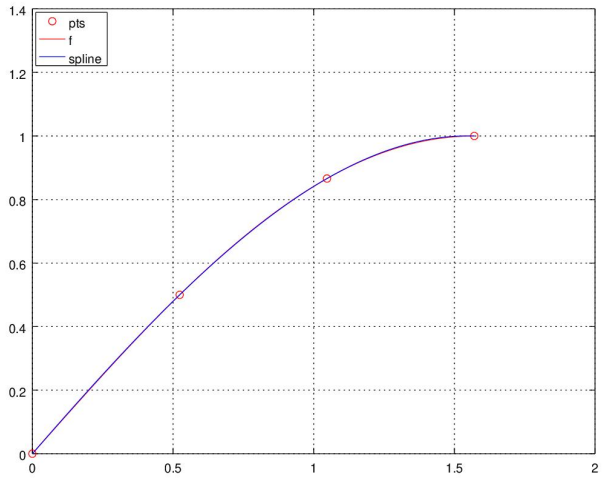

Figura 6.3: Esboço dos gráficos da função $f(x) = \text{sen}(x)$ e do spline cúbico computado no Exemplo 6.4.1.

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

```
x = [0 pi/3 pi/6 pi/2]';
y = sin(x);

s = spline(x,y)

xx=linspace(0,pi/2);
plot(x,y,'ro',...
     xx,sin(xx),'r-',...
     xx,ppval(s,xx),'b-');grid
legend("pts","f","spline","location","northwest")
```

6.4.2 Spline fixado

Os splines cúbicos fixados são obtidos com as condições de fronteira

$$s'(x_1) = y'_1, \quad \text{e} \quad s'(x_n) = y'_n, \tag{6.55}$$

onde y'_1 e y'_n são escalares dados. Quando usamos splines para aproximarmos uma dada função f, usualmente, escolhemos $y'_1 = f'(x_1)$ e $y'_n = f'(x_n)$.

Exemplo 6.4.2. Consideremos o problema de aproximar a função $f(x) = \text{sen}(x)$ pelo spline cúbico fixado com pontos suporte $x_1 = 0$, $x_2 = \pi/3$, $x_3 = \pi/6$ e $x_4 = \pi/2$. Na Figura 6.4 temos os esboços de f e do spline cúbico computado.

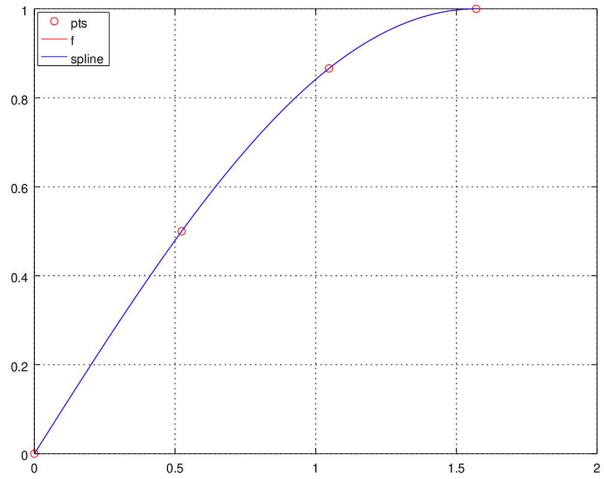

Figura 6.4: Esboço dos gráficos da função $f(x) = \text{sen}(x)$ e do spline cúbico computado no Exemplo 6.4.2.

No `GNU Octave`, podemos fazer as computações acima com o seguinte código:

```
x = [0 pi/3 pi/6 pi/2]';
y = sin(x);

s = spline(x,[1;y;0])
```

```
xx=linspace(0,pi/2);
plot(x,y,'ro',...
     xx,sin(xx),'r-',...
     xx,ppval(s,xx),'b-');grid
legend("pts","f","spline","location","northwest")
```

Exercícios

Em construção ...

Capítulo 7

Aproximação por mínimos quadrados

7.1 Problemas lineares

Dado um conjunto de n pontos $\{(x_i,y_i)\}_{i=1}^n$, $x_i \neq x_j$ para $i \neq j$, e uma família de $m \leq n$ funções $\{f_i(x)\}_{i=1}^m$, o problema linear de aproximação por mínimos quadrados consiste em determinar os m coeficientes $\{c_i\}_{i=1}^m$ tal que a função

$$f(x;c) = \sum_{j=1}^{m} c_j f_j(x) \tag{7.1}$$

$$= c_1 f_1(x) + c_2 f_2(x) + c_3 f_3(x) + \cdots + c_m f_m(x) \tag{7.2}$$

aproxime o conjunto de pontos dados no sentido de mínimos quadrados, i.e. o vetor dos coeficientes $c = (c_1, c_2, \ldots, c_m)$ é solução do seguinte problema linear de minimização

$$\min_c \left\{ E := \sum_{i=1}^{n} (y_i - f(x_i;c))^2 \right\}. \tag{7.3}$$

A fim de trabalharmos com uma notação mais compacta, definimos o resíduo $r(c) = (r_1(c), r_2(c), \ldots, r_n(c))$, onde $r_i(c) := y_i - f(x_i)$ e $c = (c_1, c_2, \ldots, c_m)$. Com esta notação, o problema de mínimos quadrados se resume a resolver

$$\min_c \{E := \|r(c)\|_2^2\}. \tag{7.4}$$

7.1.1 Método das equações normais

A fim de resolver o problema de mínimos quadrados (7.4), observamos que o erro quadrático

$$
\begin{aligned}
E &= \|r(c)\|_2^2 && (7.5)\\
&= \sum_{i=1}^{n} r_i(c)^2 && (7.6)\\
&= \sum_{i=1}^{n} \left(y_i - f(x_i; c)\right)^2 && (7.7)\\
&= \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{m} c_j f_j(x_i)\right)^2 && (7.8)\\
&= \|y - Ac\|_2^2, && (7.9)
\end{aligned}
$$

onde $y = (y_1, y_2, \ldots, y_n)$ e

$$
A := \begin{bmatrix}
f_1(x_1) & f_2(x_1) & \cdots & f_m(x_1)\\
f_1(x_2) & f_2(x_2) & \cdots & f_m(x_2)\\
\vdots & \vdots & \vdots & \vdots\\
f_1(x_n) & f_2(x_n) & \cdots & f_m(x_n)
\end{bmatrix}. \qquad (7.10)
$$

Os parâmetros c_j que minimizam o erro E são solução do seguinte sistema de equações

$$
\frac{\partial E}{\partial c_j} = 2\sum_{i=0}^{n} r_i(c)\frac{\partial}{\partial c_j} r_i(c) = 0, \qquad (7.11)
$$

onde $j = 1, 2, \ldots, m$. Ou, em uma notação mais apropriada,

$$
\begin{aligned}
\nabla_c E = 0 &\Leftrightarrow A^T r(c) = 0 && (7.12)\\
&\Leftrightarrow A^T(y - Ac) = 0 && (7.13)\\
&\Leftrightarrow A^T Ac = A^T y. && (7.14)
\end{aligned}
$$

Portanto, o problema linear de mínimos quadrados se resume em resolver as chamadas **equações normais**

$$
A^T Ac = A^T y. \qquad (7.15)
$$

Logo, o problema linear de mínimos quadrados (7.4) reduz-se a resolver o sistema linear (7.15) para c. Isto nos leva a questão de verificar se A^TA é invertível. De sorte, da disciplina de álgebra linear temos o seguinte teorema.

Teorema 7.1.1. *A matriz A^TA é positiva definida se, e somente se, as colunas de A são linearmente independentes (i.e. posto$(A) = n$).*

Demonstração. Se as colunas de A são linearmente independentes, então $x \neq 0$ implica $Ax \neq 0$ e, equivalentemente, $x^TA^T \neq 0$. Portanto, $x \neq 0$ implica $x^TA^TAx = \|Ax\|_2^2 > 0$, o que mostra que A^TA é positiva definida.

Suponhamos, agora, que as colunas de A não são linearmente independentes. Então, existe $x_0 \neq 0$ tal que $Ax_0 = 0$. Mas, então, $x_0^TA^TAx_0 = 0$, o que mostra que A^TA não é positiva definida. □

Este teorema nos fornece uma condição suficiente para a existência (e unicidade) de solução do problema linear de mínimos quadrados. Mais especificamente, se as colunas da matriz A são linearmente independentes, então os coeficientes da função $f(x)$ que melhor ajustam os pontos dados são

$$c = (A^TA)^{-1}A^Ty. \tag{7.16}$$

Exemplo 7.1.1. (Ajuste de polinômios) Considere o problema de ajustar o conjunto de pontos

i	1	2	3	4
x_i	-1	0	1	1,5
y_i	1,2	$-0,1$	0,7	2,4

por um polinômio quadrático da forma

$$p(x) = p_1x^2 + p_2x + p_3 \tag{7.17}$$

no sentido de mínimos quadrados.

Neste caso, a família de funções do problema de mínimos quadrados é $f_1(x) = x^2$, $f_2(x) = x$ e $f_3(x) = 1$. Assim sendo, os coeficientes $p = (p_1, p_2, p_3)$ são solução do seguinte sistema linear

$$A^TAp = A^Ty, \tag{7.18}$$

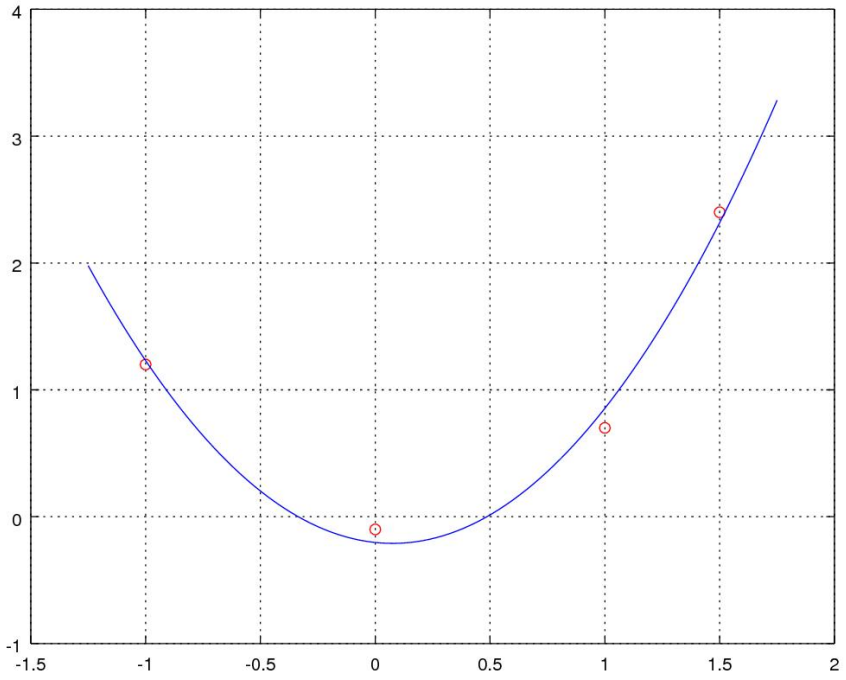

Figura 7.1: Esboço do polinômio ajustado no Exemplo 7.1.1.

onde $y = (y_1, y_2, y_3)$ e

$$A := \begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \\ x_4^2 & x_4 & 1 \end{bmatrix}. \tag{7.19}$$

Emfim, resolvendo as equações normais (7.18), obtemos

$$p(x) = 1{,}25x^2 - 0{,}188x - 0{,}203. \tag{7.20}$$

A Figura 7.1 mostra um esboço dos pontos (em vermelho) e do polinômio ajustado (em azul).

Os coeficientes e um esboço do polinômio ajustado podem ser obtidos no `GNU Octave` com o seguinte código:

```
#pontos
x = [-1 0 1 1.5]';
y = [1.2, -0.1, 0.7, 2.4]';

#resol. as eqs. normais
A = [x.^2 x.^1 x.^0];
p = inv(A'*A)*A'*y

#esboco do pol. ajustado
xx = linspace(-1.25,1.75);
plot(x,y,'ro',...
     xx,polyval(p,xx));grid
```

Exemplo 7.1.2. (Ajuste de curvas) Consideremos o mesmo conjunto de pontos do exemplo anterior (Exemplo 7.1.1). Aqui, vamos ajustar uma curva da forma

$$f(x) = c_1 \operatorname{sen}(x) + c_2 \cos(x) + c_3 \tag{7.21}$$

no sentido de mínimos quadrados. Para tanto, formamos a matriz

$$A := \begin{bmatrix} \operatorname{sen}(x_1) & \cos(x_1) & 1 \\ \operatorname{sen}(x_2) & \cos(x_2) & 1 \\ \operatorname{sen}(x_3) & \cos(x_3) & 1 \\ \operatorname{sen}(x_4) & \cos(x_4) & 1 \end{bmatrix} \tag{7.22}$$

e, então, resolvemos as equações normais $A^T Ac = A^T y$ para o vetor de coeficientes $c = (c_1, c_2)$. Fazendo isso, obtemos $c_1 = -0{,}198$, $c_2 = -2{,}906$ e $c_3 = 2{,}662$. A Figura 7.2 mostra um esboço da curva ajustada (linha azul) aos pontos dados (círculos vermelhos).

Os coeficientes e um esboço do polinômio ajustado podem ser obtidos no `GNU Octave` com o seguinte código:

```
#pontos
x = [-1 0 1 1.5]';
y = [1.2, -0.1, 0.7, 2.4]';

#resol. as eqs. normais
A = [sin(x) cos(x) ones(4,1)];
```

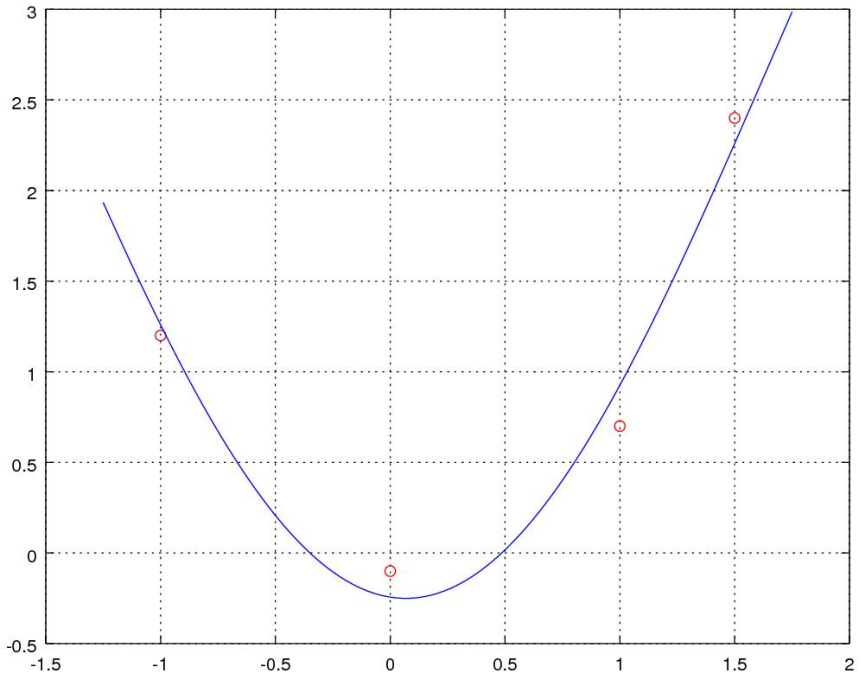

Figura 7.2: Esboço da curva ajustada no Exemplo 7.1.2.

```
c = inv(A'*A)*A'*y

#curva ajustada
f = @(x) c(1)*sin(x) + c(2)*cos(x) + c(3)

#esboco da fun. ajustada
xx = linspace(-1.25,1.75);
plot(x,y,'ro',...
     xx,f(xx));grid
```

Exemplo 7.1.3. (Um problema não linear) Consideremos o problema de ajustar, no sentido de mínimos quadrados, à função

$$f(x) = c_1 e^{c_2 x} \tag{7.23}$$

ao seguinte conjunto de pontos

i	1	2	3	4
x_i	-1	0	1	1,5
y_i	8,0	1,5	0,2	0,1

Aqui, temos um problema não linear de mínimos quadrados que pode ser transformado em um problema linear fazendo-se

$$y = c_1 e^{c_2 x} \Rightarrow \ln y = \ln c_1 e^{c_2 x} \tag{7.24}$$
$$\Rightarrow \ln y = \ln c_1 + c_2 x. \tag{7.25}$$

Isto é, denotando $d_1 := \ln c_1$ e $d_2 := c_2$, o problema se resume a ajustar uma reta $r(x) = d_1 + d_2 x$ ao conjunto de pontos $\{(x_i, \ln y_i)\}_{i=1}^4$.

Para resolver o problema transformado, formamos a matriz

$$A := \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ 1 & x_4 \end{bmatrix} \tag{7.26}$$

e, então, resolvemos as equações normais $A^T A d = A^T \ln y$, com $\ln y = (\ln y_1, \ln y_2, \ln y_3, \ln y_4)$, donde obtemos $d_1 = 0{,}315$ e $d_2 = -1{,}792$. Das definições de d_1 e d_2, temos $c_2 = d_2 = -1{,}792$ e $c_1 = e^{d_1} = 1{,}371$. A Figura 7.3 mostra um esboço da curva $f(x) = c_1 e^{c_2 x}$ ajustada (linha azul) aos pontos dados (círculos vermelhos).

O ajuste e um esboço da função ajustada podem ser feitos no `GNU Octave` com o seguinte código:

```
#pontos
x = [-1 0 1 1.5]';
y = [8.0 1.5 0.2 0.1]';

#resol. as eqs. normais
A = [ones(4,1) x];
d = inv(A'*A)*A'*log(y)

#fun. ajustada
c = [exp(d(1)); d(2)]
f = @(x) c(1)*exp(c(2)*x);
```

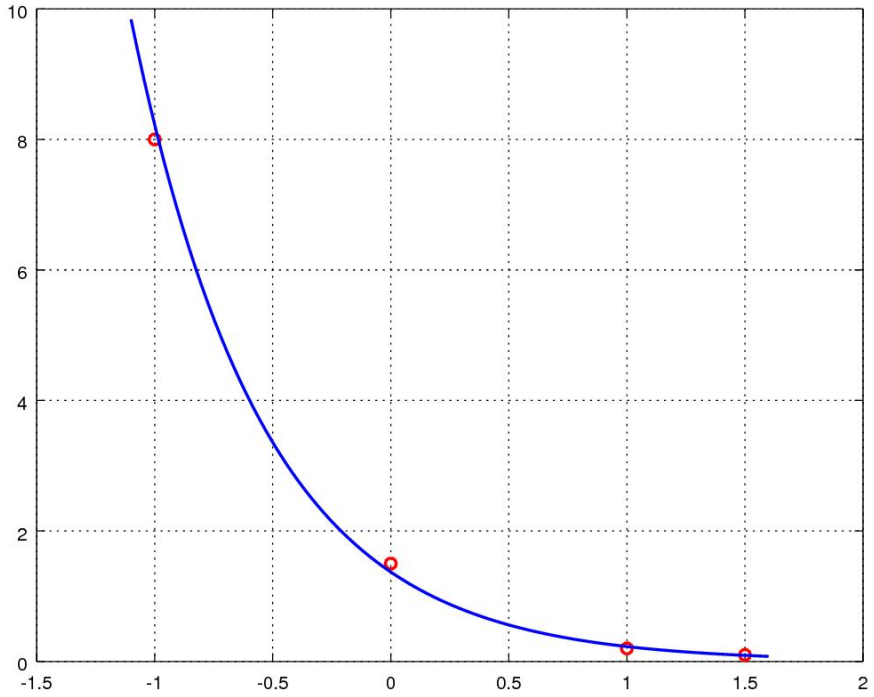

Figura 7.3: Esboço da curva ajustada no Exemplo 7.1.3.

```
#esboco da fun. ajustada
xx = linspace(-1.1,1.6);
plot(x,y,'ro','linewidth',1.5,...
     xx,f(xx),'b-','linewidth',1.5);grid
```

Exercícios

Exercício 7.1.1. Determine a reta $y = c_1x + c_2$ que melhor se ajusta, no sentido de mínimos quadrados, aos pontos

i	1	2	3	4	5
x_i	$-2{,}5$	$-1{,}3$	$0{,}2$	$1{,}7$	$2{,}3$
y_i	$3{,}8$	$1{,}5$	$-0{,}7$	$-1{,}5$	$-3{,}2$

Por fim, compute a norma L^2 do resíduo, i.e. $\|r(c)\|_2 = \|y - (c_1x - c_2)\|_2$

para os pontos dados.

Exercício 7.1.2. Determine o polinômio $y = c_1x^3 + c_2x^2 + c_3x + c_4$ que melhor se ajusta, no sentido de mínimos quadrados, aos pontos

i	1	2	3	4	5
x_i	$-2{,}5$	$-1{,}3$	0,2	1,7	2,3
y_i	3,8	0,5	2,7	1,2	$-1{,}3$

Por fim, compute a norma L^2 do resíduo, i.e. $\|r(c)\|_2$.

Exercício 7.1.3. Determine a curva $y = c_1 \operatorname{sen} x + c_2 \cos x + c_3$ que melhor se ajusta, no sentido de mínimos quadrados, aos pontos

i	1	2	3	4	5
x_i	$-2{,}5$	$-1{,}3$	0,2	1,7	2,3
y_i	3,8	0,5	2,7	1,2	$-1{,}3$

Por fim, compute a norma L^2 do resíduo, i.e. $\|r(c)\|_2$.

Exercício 7.1.4. Use a transformação $z = \ln y$ para ajustar, no sentido de mínimos quadrados, a curva $y = c_1 e^{c_2(x-c_3)^2}$ aos pontos

i	1	2	3	4	5	6
x_i	$-0{,}5$	0,5	1,3	2,1	2,7	3,1
y_i	0,1	1,2	2,7	0,9	0,2	0,1

7.2 Problemas não lineares

Um problema não linear de mínimos quadrados consiste em ajustar uma dada função $f(x;c)$ que dependa não linearmente dos parâmetros $c = (c_1, c_2, \ldots, c_m)$, $m \geq 1$, a um dado conjunto de $n \geq m$ pontos $\{(x_i, y_i)\}_{i=1}^n$. Mais especificamente, buscamos resolver o seguinte problema de minimização

$$\min_{\{c_1,c_2,\ldots,c_m\}} \left[E := \sum_{i=1}^{n} \left(y_i - f(x_i;c)\right)^2 \right]. \tag{7.27}$$

Aqui, denotaremos por $r(c) := (r_1(c), r_2(c), \ldots, r_n(c))$ o vetor dos resíduos $r_i(c) := y_i - f(x_i,c)$. Com isso, o problema se resume a encontrar o vetor de

parâmetros c que minimiza

$$E = \|r(c)\|_2^2. \tag{7.28}$$

Tais parâmetros são solução do seguinte sistema de equações

$$\frac{\partial E}{\partial c_j} = 2\sum_{i=1}^{n} r_i(c)\frac{\partial}{\partial c_j} r_i(c) = 0 \tag{7.29}$$

ou, equivalentemente, da equação

$$\nabla E = 0 \Leftrightarrow J_R^T(c) r(c) = 0, \tag{7.30}$$

onde

$$J_R(c) := \begin{bmatrix} \frac{\partial r_1}{\partial c_1} & \frac{\partial r_1}{\partial c_2} & \cdots & \frac{\partial r_1}{\partial c_m} \\ \frac{\partial r_2}{\partial c_1} & \frac{\partial r_2}{\partial c_2} & \cdots & \frac{\partial r_2}{\partial c_m} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial r_n}{\partial c_1} & \frac{\partial r_n}{\partial c_2} & \cdots & \frac{\partial r_n}{\partial c_m} \end{bmatrix} \tag{7.31}$$

é a jacobiana do resíduo r em relação aos parâmetros c.

Podemos usar o método de Newton para resolver (7.30). Para tanto, escolhemos uma aproximação inicial para $c^{(1)} = (c_1^{(1)}, c_2^{(1)}, \ldots, c_m^{(1)})$ e iteramos

$$H_R(c^{(k)})\delta^{(k)} = -J_R^T(c) r(c) \tag{7.32}$$

$$c^{(k+1)} = c^{(k)} + \delta^{(k)}, \tag{7.33}$$

onde $\delta^{(k)} = (\delta_1^{(k)}, \delta_2^{(k)}, \delta_m^{(k)})$ é a atualização de Newton (ou direção de busca) e $H_R(c) := [h_{p,q}(c)]_{p,q=1}^{m,m}$ é a matriz hessiana, cujos elementos são

$$h_{p,q} := \sum_{i=1}^{n} \left\{ \frac{\partial r_i}{\partial c_q}\frac{\partial r_i}{\partial c_p} + r_i \frac{\partial^2 r_i}{\partial c_q \partial c_p} \right\}. \tag{7.34}$$

Exemplo 7.2.1. Consideremos o problema de ajustar, no sentido de mínimos quadrados, a função

$$f(x; c) = c_1 e^{c_2 x} \tag{7.35}$$

ao seguinte conjunto de pontos

i	1	2	3	4
x_i	-1	0	1	1,5
y_i	8,0	1,5	0,2	0,1

Aqui, vamos utilizar a iteração de Newton para o problema de mínimos quadrados, i.e. a iteração dada em (7.32)-(7.33). Para tanto, para cada $i = 1, 2, 3, 4$, precisamos das seguintes derivadas parciais do resíduo $r_i(c) := y_i - c_1 e^{c_2 x_i}$:

$$\frac{\partial}{\partial c_1} r_i(c) = -e^{c_2 x_i}, \tag{7.36}$$

$$\frac{\partial}{\partial c_2} r_i(c) = -c_1 x_i e^{c_2 x_i}, \tag{7.37}$$

$$\frac{\partial^2}{\partial c_1^2} r_i(c) = 0, \tag{7.38}$$

$$\frac{\partial^2}{\partial c_1 \partial c_2} r_i(c) = \frac{\partial^2}{\partial c_2 \partial c_1} r_i(c) = -x_i e^{c_2 x_i}, \tag{7.39}$$

$$\frac{\partial^2}{\partial c_2^2} r_i(c) = -c_1 x_i^2 e^{c_2 x_i}. \tag{7.40}$$

Com isso e tomando $c^{(1)} = (1{,}4,\ -1{,}8)$ (motivado do Exemplo 7.1.3), computamos as iterações de Newton (7.32)-(7.33). Iterando até a precisão de $TOL = 10^{-4}$, obtemos a solução $c_1 = 1{,}471$ e $c_2 = -1{,}6938$. Na Figura 7.4 vemos uma comparação entre a curva aqui ajustada $(-)$ e aquela obtida no Exemplo 7.1.3 $(--)$.

O ajuste discutido neste exemplo pode ser computado no `GNU Octave` com o seguinte código:

```
#pontos
global x = [-1 0 1 1.5]';
global y = [8.0 1.5 0.2 0.1]';

#fun. objetivo
f = @(x,c) c(1)*exp(c(2)*x);

#residuo
r = @(c) y - f(x,c);
```

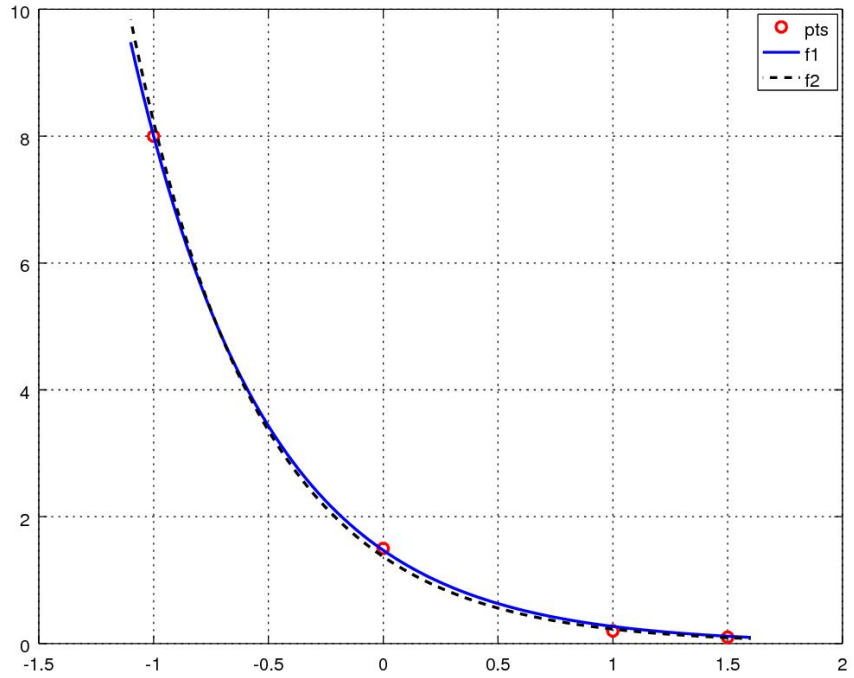

Figura 7.4: Esboço da curva ajustada no Exemplo 7.2.1.

```
#jacobiana
function A = J(c)
  global x
  A = zeros(4,2);
  A(:,1) = - exp(c(2)*x);
  A(:,2) = - c(1)*x.*exp(c(2)*x);
endfunction

#hessiana
function A = H(c)
  global x
  global y
  A = zeros(2,2);
  A = J(c)'*J(c);
```

```
  for i=1:4
    A(1,1) += 0;
    A(1,2) += (y(i) - c(1)*exp(c(2)*x(i))) * ...
              (- x(i)*exp(c(2)*x(i)));
    A(2,1) += (y(i) - c(1)*exp(c(2)*x(i))) * ...
              (- x(i)*exp(c(2)*x(i)));
    A(2,2) += (y(i) - c(1)*exp(c(2)*x(i))) * ...
              (- c(1)*x(i)^2*exp(c(2)*x(i)));
  endfor
endfunction

#aprox. inicial
c = [1.4 -1.8]';

#iteracoes de Newton
k=0;
do
  k+=1;
  delta = - inv(H(c))*J(c)'*r(c);
  c = c + delta;
  [k,c',norm(delta)]
until ((k>10) | (norm(delta)<1e-4))
```

Observamos que a solução obtida no exemplo anterior (Exemplo 7.2.1) difere da previamente encontrada no Exemplo 7.1.3. Naquele exemplo, os parâmetros obtidos nos fornecem $E = 6{,}8\text{e}-2$, enquanto que a solução do exemplo anterior fornece $E = 6{,}1\text{e}{-3}$. Isto é esperado, pois naquele exemplo resolvemos um problema aproximado, enquanto no exemplo anterior resolvemos o problema por si.

O emprego do método de Newton para o problema de mínimos quadrados tem a vantagem da taxa de convergência quadrática, entretanto requer a computação das derivadas parciais de segunda ordem do resíduo. Na sequência discutimos alternativas comumente empregadas.

7.2.1 Método de Gauss-Newton

O método de Gauss-Newton é uma técnica iterativa que aproxima o problema não linear de mínimos quadrados (7.27) por uma sequência de pro-

blemas lineares. Para seu desenvolvimento, começamos de uma aproximação inicial $c^{(1)} = (c_1^{(1)}, c_2^{(1)}, \ldots, c_m^{(1)})$ dos parâmetros que queremos ajustar. Também, assumindo que a n-ésima iterada $c^{(k)}$ é conhecida, faremos uso da aproximação de primeira ordem de $f(x,c)$ por polinômio de Taylor, i.e.

$$f(x; c^{(k+1)}) \approx f(x; c^{(k)}) + \nabla_c f(x; c^{(k)})(c^{(k+1)} - c^{(k)}), \tag{7.41}$$

onde

$$\nabla_c f(x; c) = \left[\frac{\partial}{\partial c_1} f(x; c) \;\; \frac{\partial}{\partial c_2} f(x; c) \;\; \cdots \;\; \frac{\partial}{\partial c_m} f(x; c) \right]. \tag{7.42}$$

O método consiste em obter a solução do problema não linear (7.27) pelo limite dos seguintes problemas lineares de mínimos quadrados

$$\min_{\delta^{(k)}} \left[\tilde{E} := \sum_{i=1}^{n} (y_i - f(x_i, c^{(k)}) - \nabla_c f(x_i; c^{(k)}) \delta^{(k)})^2 \right] \tag{7.43}$$

$$c^{(k+1)} = c^{(k)} + \delta^{(k)}. \tag{7.44}$$

Agora, usando a notação de resíduo $r(c) = y - f(x; c)$, observamos que (7.50) consiste no problema linear de mínimos quadrados

$$\min_{\delta^{(k)}} \| r(c^{(k)}) + J_R(c^{(k)}) \delta^{(k)} \|_2^2, \tag{7.45}$$

o qual é equivalente a resolver as equações normais

$$J_R^T(c^{(n)}) J_R(c^{(n)}) \delta^{(n)} = -J_R^T(c) r(c). \tag{7.46}$$

Com isso, dada uma aproximação inicial $c^{(1)}$, a **iteração do método de Gauss-Newton** consiste em

$$J_R^T(c^{(k)}) J_R(c^{(k)}) \delta^{(k)} = -J_R^T(c) r(c) \tag{7.47}$$

$$c^{(k+1)} = c^{(k)} + \delta^{(k)}. \tag{7.48}$$

Exemplo 7.2.2. A aplicação da iteração de Gauss-Newton ao problema de mínimos quadrados discutido no Exemplo 7.2.1 nos fornece a mesma solução obtida naquele exemplo (preservadas a aproximação inicial e a tolerância de precisão).

A implementação do método de Gauss-Newton para este problema no `GNU Octave` pode ser feita com o seguinte código:

```
#pontos
global x = [-1 0 1 1.5]';
y = [8.0 1.5 0.2 0.1]';

#fun. objetivo
f = @(x,c) c(1)*exp(c(2)*x);

#residuo
r = @(c) y - f(x,c);

#jacobiana
function A = J(c)
  global x
  A = zeros(4,2);
  A(:,1) = - exp(c(2)*x);
  A(:,2) = - c(1)*x.*exp(c(2)*x);
endfunction

#aprox. inicial
c = [1.4 -1.8]';

#iteracoes de Gauss-Newton
k=0;
do
  k+=1;
  delta = - inv(J(c)'*J(c))*J(c)'*r(c);
  c = c + delta;
  [k,c',norm(delta)]
until ((k>10) | (norm(delta)<1e-4))
```

O método de Gauss-Newton pode ser lentamente convergente para problemas muito não lineares ou com resíduos grandes. Nesse caso, métodos de Gauss-Newton com amortecimento são alternativas robustas [1, 4]. Na sequência, introduziremos um destes métodos, conhecido como método de Levenberg-Marquardt.

7.2.2 Método de Levenberg-Marquardt

O método de Levenberg-Marquardt é uma variação do método de Gauss-Newton no qual a direção de busca $\delta^{(n)}$ é obtida da solução do seguinte problema regularizado

$$\min_{\delta^{(k)}}\{\|r(c^{(k)}) + J_R(c^{(k)})\delta^{(k)}\|_2^2 + \mu^{(k)}\|\delta^{(k)}\|_2^2\} \tag{7.49}$$

ou, equivalentemente,

$$\min_{\delta^{(k)}} \left\| \begin{bmatrix} r(c^{(k)}) \\ 0 \end{bmatrix} + \begin{bmatrix} J_R(c^{(k)}) \\ \mu^{(k)} I \end{bmatrix} \delta^{(k)} \right\|_2^2 \tag{7.50}$$

A taxa de convergência das iterações de Levenberg-Marquardt é sensível a escolha do parâmetro $\mu^{(k)} \geq 0$. Aqui, faremos esta escolha por tentativa e erro. O leitor pode aprofundar-se mais sobre esta questão na literatura especializada (veja, por exemplo, [1, 4]).

Observação 7.2.1. Quando $\mu^{(k)} \equiv 0$ para todo n, o método de Levenberg-Marquardt é equivalente ao método de Gauss-Newton.

Exemplo 7.2.3. Consideremos o problema de mínimos quadrados discutido no Exemplo 7.2.1. O método de Gauss-Newton falha para este problema se escolhermos, por exemplo, $c^{(1)} = (0, 0)$. Isto ocorre pois, para esta escolha de $c^{(1)}$, a jacobiana $J(c^{(1)})$ não tem posto completo. Entretanto, o método de Levenberg-Marquardt com $\mu^{(k)} = 0{,}1$ é convergente, mesmo para esta escolha de $c^{(1)}$.

A implementação no `GNU Octave` do método de Levenberg-Marquardt (com $\mu^{(k)} = 0{,}1$ constante) para este problema pode ser feita com o seguinte código:

```
#pontos
global x = [-1 0 1 1.5]';
y = [8.0 1.5 0.2 0.1]';

#fun. objetivo
f = @(x,c) c(1)*exp(c(2)*x);

#residuo
```

```
r = @(c) y - f(x,c);

#jacobiana
function A = JR(c)
  global x;
  A = zeros(4,2);
  A(:,1) = - exp(c(2)*x);
  A(:,2) = - c(1)*x.*exp(c(2)*x);
endfunction

#aprox. inicial
c = [0 0]';

#param. de amortecimento
mu = 0.1;

#iteracoes de Gauss-Newton
k=0;
do
  k+=1;
  JJ = [JR(c);mu*eye(2,2)];
  delta = - inv(JJ'*JJ)*JJ'*[r(c);zeros(2,1)];
  c = c + delta;
  printf("%d %1.1e %1.3e %1.3e\n", k,norm(delta),c')
until ((k>10) | (norm(delta)<1e-4))
```

Exercícios

Exercício 7.2.1. Use o método de Gauss-Newton para ajustar, no sentido de mínimos quadrados e com precisão de 10^{-4}, a curva $y = c_1 e^{c_2(x-c_3)^2}$ aos pontos

i	1	2	3	4	5	6
x_i	$-0{,}5$	0,5	1,3	2,1	2,7	3,1
y_i	0,1	1,2	2,7	0,9	0,2	0,1

Use as condições iniciais:

a) $c_1 = 2{,}1$, $c_2 = -1$ e $c_3 = 1{,}3$.

b) $c_1 = 1$, $c_2 = -1$ e $c_3 = -1$.

Exercício 7.2.2. Resolva o exercício anterior (Exercício 7.2.1) usando o método de Levenberg-Marquardt com amortecimento constante $\mu = 0{,}2$.

Capítulo 8

Derivação

Neste capítulo, discutimos os métodos fundamentais de derivação numérica de funções.

8.1 Derivadas de primeira ordem

A derivada de uma função f num ponto x é, por definição,

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}. \tag{8.1}$$

Assim sendo e assumindo $h > 0$[1] próximo de zero, temos que $f'(x)$ pode ser aproximada pela razão fundamental, i.e.

$$f'(x) \approx \underbrace{\frac{f(x+h) - f(x)}{h}}_{D_h f(x)}. \tag{8.2}$$

Analisando a Figura 8.1 vemos que, geometricamente, isto é análogo a aproximar a declividade da reta tangente ao gráfico da função f no ponto $(x,f(x))$ pela declividade da reta secante ao gráfico da função f pelos pontos $(x,f(x))$ e $(x+h,f(x+h))$.

[1] Para fixar notação, assumiremos $h > 0$ ao longo deste capítulo.

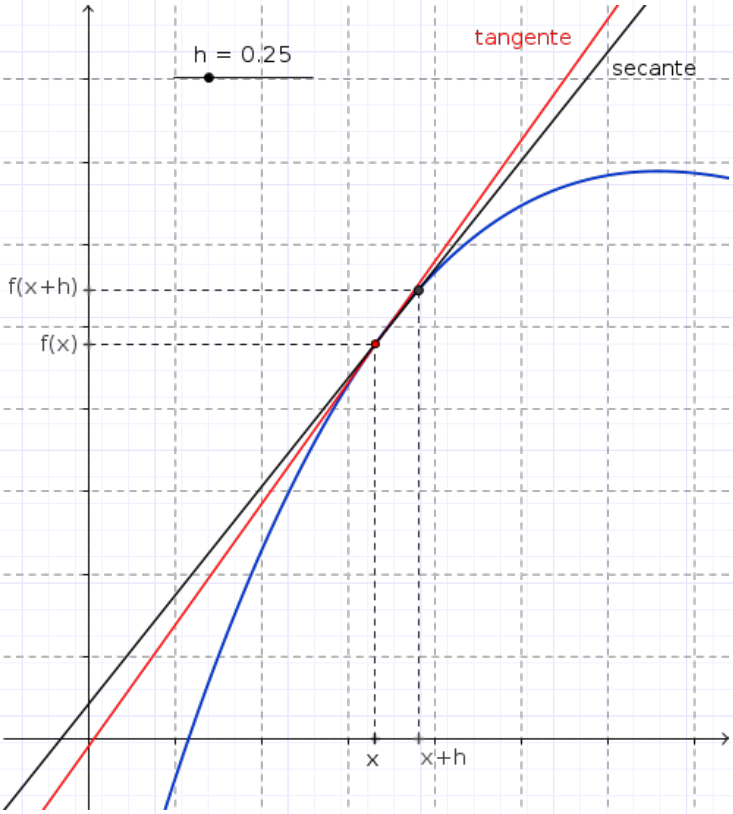

Figura 8.1: Interpretação geométrica da aproximação da derivada pela razão fundamental. Veja no Geogebra.

Exemplo 8.1.1. A derivada de $f(x) = \text{sen}(x)$ no ponto $\pi/3$ é $f'(\pi/3) = \cos(\pi/3) = 0{,}5$. Agora, usando a aproximação pela razão fundamental (8.2), temos

$$f'\left(\frac{\pi}{3}\right) \approx D_h f(x) = \frac{f\left(\frac{\pi}{3}+h\right) - f\left(\frac{\pi}{3}\right)}{h} \tag{8.3}$$

$$= \frac{\text{sen}\left(\frac{\pi}{3}\right) - \text{sen}\left(\frac{\pi}{3}\right)}{h}. \tag{8.4}$$

Na Tabela 8.1 temos os valores desta aproximação para diferentes escolhas da passo h.

h	$Df(\pi/3)$
10^{-1}	4,55902e−1
10^{-2}	4,95662e−1
10^{-3}	4,99567e−1
10^{-5}	4,99996e−1
10^{-10}	5.00000e−1

Tabela 8.1: Valores aproximados da derivada de $f(x) = \text{sen}(x)$ no ponto $x = \pi/6$ usado a expressão (8.2).

No `GNU Octave`, podemos fazer estes cálculos com o seguinte código:

```
f = @(x) sin(x);
Df = @(x,h) (f(x+h)-f(x))/h;
x=pi/3;
h=1e-1;
printf('%1.5e\n',Df(x,h))
```

A aproximação (8.2) é uma **fórmula de diferenças finitas**. Existem várias aproximações deste tipo que podem ser derivadas. Além disso, tais derivações nos permitem estimar o erro na utilização de tais fórmulas para a aproximação de derivadas. Na sequência, discutiremos o desenvolvimento de fórmulas de diferenças finitas usando polinômios de Taylor.

8.1.1 Desenvolvimento por polinômio de Taylor

Aqui, discutimos a obtenção de fórmulas de diferenças finitas via polinômio de Taylor.

Diferenças finitas progressiva de ordem h

A aproximação por polinômio de Taylor de grau 1 de uma dada função f em torno no ponto x é

$$f(x+h) = f(x) + hf'(x) + O(h^2). \tag{8.5}$$

Agora, isolando $f'(x)$, obtemos

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h). \tag{8.6}$$

Isto nos fornece a chamada fórmula de diferenças finitas progressiva de ordem h

$$D_{+,h}f(x) := \frac{f(x+h) - f(x)}{h}. \tag{8.7}$$

Observemos que a ordem da fórmula se refere a ordem do **erro de truncamento** com respeito ao passo h.

Exemplo 8.1.2. Consideremos o problema de aproximar a derivada da função $f(x) = \text{sen}(x)$ no ponto $\pi/3$. Usando a fórmula de diferenças finitas progressiva de ordem h obtemos

$$f'\left(\frac{\pi}{3}\right) \approx D_{+,h}f(x) = \frac{f\left(\frac{\pi}{3}+h\right) - f\left(\frac{\pi}{3}\right)}{h} \tag{8.8}$$

$$= \frac{\text{sen}\left(\frac{\pi}{3}+h\right) - \text{sen}\left(\frac{\pi}{3}\right)}{h}. \tag{8.9}$$

Na Tabela 8.2 temos os valores desta aproximação para diferentes escolhas de h, bem como, o erro absoluto da aproximação de $f'(\pi/3)$ por $D_{+,h}f(\pi/3)$.

No `GNU Octave`, podemos fazer estes cálculos com o seguinte código:

```
f = @(x) sin(x);
Df = @(x,h) (f(x+h)-f(x))/h;
x=pi/3;
h=1e-1;
printf('%1.5e %1.1e\n',Df(x,h),abs(cos(x)-Df(x,h)))
```

Tabela 8.2: Resultados referente ao Exemplo 8.1.2.

h	$D_{+,h}f(\pi/3)$	$\lvert f'(\pi/3) - D_{+,h}f(\pi/3)\rvert$
10^{-1}	4,55902e−1	4,4e−2
10^{-2}	4,95662e−1	4,3e−3
10^{-3}	4,99567e−1	4,3e−4
10^{-5}	4,99996e−1	4,3e−6
10^{-10}	5.00000e−1	4,1e−8

Observação 8.1.1. No exemplo acima (Exemplo 8.1.2), podemos observar que o erro absoluto na aproximação de $f'(x)$ por $D_{+,h}f(x)$ decresce conforme a ordem do erro de truncamento para valores moderados de h (veja, Tabela 8.2). Agora, para valores de h muito pequenos (por exemplo, $h = 10^{-10}$), o erro $|f'(x) - D_{+,h}f(x)|$ não segue mais a tendência de decaimento na mesma do de truncamento. Isto se deve a dominância dos erros de arredondamento para valores muito pequenos de h.

Para mais informações sobre o comportamento do erro de arredondamento em fórmulas de diferenças finitas, veja, por exemplo, REAMAT - Cálculo Numérico - Versão GNU Octave - Diferenças Finitas - Erro de arredondamento.

Diferenças finitas regressiva de ordem h

Substituindo h por $-h$ na equação (8.5), obtemos

$$f(x-h) = f(x) - hf'(x) + O(h^2), \tag{8.10}$$

donde obtemos a fórmula de diferenças finitas regressiva de ordem h

$$D_{-,h}f(x) = \frac{f(x) - f(x-h)}{h}. \tag{8.11}$$

Exemplo 8.1.3. Consideremos o problema de aproximar a derivada da função $f(x) = \operatorname{sen}(x)$ no ponto $\pi/3$. Usando a fórmula de diferenças finitas regressiva de ordem h obtemos

$$f'\left(\frac{\pi}{3}\right) \approx D_{-,h}f(x) = \frac{f\left(\frac{\pi}{3}\right) - f\left(\frac{\pi}{3} - h\right)}{h} \tag{8.12}$$

$$= \frac{\operatorname{sen}\left(\frac{\pi}{3}\right) - \operatorname{sen}\left(\frac{\pi}{3} - h\right)}{h}. \tag{8.13}$$

Na Tabela 8.3 temos os valores desta aproximação para diferentes escolhas de h, bem como, o erro absoluto da aproximação de $f'(\pi/3)$ por $D_{-,h}f(\pi/3)$.

Tabela 8.3: Resultados referente ao Exemplo 8.1.3.

h	$D_{-,h}f(\pi/3)$	$\lvert f'(\pi/3) - D_{-,h}f(\pi/3)\rvert$
10^{-1}	5,42432e−1	4,2e−2
10^{-2}	5,04322e−1	4,3e−3
10^{-3}	5,00433e−1	4,3e−4
10^{-5}	5,00004e−1	4,3e−6
10^{-10}	5.00000e−1	4,1e−8

No `GNU Octave`, podemos fazer estes cálculos com o seguinte código:

```
f = @(x) sin(x);
Df = @(x,h) (f(x)-f(x-h))/h;
x=pi/3;
h=1e-1;
printf('%1.5E %1.1E\n',Df(x,h),abs(cos(x)-Df(x,h)))
```

Diferenças finitas central de ordem h^2

Usando o polinômio de Taylor de grau 2 para aproximar a função $f(x)$ em torno de x, obtemos

$$f(x+h) = f(x) + hf'(x) + \frac{h}{2}f''(x) + O(h^3) \tag{8.14}$$

$$f(x-h) = f(x) - hf'(x) + \frac{h}{2}f''(x) + O(h^3). \tag{8.15}$$

Então, subtraindo esta segunda equação da primeira, temos

$$f(x+h) - f(x-h) = 2hf'(x) + O(h^3). \tag{8.16}$$

Agora, isolando $f(x)$

$$f'(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2), \tag{8.17}$$

o que nos fornece a chamada fórmula de diferenças finitas central de ordem h^2

$$D_{0,h^2}f(x) := \frac{f(x+h) - f(x-h)}{2h}. \tag{8.18}$$

Exemplo 8.1.4. Consideremos o problema de aproximar a derivada da função $f(x) = \text{sen}(x)$ no ponto $\pi/3$. Usando a fórmula de diferenças finitas central de ordem h^2 obtemos

$$f'\left(\frac{\pi}{3}\right) \approx D_{0,h^2}f(x) = \frac{f\left(\frac{\pi}{3}+h\right) - f\left(\frac{\pi}{3}-h\right)}{2h} \tag{8.19}$$

$$= \frac{\text{sen}\left(\frac{\pi}{3}+h\right) - \text{sen}\left(\frac{\pi}{3}-h\right)}{2h}. \tag{8.20}$$

Tabela 8.4: Resultados referente ao Exemplo 8.1.4.

h	$D_{0,h^2}f(\pi/3)$	$\lvert f'(\pi/3) - D_{0,h^2}f(\pi/3)\rvert$
10^{-1}	4,99167e−1	8,3e−04
10^{-2}	4,99992e−1	8,3e−06
10^{-3}	5,00000e−1	8,3e−08
10^{-5}	5,00000e−1	8,3e−10
10^{-10}	5.00000e−1	7,8e−12

Na Tabela 8.4 temos os valores desta aproximação para diferentes escolhas de h, bem como, o erro absoluto da aproximação de $f'(\pi/3)$ por $D_{0,h^2}f(\pi/3)$.

No `GNU Octave`, podemos fazer estes cálculos com o seguinte código:

```
f = @(x) sin(x);
Df = @(x,h) (f(x+h)-f(x-h))/(2*h);
x=pi/3;
h=1e-1;
printf('%1.5E %1.1E\n',Df(x,h),abs(cos(x)-Df(x,h)))
```

Exercícios

Exercício 8.1.1. Calcule aproximações da derivada de

$$f(x) = \frac{\text{sen}(x+2) - e^{-x^2}}{x^2 + \ln(x+2)} + x \tag{8.21}$$

no ponto $x = 2{,}5$ dadas pelas seguintes fórmulas de diferenças finitas com $h = 10^{-2}$:

a) progressiva de ordem h.

b) regressiva de ordem h.

c) central de ordem h^2.

Exercício 8.1.2. Considere a seguinte tabela de pontos

i	1	2	3	4	5	6
x_i	2,0	2,1	2,2	2,3	2,4	2,5
y_i	1,86	1,90	2,01	2,16	2,23	2,31

Calcule aproximações de dy/dx usando diferenças finitas centrais de ordem h^2 quando possível e, caso contrário, diferenças finitas progressiva ou regressiva conforme o caso.

8.2 Derivadas de segunda ordem

Diferentemente do que é costumeiro em técnicas analíticas, no âmbito da matemática numérica é preferível obter aproximações diretas de derivadas de segunda ordem, em vez de utilizar aproximações sucessivas de derivadas de primeira ordem. Na sequência, desenvolveremos e aplicaremos uma fórmula de diferenças finitas central para a aproximação de derivadas de segunda ordem.

Consideremos os seguintes polinômios de Taylor de grau 3 de $f(x)$ em torno do ponto x

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{3!}f'''(x) + O(h^4), \tag{8.22}$$

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{3!}f'''(x) + O(h^4). \tag{8.23}$$

$$\tag{8.24}$$

Somando estas duas equações, obtemos

$$f(x+h) + f(x-h) = 2f(x) + h^2 f''(x) + O(h^4). \tag{8.25}$$

Então, isolando $f''(x)$ temos

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2). \tag{8.26}$$

Isto nos leva a definição da **fórmula de diferenças finitas de ordem h^2 para a derivada segunda**

$$D^2_{0,h^2} f(x) := \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \tag{8.27}$$

Exemplo 8.2.1. Consideremos o problema de computar a derivada segunda de $f(x) = x^2 + \operatorname{sen} x$ no ponto $x = \pi/6$. Analiticamente, $f''(\pi/6) = 2 - \operatorname{sen}(\pi/6) = 1{,}5$. Numericamente, vamos explorar as seguintes duas aproximações:

a) Aplicação de sucessivas diferenças finitas centrais de ordem h^2 para derivada primeira, i.e.

$$f''(x) \approx D_{0,h^2} D_{0,h^2} f(x) = \frac{D_{0,h^2} f(x+h) - D_{0,h^2} f(x-h)}{2h} \tag{8.28}$$

b) Aplicação da fórmula de diferenças finitas central de ordem h^2 para a derivada segunda, i.e.

$$f''(x) \approx D^2_{0,h^2} f(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \tag{8.29}$$

Tabela 8.5: Resultados referente ao Exemplo 8.2.1. Notação: $\delta_{DD} := |f''(\pi/6) - D_{0,h^2} D_{0,h^2} f(\pi/6)|$ e $\delta_{D^2} := |f''(\pi/6) - D^2_{0,h^2} f(\pi/6)|$.

h	$D_{0,h^2} D_{0,h^2} f(\pi/6)$	δ_{DD}	$D^2_{0,h^2} f(\pi/6)$	δ_{D^2}
10^{-1}	1,50166	1,7e−03	1,50042	4,2e−04
10^{-2}	1,50002	1,7e−05	1,50000	4,2e−06
10^{-3}	1,50000	1,7e−07	1,50000	4,2e−08
10^{-5}	1,50000	1,2e−07	1,50000	1,2e−07

Na Tabela 8.5 temos os valores computados em ambos os casos e seus respectivos erros absolutos para diversas escolhas de h. Observamos que a aplicação da diferença finita D^2_{0,h^2} fornece resultados mais precisos (para valores moderados de h) do que as sucessivas aplicações de D_{0,h^2}. De fato, uma rápida inspeção de (8.28) mostra que

$$D_{0,h^2} D_{0,h^2} f(x) = \underbrace{\frac{f(x+2h) - 2f(x) + f(x-2h)}{4h^2}}_{D^2_{0,(2h)^2} f(x)}. \tag{8.30}$$

No GNU Octave, podemos fazer estes cálculos com o seguinte código:

```
f = @(x) sin(x) + x^2;
Df = @(x,h) (f(x+h)-f(x-h))/(2*h);
DDf = @(x,h) (Df(x+h,h)-Df(x-h,h))/(2*h);
D2f = @(x,h) (f(x+h) - 2*f(x) + f(x-h))/(h^2);
x=pi/6;
h=1e-1;
printf('%1.5E %1.1E %1.5E %1.1E\n',...
  DDf(x,h),abs(1.5-DDf(x,h)),...
  D2f(x,h),abs(1.5-D2f(x,h)))
```

Exercícios

Exercício 8.2.1. Use a fórmula de diferenças finitas central de ordem h^2 para computar aproximações da segunda derivada de

$$f(x) = \frac{\operatorname{sen}(x+2) - e^{-x^2}}{x^2 + \ln(x+2)} + x \tag{8.31}$$

no ponto $x = 2{,}5$. Para tanto, use os passos

a) $h = 10^{-1}$

b) $h = 10^{-2}$

c) $h = 10^{-3}$

d) $h = 10^{-4}$

Por fim, com base nos resultados obtidos, qual foi o maior passo que forneceu a aproximação com precisão de pelo menos 5 dígitos significativos? Justifique sua resposta.

Exercício 8.2.2. Considere a seguinte tabela de pontos

i	1	2	3	4	5	6
x_i	2,0	2,1	2,2	2,3	2,4	2,5
y_i	1,86	1,90	2,01	2,16	2,23	2,31

Calcule a aproximação d^2y/dx^2 no ponto $x = 2{,}2$ usando a fórmula de diferenças finitas central de ordem h^2.

8.3 Diferenças finitas por polinômios interpoladores

Aqui, discutimos a obtenção de fórmulas de diferenças finitas por polinômios interpoladores. Seja $p(x)$ o polinômio interpolador dos pontos $\{(x_i,f(x_i))\}_{i=1}^{n+1}$ de uma dada função $f(x)$, com $x_1 < x_2 < \cdots < x_{n+1}$. Então, pelo teorema de Lagrange temos

$$f(x) = p(x) + R_{n+1}(x), \tag{8.32}$$

onde $R(x)$ é o erro na aproximação de $f(x)$ por $p(x)$ e tem a forma

$$R_{n+1}(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{j=1}^{n+1}(x-x_j). \tag{8.33}$$

onde $\xi = \xi(x)$.

Deste modo, a ideia para obtermos as fórmulas de diferenças é aproximarmos $f'(x)$ por $p'(x)$. Entretanto, isto nos coloca a questão de estimarmos o erro $|f'(x) - p'(x)|$. Por sorte temos os seguinte teorema.

Teorema 8.3.1. *Seja $p(x)$ o polinômio interpolador de uma dada função $f(x)$ pelo pontos $\{(x_i, f(x_i))\}_{i=1}^{n+1}$, com $x_1 < x_2 < \cdots < x_{n+1}$. Se $f(x)$ é $(n+1)$ continuamente diferenciável, então o resíduo $R_{n+1}^{(k)}(x) = f^{(k)}(x) - p^{(k)}(x)$ é*

$$R_{n+1}^{(k)} = \frac{f^{(n+1)}(\eta)}{(n+1-k)!}\prod_{j=1}^{n+1-k}(x-\xi_j), \tag{8.34}$$

onde ξ_j é um ponto tal que $x_j < \xi_j < x_{j+k}$, $j = 1, 2, \ldots, n+1+k$, e $\eta = \eta(x)$ é algum ponto no intervalo de extremos x e ξ_j.

Demonstração. Veja [3, Ch.6, Sec.5]. □

8.3.1 Fórmulas de dois pontos

Seja $p(x)$ o polinômio interpolador de Lagrange de $f(x)$ pelos pontos $(x_1, f(x_1))$ e $(x_2, f(x_2))$, com $x_1 < x_2$, i.e.

$$f(x) = p(x) + R_2(x) \tag{8.35}$$

$$= f(x_1)\frac{x - x_2}{x_1 - x_2} + f(x_2)\frac{x - x_1}{x_2 - x_1} + R_2(x). \tag{8.36}$$

Denotando $h = x_2 - x_1$, temos

$$f(x) = f(x_1)\frac{x - x_2}{-h} + f(x_2)\frac{x - x_1}{h} + R_2(x). \tag{8.37}$$

e, derivando com respeito a x

$$f'(x) = \frac{f(x_2) - f(x_1)}{h} + R_2^{(1)}(x), \tag{8.38}$$

onde $R_2^{(1)}(x)$ é dado conforme o teorema 8.3.1.

Agora, escolhendo $x = x_1$, temos $x_2 = x_1 + h = x + h$ e, obtemos a **fórmula de diferenças finitas progressiva de ordem** h

$$f(x) = \underbrace{\frac{f(x+h) - f(x)}{h}}_{D_{+,h}f(x)} + O(h). \tag{8.39}$$

Se escolhermos $x = x_2$, temos $x_1 = x_2 - h = x - h$, obtemos a **fórmula de diferenças finitas regressiva de ordem** h

$$f(x) = \underbrace{\frac{f(x) - f(x-h)}{h}}_{D_{-,h}f(x)} + O(h). \tag{8.40}$$

Fórmulas de três pontos

Para obtermos fórmulas de diferenças finitas de três pontos consideramos o polinômio interpolador de Lagrange de $f(x)$ pelos pontos $(x_1, f(x_1))$, $(x_2, f(x_2))$ e $(x_3, f(x_3))$, $x_1 < x_2 < x_3$, i.e.

$$f(x) = f(x_1)\frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} \tag{8.41}$$

$$+ f(x_2)\frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} \tag{8.42}$$

$$+ f(x_3)\frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)} + R_3(x). \tag{8.43}$$

Derivando em relação a x, obtemos

$$f'(x) = f(x_1)\frac{(x_2 - x_3)(2x - x_2 - x_3)}{(x_1 - x_2)(x_1 - x_3)(x_2 - x_3)} \tag{8.44}$$

$$+ f(x_2)\frac{(x_1 - x_3)(-2x + x_1 + x_3)}{(x_1 - x_2)(x_1 - x_3)(x_2 - x_3)} \tag{8.45}$$

$$+ f(x_3)\frac{(x_1 - x_2)(2x - x_1 - x_2)}{(x_1 - x_2)(x_1 - x_3)(x_2 - x_3)} + R_3^{(1)}(x). \tag{8.46}$$

Aqui, podemos escolher por obter fórmulas de diferenças com passo constante ou não. Por exemplo, denotando $h_1 = x_2 - x_1$ e $h_2 = x_3 - x_2$ e escolhendo $x = x_1$, temos $x_2 = x + h_1$ e $x_3 = x + h_1 + h_2$. Fazendo estas substituições na expressão acima, obtemos seguinte fórmula de diferenças finitas progressiva

$$D_{+,h1,h2}f(x) = \frac{1}{h_1 h_2 (h_1 + h_2)}\left(-h_2(2h_1 + h_2)f(x)\right. \tag{8.47}$$

$$+ (h_1 + h_2)^2 f(x + h_1) \tag{8.48}$$

$$\left. - h_1^2 f(x + h_1 + h_2)\right). \tag{8.49}$$

Agora, assumindo um passo constante $h = h_1 = h_2$, obtemos a **fórmula de diferenças progressiva de ordem** h^2

$$D_{+,h^2}f(x) = \frac{1}{2h}\left[-3f(x) + 4f(x + h) - f(x + 2h)\right]. \tag{8.50}$$

Escolhendo $x = x_2$, $x_1 = x - h$ e $x_3 = x + h$ na equação (8.44), obtemos a **fórmula de diferenças finitas central de ordem** h^2

$$D_{0,h^2} = \frac{1}{2h}\left[f(x + h) - f(x - h)\right]. \tag{8.51}$$

Por fim, escolhendo $x = x_3$, $x_1 = x - 2h$ e $x_2 = x - h$ na equação (8.44), obtemos a **fórmula de diferenças finitas regressiva de ordem** h^2

$$D_{-,h^2} = \frac{1}{2h}\left[3f(x) - 4f(x - h) + f(x - 2h)\right]. \tag{8.52}$$

8.3.2 Fórmulas de cinco pontos

Aqui, usamos o polinômio interpolador de Lagrange da função $f(x)$ pelos pontos $(x_1, f(x_1))$, $(x_2, f(x_2))$, $(x_3, f(x_3))$ e $(x_5, f(x_5))$, com $x_1 < x_2 < x_3 < x_4 < x_5$. Isto nos fornece

$$f(x) = \sum_{i=1}^{5} f(x_i) \left(\prod_{j=1, j\neq i}^{5} \frac{x - x_j}{x_i - x_j} \right) + R_5(x). \tag{8.53}$$

Calculando a derivada em relação a x, temos

$$f'(x) = \sum_{i=1}^{5} f(x_i) \left(\sum_{\substack{j=1 \\ j\neq i}}^{5} \prod_{\substack{k=1 \\ k\neq i, k\neq j}}^{5} \frac{x - x_k}{x_i - x_k} \right) + R_5^{(1)}(x). \tag{8.54}$$

Por exemplo, substituindo $x_1 = x - 2h$, $x_2 = x - h$, $x_3 = x$, $x_4 = x + h$ e $x_5 = x + 2h$ na equação acima, obtemos fórmula de diferenças finitas central de ordem h^4

$$\begin{aligned} D_{+,h^4} f(x) := \frac{1}{12h} [&f(x - 2h) - 8f(x - h) \\ &+ 8f(x + h) - f(x + 2h)]. \end{aligned} \tag{8.55}$$

Exercícios

Exercício 8.3.1. Use a fórmula de diferenças finitas central de ordem h^4 para computar a aproximação da derivada de

$$f(x) = \frac{\operatorname{sen}(x + 2) - e^{-x^2}}{x^2 + \ln(x + 2)} + x \tag{8.56}$$

no ponto $x = 2{,}5$ com passo $h = 0{,}1$.

Exercício 8.3.2. Obtenha as seguintes fórmulas de diferenças finitas de 5 pontos com passo h constante e com:

a) 4 pontos para frente.

b) 1 ponto para traz e 3 pontos para frente.

c) 2 pontos para traz e 2 pontos para frente.

d) 3 pontos para traz e 1 pontos para frente.

e) 4 pontos para traz.

Exercício 8.3.3. Considere a seguinte tabela de pontos

i	1	2	3	4	5	6
x_i	2,0	2,1	2,2	2,3	2,4	2,5
y_i	1,86	1,90	2,01	2,16	2,23	2,31

Calcule a aproximação dy/dx nos pontos tabelados usando as fórmulas de diferenças finitas obtidas no exercício anteriores (Exercício 8.3.2). Para tanto, dê preferência para fórmulas centrais sempre que possível.

Capítulo 9

Técnicas de extrapolação

Neste capítulo, estudamos algumas técnicas de extrapolação, as quais serão usadas nos próximos capítulos.

9.1 Extrapolação de Richardson

Seja $F_1(h)$ uma aproximação de I tal que

$$I = F_1(h) + \underbrace{k_1 h + k_2 h^2 + k_3 h^3 + O(h^4)}_{\text{erro de truncamento}}. \tag{9.1}$$

Então, dividindo h por 2, obtemos

$$I = F_1\left(\frac{h}{2}\right) + k_1\frac{h}{2} + k_2\frac{h^2}{4} + k_3\frac{h^3}{8} + O(h^4). \tag{9.2}$$

Agora, de forma a eliminarmos o termo de ordem h das expressões acima, subtraímos (9.1) de 2 vezes (9.2), o que nos leva a

$$I = \underbrace{\left[F_1\left(\frac{h}{2}\right) + \left(F_1\left(\frac{h}{2}\right) - F_1(h)\right)\right]}_{F_2(h)} - k_2\frac{h^2}{2} - k_3\frac{3h^3}{4} + O(h^4). \tag{9.3}$$

Ou seja, denotando

$$F_2(h) := F_1\left(\frac{h}{2}\right) + \left(F_1\left(\frac{h}{2}\right) - F_1(h)\right) \tag{9.4}$$

temos que $N_2(h)$ é uma aproximação de I com erro de truncamento da ordem de h^2, uma ordem a mais de $N_1(h)$. Ou seja, esta combinação de aproximações de ordem de truncamento h nos fornece uma aproximação de ordem de truncamento h^2.

Analogamente, consideremos a aproximação de I por $N_2(h/2)$, *i.e.*

$$I = F_2\left(\frac{h}{2}\right) - k_2\frac{h^2}{8} - k_2\frac{3h^3}{32} + O(h^4) \tag{9.5}$$

Então, subtraindo (9.3) de 4 vezes (9.5) de, obtemos

$$I = \underbrace{\left[3F_2\left(\frac{h}{2}\right) + \left(F_2\left(\frac{h}{2}\right) - F_2(h)\right)\right]}_{F_3(h)} + k_3\frac{3h^3}{8} + O(h^4). \tag{9.6}$$

Observemos, ainda, que $N_3(h)$ pode ser reescrita na forma

$$F_3(h) = F_2\left(\frac{h}{2}\right) + \frac{F_2\left(\frac{h}{2}\right) - F_2(h)}{3}, \tag{9.7}$$

a qual é uma aproximação de ordem h^3 para I.

Para fazermos mais um passo, consideramos a aproximação de I por $F_3(h/2)$, i.e.

$$I = F_3\left(\frac{h}{2}\right) + k_3\frac{3h^3}{64} + O(h^4). \tag{9.8}$$

E, então, subtraindo (9.6) de 8 vezes (9.8), temos

$$I = \underbrace{\left[F_3\left(\frac{h}{2}\right) + \left(\frac{F_3\left(\frac{h}{2}\right) - F_3(h)}{7}\right)\right]}_{F_4(h)} + O(h^4). \tag{9.9}$$

Ou seja,

$$F_4(h) = \left[F_3\left(\frac{h}{2}\right) + \frac{F_3\left(\frac{h}{2}\right) - F_3(h)}{7}\right] \tag{9.10}$$

é uma aproximação de I com erro de truncamento da ordem h^4. Estes cálculos nos motivam o seguinte teorema.

Teorema 9.1.1. *Seja $F_1(h)$ uma aproximação de I com erro de truncamento da forma*

$$I - F_1(h) = \sum_{i=1}^{n} k_1 h^i + O(h^{n+1}). \tag{9.11}$$

Então, para $j \geq 2$,

$$F_j(h) := F_{j-1}\left(\frac{h}{2}\right) + \frac{F_{j-1}\left(\frac{h}{2}\right) - F_{j-1}(h)}{2^{j-1} - 1} \tag{9.12}$$

é uma aproximação de I com erro de truncamento da forma

$$\begin{aligned} I - F_j(h) = &\sum_{i=j}^{n} (-1)^{j-1} \frac{(2^{i-1} - 1)\prod_{l=1}^{j-2}\left(2^{i-l-1} - 1\right)}{2^{(j-1)(i-j+1)} d_j} k_i h^i \\ &+ O(h^{n+1}), \end{aligned} \tag{9.13}$$

onde d_j é dado recursivamente por $d_{j+1} = 2^{j-1} d_j$, com $d_2 = 1$.

Demonstração. Fazemos a demonstração por indução. O resultado para $j = 2$ segue de (9.3). Assumimos, agora, que vale

$$\begin{aligned} I - F_j(h) = &(-1)^{j-1} \frac{(2^{j-1} - 1)\prod_{l=1}^{j-2}\left(2^{j-l-1} - 1\right)}{2^{(j-1)} d_j} k_j h^j \\ &+ \sum_{i=j+1}^{n} (-1)^{j-1} \frac{(2^{i-1} - 1)\prod_{l=1}^{j-2}\left(2^{i-l-1} - 1\right)}{2^{(j-1)(i-j+1)} d_j} k_i h^i \\ &+ O(h^{n+1}). \end{aligned} \tag{9.14}$$

para $j \geq 2$. Então, tomamos

$$\begin{aligned} I - F_j\left(\frac{h}{2}\right) = &(-1)^{j-1} \frac{(2^{j-1} - 1)\prod_{l=1}^{j-2}\left(2^{j-l-1} - 1\right)}{2^{(j-1)} d_j} k_j \frac{h^j}{2^j} \\ &+ \sum_{i=j+1}^{n} (-1)^{j-1} \frac{(2^{i-1} - 1)\prod_{l=1}^{j-2}\left(2^{i-l-1} - 1\right)}{2^{(j-1)(i-j+1)} d_j} k_i \frac{h^i}{2^i} \\ &+ O(h^{n+1}). \end{aligned} \tag{9.15}$$

Agora, subtraímos (9.14) de 2^j vezes (9.15), o que nos fornece

$$I = \left[F_j\left(\frac{h}{2}\right) + \frac{F_j\left(\frac{h}{2}\right) - F_j(h)}{2^j - 1}\right] + \sum_{i=j+1}^{n} (-1)^{(j+1)-1} \frac{(2^{i-1}-1)\prod_{l=1}^{(j+1)-2}\left(2^{i-l-1}-1\right)}{2^{((j+1)-1)(i-(j+1)+1)}2^{j-1}d_j} k_i h^i + O(h^{n+1}). \tag{9.16}$$

□

Corolário 9.1.1. *Seja $F_1(h)$ uma aproximação de I com erro de truncamento da forma*

$$I - F_1(h) = \sum_{i=1}^{n} k_1 h^{2i} + O(h^{2n+2}). \tag{9.17}$$

Então, para $j \geq 2$,

$$F_j(h) := F_{j-1}\left(\frac{h}{2}\right) + \frac{F_{j-1}\left(\frac{h}{2}\right) - F_{j-1}(h)}{4^{j-1} - 1} \tag{9.18}$$

é uma aproximação de I com erro de truncamento da forma

$$I - F_j(h) = \sum_{i=j}^{n} (-1)^{j-1} \frac{(4^{i-1}-1)\prod_{l=1}^{j-2}\left(4^{i-l-1}-1\right)}{4^{(j-1)(i-j+1)}d_j} k_i h^{2i} + O(h^{n+1}), \tag{9.19}$$

onde d_j é dado recursivamente por $d_{j+1} = 4^{j-1}d_j$, com $d_2 = 1$.

Demonstração. A demonstração é análoga ao do Teorema 9.1.1. □

Exemplo 9.1.1. Dada uma função $f(x)$, consideremos sua aproximação por diferenças finitas progressiva de ordem h, i.e.

$$\underbrace{f'(x)}\, I = \underbrace{\frac{f(x+h) - f(x)}{h}}_{F_1(h)}$$

$$+\frac{f''(x)}{2}h+\frac{f'''(x)}{6}h^2+O(h^3). \tag{9.20}$$

Estão, considerando a primeira extrapolação de Richardson, temos

$$F_2(h)=F_1\left(\frac{h}{2}\right)+\left(F_1\left(\frac{h}{2}\right)-F_1(h)\right) \tag{9.21}$$

$$=4\frac{f(x+h/2)-f(x)}{h}-\frac{f(x+h)-f(x)}{h} \tag{9.22}$$

$$=\frac{-f(x+h)+4f(x+h/2)-3f(x)}{h}, \tag{9.23}$$

a qual é a fórmula de diferenças finitas progressiva de três pontos com passo $h/2$, i.e. $D_{+,(h/2)^2}f(x)$ (veja, Fórmula (8.50)).

Exemplo 9.1.2. Dada uma função $f(x)$, consideremos sua aproximação por diferenças finitas central de ordem h^2, i.e.

$$\underbrace{f'(x)}\, I=\underbrace{\frac{f(x+h)-f(x-h)}{2h}}_{F_1(h)}$$

$$-\frac{f'''}{6}h^2-\frac{f^{(5)}(x)}{120}h^4+O(h^6). \tag{9.24}$$

Estão, considerando a primeira extrapolação de Richardson, temos

$$F_2(h)=F_1\left(\frac{h}{2}\right)+\frac{\left(F_1\left(\frac{h}{2}\right)-F_1(h)\right)}{3} \tag{9.25}$$

$$=\frac{1}{6h}\left[f(x-h)-8f(x-h/2)+8f(x+h/2)-f(x+h)\right] \tag{9.26}$$

a qual é a fórmula de diferenças finitas central de cinco pontos com passo $h/2$, i.e. $D_{+,(h/2)^4}f(x)$ (veja, Fórmula (8.55)).

9.1.1 Sucessivas extrapolações

Sucessivas extrapolações de Richardson podem ser computadas de forma robusta com o auxílio de uma tabela. Seja $F_1(h)$ uma dada aproximação de uma quantidade de interesse I com erro de truncamento da forma

$$I-F_1(h)=k_1h+k_2h^2+k_3h^3+\cdots+k_nh^n+O(h^{n+1}). \tag{9.27}$$

Então, as sucessivas extrapolações $F_2(h)$, $F_3(h)$, $\ldots$, $F_n(h)$ podem ser organizadas na seguinte forma tabular

$$T = \begin{bmatrix} F_1(h) & & & & \\ F_1(h/2) & F_2(h) & & & \\ F_1(h/2^2) & F_2(h/2) & F_3(h) & & \\ \vdots & \vdots & \vdots & & \\ F_1(h/2^n) & F_2(h/2^{n-1}) & F_3(h/2^{n-2}) & \cdots & F_n(h) \end{bmatrix} \tag{9.28}$$

Desta forma, temos que

$$F_j\left(\frac{h}{2^{i-1}}\right) = t_{i,j-1} + \frac{t_{i,j-1} - t_{i-1,j-1}}{2^{j-1} - 1} \tag{9.29}$$

com $j = 2, 3, \ldots, n$ e $j \geq i$, onde $t_{i,j}$ é o elemento da i-ésima linha e j-ésima coluna da matriz T.

Exemplo 9.1.3. Consideremos o problema de aproximar a derivada da função $f(x) = \operatorname{sen}(x)$ no ponto $\pi/3$. Usando a fórmula de diferenças finitas progressiva de ordem h obtemos

$$f'\left(\frac{\pi}{3}\right) = \underbrace{\frac{f\left(\frac{\pi}{3} + h\right) - f\left(\frac{\pi}{3}\right)}{h}}_{F_1(h) := D_{+,h} f(\pi/3)} + \frac{f''(x)}{2}h + \frac{f'''(x)}{6}h^2 + \cdots \tag{9.30}$$

Na Tabela 9.1 temos os valores das aproximações de $f'(\pi/3)$ computadas via sucessivas extrapolações de Richardson a partir de (9.30) com $h = 0.1$.

Tabela 9.1: Resultados referente ao Exemplo 9.1.3.

$O(h)$	$O(h^2)$	$O(h^3)$	$O(h^4)$
4,55902e−1			
4,78146e−1	5,00389e−1		
4,89123e−1	5,00101e−1	5,00005e−1	
4,94574e−1	5,00026e−1	5,00001e−1	5,00000e−1

No `GNU Octave`, podemos fazer estes cálculos com o seguinte código:

```
#funcao
f = @(x) sin(x);
x=pi/3;

#aproximacao de ordem 1
dfp = @(x,h) (f(x+h)-f(x))/h;
h=0.1;

#tabela c/ sucessivas extrapolacoes
T=zeros(4,4);
for i=1:4
  T(i,1) = dfp(x,h/2^(i-1));
endfor
for j=2:4
  for i=j:4
    T(i,j) = T(i,j-1) ...
           + (T(i,j-1)-T(i-1,j-1))/(2^(j-1)-1);
  endfor
endfor
```

Exemplo 9.1.4. Novamente, consideremos o problema de aproximar a derivada da função $f(x) = \text{sen}(x)$ no ponto $\pi/3$. A fórmula de diferenças finitas central de ordem h^2 tem a forma

$$f'\left(\frac{\pi}{3}\right) = \underbrace{\frac{f\left(\frac{\pi}{3}+h\right) - f\left(\frac{\pi}{3}-h\right)}{2h}}_{F_1(h):=D_{0,h^2}f(\pi/3)} - \frac{f'''(x)}{6}h^2 + \frac{f^{(5)}(x)}{120}h^4 - \cdots \tag{9.31}$$

Na Tabela 9.2 temos os valores das aproximações de $f'(\pi/3)$ computadas via sucessivas extrapolações de Richardson a partir de (9.31) com $h = 1$.

No `GNU Octave`, podemos fazer estes cálculos com o seguinte código:

```
#funcao obj.
f = @(x) sin(x);
x=pi/3;

#aprox. O(h^2)
```

Tabela 9.2: Resultados referente ao Exemplo 9.1.4.

$O(h^2)$	$O(h^4)$	$O(h^6)$	$O(h^8)$
4,20735e−1			
4,79426e−1	4,98989e−1		
4,94808e−1	4,99935e−1	4,99998e−1	
4,98699e−1	4,99996e−1	5,00000e−1	5,00000e−1

```
h=1;
dfp = @(x,h) (f(x+h)-f(x-h))/(2*h);

#tabela c/ sucessivas extrapolacoes
T=zeros(4,4);
for i=1:4
  T(i,1) = dfp(x,h/2^(i-1));
endfor
for j=2:4
  for i=j:4
    T(i,j) = T(i,j-1) ...
           + (T(i,j-1)-T(i-1,j-1))/(4^(j-1)-1);
  endfor
endfor
```

9.1.2 Exercícios

Exercício 9.1.1. Mostre que a primeira extrapolação de Richardson de

$$D_{-,h}f(x) = \frac{f(x) - f(x-h)}{h} \tag{9.32}$$

é igual a

$$D_{-,(h/2)^2}f(x) = \frac{3f(x) - 4f(x-h) + f(x-2h)}{h}. \tag{9.33}$$

Exercício 9.1.2. Considere o problema de aproximar a derivada de

$$f(x) = \frac{\operatorname{sen}(x+2) - e^{-x^2}}{x^2 + \ln(x+2)} + x \tag{9.34}$$

no ponto $x = 2{,}5$. Para tanto, use de sucessivas extrapolações de Richardson a partir da aproximação por diferenças finitas:

a) progressiva de ordem h, com $h = 0{,}5$.

b) regressiva de ordem h, com $h = 0{,}5$.

c) central de ordem h^2, com $h = 0{,}5$.

Nas letras a) e b), obtenha as aproximações de ordem h^3 e, na letra c) obtenha a aproximação de ordem h^6.

Capítulo 10

Integração

Neste capítulo, discutimos os métodos numéricos fundamentais para a aproximação de integrais definidas de funções. Tais métodos são chamados de **quadraturas numéricas** e têm a forma

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{n} f(x_i)w_i, \tag{10.1}$$

onde x_i e w_i são, respectivamente, o i-ésimo nodo e o i-ésimo peso da quadratura, $i = 1, 2, \ldots, n$.

10.1 Regras de Newton-Cotes

Dada uma função $f(x)$ e um intervalo $[a, b]$, denotamos por

$$I := \int_a^b f(x)\,dx. \tag{10.2}$$

a integral de $f(x)$ no intervalo $[a, b]$. A ideia das regras de Newton-Cotes e aproximar I pela integral de um polinômio interpolador de $f(x)$ por pontos previamente selecionados.

Seja, então, $p(x)$ o polinômio interpolador de grau n de $f(x)$ pelos dados pontos $\{(x_i, f(x_i))\}_{i=1}^{n+1}$, com $x_1 < x_2 < \cdots < x_{n+1}$ e $x_i \in [a, b]$ para todo $i = 1, 2, \ldots, n + 1$. Então, pelo teorema de Lagrange, temos

$$f(x) = p(x) + R_{n+1}(x), \tag{10.3}$$

onde

$$p(x) = \sum_{i=1}^{n+1} f(x_i) \prod_{\substack{j=1 \\ j\neq i}}^{n+1} \frac{(x - x_j)}{x_i - x_j} \tag{10.4}$$

e

$$R_{n+1}(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{j=1}^{n+1} (x - x_j), \tag{10.5}$$

onde $\xi = \xi(x)$ pertencente ao intervalo $[x_1, x_{n+1}]$. Deste modo, temos

$$I := \int_a^b f(x) \tag{10.6}$$

$$= \int_a^b p(x)\,dx + \int_a^b R_{n+1}(x)\,dx \tag{10.7}$$

$$= \underbrace{\sum_{i=1}^{n+1} f(x_i) \int_a^b \prod_{\substack{j=1 \\ j\neq i}}^{n+1} \frac{(x - x_j)}{x_i - x_j)}\,dx}_{\text{quadratura}} + \underbrace{\int_a^b R_{n+1}(x)\,dx}_{\text{erro de truncamento}} \tag{10.8}$$

Ou seja, nas quadraturas (regras) de Newton-Cotes, os nodos são as abscissas dos pontos interpolados e os pesos são as integrais dos polinômios de Lagrange associados.

Na sequência, abordaremos as regras de Newton-Cotes mais usuais e estimaremos o erro de truncamento caso a caso. Para uma abordagem mais geral, recomenda-se consultar [3, Cap. 7,Sec. 1.1].

10.1.1 Regras de Newton-Cotes fechadas

As regras de Newton-Cotes fechadas são aqueles que a quadratura incluem os extremos do intervalo de integração, i.e. os nodos extremos são $x_1 = a$ e $x_{n+1} = b$.

Regra do trapézio

A regra do trapézio é obtida tomando-se os nodos $x_1 = a$ e $x_2 = b$. Então, denotando $h := b - a$[1], os pesos da quadratura são:

$$w_1 = \int_a^b \frac{x - b}{a - b}\,dx \tag{10.9}$$

[1]Neste capítulo, h é escolhido como a distância entre os nodos.

$$= \frac{(b-a)}{2} = \frac{h}{2} \tag{10.10}$$

e

$$w_2 = \int_a^b \frac{x-a}{b-a}\,dx \tag{10.11}$$

$$= \frac{(b-a)}{2} = \frac{h}{2}. \tag{10.12}$$

Agora, estimamos o erro de truncamento com

$$E := \int_a^b R_2(x)\,dx \tag{10.13}$$

$$= \int_a^b \frac{f''(\xi(x))}{2}(x-a)(x-b)\,dx \tag{10.14}$$

$$\leq C\left|\int_a^b (x-a)(x-b)\,dx\right| \tag{10.15}$$

$$= C\frac{(b-a)^3}{6} = O(h^3). \tag{10.16}$$

Portanto, a **regra do trapézio** é dada por

$$\int_a^b f(x)\,dx = \frac{h}{2}(f(a)+f(b)) + O(h^3). \tag{10.17}$$

Exemplo 10.1.1. Consideremos o problema de computar a integral de $f(x) = xe^{-x^2}$ no intervalo $[0, 1/4]$. Analiticamente, temos

$$I = \int_0^{1/4} xe^{-x^2}\,dx = -\frac{e^{-x^2}}{2}\Big|_0^{1/4} \tag{10.18}$$

$$= \frac{1-e^{-1/4}}{2} = 3{,}02935\text{e}{-2}. \tag{10.19}$$

Agora, usando a regra do trapézio, obtemos a seguinte aproximação para I

$$I \approx \frac{h}{2}(f(0)+f(1/2)) \tag{10.20}$$

$$= \frac{1/4}{2}\left(0 + \frac{1}{4}e^{-(1/4)^2}\right) = 2{,}93567\text{e}{-2}. \tag{10.21}$$

Podemos obter a aproximação dada pela regra do trapézio no `GNU Octave` com o seguinte código:

```
f = @(x) x*exp(-x^2);
a=0;
b=0.25;
h=b-a;
Itrap = (h/2)*(f(a)+f(b));
printf("%1.5E\n",Itrap)
```

Regra de Simpson

A regra de Simpson é obtida escolhendo-se os nodos $x_1 = a$, $x_2 = (a+b)/2$ e $x_3 = b$. Com isso e denotando $h = (b-a)/2$, calculamos os seguintes pesos:

$$w_1 = \int_a^b \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)}\,dx \tag{10.22}$$

$$= \frac{(b-a)}{6} = \frac{h}{3}, \tag{10.23}$$

$$w_2 = \int_a^b \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)}\,dx \tag{10.24}$$

$$= 4\frac{(b-a)}{6} = 4\frac{h}{3} \tag{10.25}$$

e

$$w_3 = \int_a^b \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)}\,dx \tag{10.26}$$

$$= \frac{(b-a)}{6} = \frac{h}{3}. \tag{10.27}$$

Isto nos fornece a chamada **regra de Simpson**

$$I \approx \frac{h}{3}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] \tag{10.28}$$

Nos resta estimar o erro de truncamento da regra de Simpson. Para tanto, consideramos a expansão em polinômio de Taylor de grau 3 de $f(x)$ em torno do ponto x_2, i.e.

$$f(x) = f(x_2) + f'(x_2)(x-x_2) + \frac{f''(x_2)}{2}(x-x_2)^2$$

$$+\frac{f'''(x_2)}{6}(x-x_2)^3 + \frac{f^{(4)}(\xi_1(x))}{24}(x-x_2)^4, \tag{10.29}$$

donde

$$\int_a^b f(x)\,dx = 2hf(x_2) + \frac{h^3}{3}f''(x_2) + \frac{1}{24}\int_a^b f^{(4)}(\xi_1(x))(x-x_2)^4\,dx. \tag{10.30}$$

Daí, usando da fórmula de diferenças finitas central de ordem h^2, temos

$$f''(x_2) = \frac{f(x_1) - 2f(x_2) + f(x_3)}{h^2} + O(h^2). \tag{10.31}$$

Ainda, o último termo da equação (10.30) pode ser estimado por

$$\left|\frac{1}{24}\int_a^b f^{(4)}(\xi_1(x))(x-x_2)^4\,dx\right| \leq C\left|\int_a^b (x-x_2)^4\,dx\right| \tag{10.32}$$

$$= C(b-a)^5 = O(h^5). \tag{10.33}$$

Então, de (10.30), (10.31) e (10.1.1), temos

$$\int_a^b f(x)\,dx = \frac{h}{3}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] + O(h^5), \tag{10.34}$$

o que mostra que a **regra de Simpson tem erro de truncamento da ordem** h^5.

Exemplo 10.1.2. Aproximando a integral dada no Exemplo 10.1.1 pela a regra de Simpson, temos

$$\int_0^{1/4} f(x)\,dx \approx \frac{1/8}{3}\left[f(0) + 4f\left(\frac{1}{8}\right) + f\left(\frac{1}{4}\right)\right] \tag{10.35}$$

$$= \frac{1}{24}\left[\frac{1}{2}e^{-(1/8)^2} + \frac{1}{4}e^{-(1/4)^2}\right] \tag{10.36}$$

$$= 3{,}02959\text{e}{-2}. \tag{10.37}$$

Podemos computar a aproximação dada pela regra de Simpson no `GNU Octave` com o seguinte código:

```
f = @(x) x*exp(-x^2);
a=0;
b=1/4;
h=(b-a)/2;
Isimp = (h/3)*(f(a)+4*f((a+b)/2)+f(b));
printf("%1.5E\n",Isimp)
```

10.1.2 Regras de Newton-Cotes abertas

As regras de Newton-Cotes abertas não incluem os extremos dos intervalos como nodos das quadraturas.

Regra do ponto médio

A regra do ponto médio é obtida usando apenas o nodo $x_1 = (a+b)/2$. Desta forma, temos

$$\int_a^b f(x)\,dx = \int_a^b f(x_1)\,dx + \int_a^b f'(\xi(x))(x-x_1)\,dx, \tag{10.38}$$

donde, denotando $h := (b-a)$, temos

$$\int_a^b f(x),dx = hf\left(\frac{a+b}{2}\right) + O(h^3). \tag{10.39}$$

Deixa-se para o leitor a verificação do erro de truncamento (veja, Exercício 10.1.3).

Exemplo 10.1.3. Aproximando a integral dada no Exemplo 10.1.1 pela a regra do ponto médio, temos

$$\begin{aligned}\int_0^{1/4} f(x)\,dx &\approx \frac{1}{4}f\left(\frac{1}{8}\right) && (10.40)\\ &= \frac{1}{32}e^{-(1/8)^2} && (10.41)\\ &= 3{,}07655\text{e}{-2} && (10.42)\end{aligned}$$

Podemos computar a aproximação dada pela regra do ponto médio no `GNU Octave` com o seguinte código:

```
f = @(x) x*exp(-x^2);
a=0;
b=0.25;
h=b-a;
Ipmd = h*f((a+b)/2);
printf("%1.5E\n",Ipmd)
```

Exercício

Exercício 10.1.1. Aproxime

$$\int_{-1}^{0} \frac{\operatorname{sen}(x+2) - e^{-x^2}}{x^2 + \ln(x+2)}\, dx \tag{10.43}$$

usando a:

a) regra do ponto médio.

b) regra do trapézio.

c) regra de Simpson.

Exercício 10.1.2. Considere a seguinte tabela de pontos

i	1	2	3	4	5	6
x_i	2,0	2,1	2,2	2,3	2,4	2,5
y_i	1,86	1,90	2,01	2,16	2,23	2,31

Assumindo que $y = f(x)$, calcule:

a) $\int_{2,1}^{2,3} f(x)\, dx$ usando a regra do ponto médio.

b) $\int_{2,0}^{2,5} f(x)\, dx$ usando a regra do trapézio.

c) $\int_{2,0}^{2,4} f(x)\, dx$ usando a regra de Simpson.

Exercício 10.1.3. Mostre que o erro de truncamento da regra do ponto médio é da ordem de h^3, onde h é o tamanho do intervalo de integração.

Exercício 10.1.4. Obtenha a regra de Newton-Cotes aberta de 2 pontos e estime seu erro de truncamento.

10.2 Regras compostas de Newton-Cotes

Regras de integração numérica compostas (ou quadraturas compostas) são aquelas obtidas da composição de quadraturas aplicadas as subintervalos do intervalo de integração. Mais especificamente, a integral de uma dada função $f(x)$ em um dado intervalo $[a, b]$ pode ser reescrita como uma soma de integrais em sucessivos subintervalos de $[a, b]$, i.e.

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} \int_{x_i}^{x_{i+1}} f(x)\,dx, \tag{10.46}$$

onde $a = x_1 < x_2 < \cdots < x_{n+1} = b$. Então, a aplicação de uma quadratura em cada integral em $[x_i, x_{i+1}]$, $i = 1, 2, \ldots, n$, nos fornece uma regra composta.

10.2.1 Regra composta do ponto médio

Consideremos uma partição uniforme do intervalo de integração $[a, b]$ da forma $a = \tilde{x}_1 < \tilde{x}_2 < \cdots < \tilde{x}_{n+1} = b$, com $h = x_{i+1} - x_i$, $i = 1, 2, \ldots, n$. Então, aplicando a regra do ponto médio a cada integral nos subintervalos $[\tilde{x}_i, \tilde{x}_{i+1}]$, temos

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} \int_{\tilde{x}_i}^{\tilde{x}_{i+1}} f(x)\,dx \tag{10.47}$$

$$= \sum_{i=1}^{n} \left[hf\left(\frac{\tilde{x}_i + \tilde{x}_{i+1}}{2}\right) + O(h^3) \right]. \tag{10.48}$$

Agora, observando que $h := (b - a)/n$ e escolhendo os nodos $x_i = a + (i - 1/2)h$, $i = 1, 2, \ldots, n$, obtemos a **regra composta do ponto médio com n subintervalos**

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} hf(x_i) + O(h^2). \tag{10.49}$$

Exemplo 10.2.1. Consideremos o problema de computar a integral de $f(x) = xe^{-x^2}$ no intervalo $[0, 1]$. Usando a regra composta do ponto médio com n subintervalos, obtemos a aproximação

$$\underbrace{\int_a^b f(x)\,dx}_{I} \approx \underbrace{\sum_{i=1}^{n} hf(x_i)}_{S}, \tag{10.50}$$

onde $h = 1/(4n)$ e $x_i = (i - 1/2)h$, $i = 1, 2, \ldots, n$. Na Tabela 10.1, temos as aproximações computadas com diversos números de subintervalos, bem como, seus erros absolutos.

Tabela 10.1: Resultados referentes ao Exemplo 10.2.1.

n	S	$\lvert I - S\rvert$
1	3,89400e−1	7,3e−2
10	3,16631e−1	5,7e−4
100	3,16066e−1	5,7e−6
1000	3.16060e−1	5,7e−8

Podemos fazer estas computações com o auxílio do seguinte código `GNU Octave`:

```
f = @(x) x*exp(-x^2);
a=0;
b=1;
n=10;
h=(b-a)/n;
s=0;
for i=1:n
  x=a+(i-1/2)*h;
  s+=h*f(x);
endfor
printf("%1.5E %1.1E\n",s,abs((1-e^(-1))/2-s))
```

10.2.2 Regra composta do trapézio

Para obtermos a regra composta do trapézio, consideramos uma partição uniforme do intervalo de integração $[a, b]$ da forma $a = x_1 < x_2 < \cdots <$

$x_{n+1} = b$ com $h = x_{i+1} - x_i$, $i = 1, 2, \ldots, n$. Então, aplicando a regra do trapézio em cada integração nos subintervalos, obtemos

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} \int_{x_i}^{x_{i+1}} f(x)\,dx \tag{10.51}$$

$$= \sum_{i=1}^{n} \left\{ \frac{h}{2}\left[f(x_i) + f(x_{i+1})\right] + O(h^3) \right\} \tag{10.52}$$

$$= f(x_1)\frac{h}{2} + \sum_{i=2}^{n} hf(x_i) + f(x_{n+1})\frac{h}{2} + O(h^2). \tag{10.53}$$

Desta forma, a regra composto do trapézio com n subintervalos é

$$\int_a^b f(x)\,dx = \frac{h}{2}\left[f(x_1) + \sum_{i=2}^{n} 2f(x_i) + f(x_{n+1})\right] + O(h^2), \tag{10.54}$$

onde $h = (b-a)/n$ e $x_i = a + (i-1)h$, $i = 1, 2, \ldots, n$.

Exemplo 10.2.2. Consideremos o problema de computar a integral de $f(x) = xe^{-x^2}$ no intervalo $[0, 1]$. Usando a regra composta do trapézio com n subintervalos, obtemos a aproximação

$$\underbrace{\int_a^b f(x)\,dx}_{I} \approx \underbrace{\frac{h}{2}\left[f(x_1) + 2\sum_{i=2}^{n} f(x_i) + f(x_{n+1})\right]}_{S}, \tag{10.55}$$

onde $h = 1/(4n)$ e $x_i = (i-1)h$, $i = 1, 2, \ldots, n$. Na Tabela 10.2, temos as aproximações computadas com diversos números de subintervalos, bem como, seus erros absolutos.

Tabela 10.2: Resultados referentes ao Exemplo 10.2.2.

n	S	$\lvert I - S\rvert$
1	1,83940e−1	1,3e−1
10	3,14919e−1	1,1e−3
100	3.16049e−1	1,1e−5
1000	3,16060e−1	1,1e−7

Podemos fazer estas computações com o auxílio do seguinte código `GNU Octave`:

```
f = @(x) x*exp(-x^2);
a=0;
b=1;
n=1000;
h=(b-a)/n;
s=f(a);
for i=2:n
  x=a+(i-1)*h;
  s+=2*f(x);
endfor
s+=f(b);
s*=h/2;
printf("%1.5E %1.1E\n",s,abs((1-e^(-1))/2-s))
```

10.2.3 Regra composta de Simpson

A fim de obtermos a regra composta de Simpson, consideramos uma partição uniforme do intervalo de integração $[a,b]$ da forma $a=\tilde{x}_1<\tilde{x}_2<\cdots<\tilde{x}_{n+1}=b$, com $h=(\tilde{x}_{i+1}-\tilde{x}_i)/2$, $i=1,2,\ldots,n$. Então, aplicando a regra de Simpson a cada integral nos subintervalos $[\tilde{x}_i,\tilde{x}_{i+1}]$, temos

$$\int_a^b f(x)\,dx=\sum_{i=1}^{n}\int_{\tilde{x}_i}^{\tilde{x}_{i+1}} f(x)\,dx \tag{10.56}$$

$$=\sum_{i=1}^{n}\left\{\frac{h}{3}\left[f(\tilde{x}_i)+4f\left(\frac{\tilde{x}_i+\tilde{x}_{i+1}}{2}\right)+f(\tilde{x_{i+1}})\right]+O(h^5)\right\}. \tag{10.57}$$

Então, observando que $h=(b-a)/(2n)$ e tomando $x_i=a+(i-1)h$, $i=1,2,\ldots,n$, obtemos a regra composta de Simpson com n subintervalos

$$\int_a^b f(x)\,dx=\frac{h}{3}\left[f(x_1)+2\sum_{i=2}^{n}f(x_{2i-1})+4\sum_{i=1}^{n}f(x_{2i})+f(x_{n+1})\right]$$
$$+O(h^4) \tag{10.58}$$

Exemplo 10.2.3. Consideremos o problema de computar a integral de $f(x)=xe^{-x^2}$ no intervalo $[0,1]$. Usando a regra composta de Simpson com n subin-

tervalos, obtemos a aproximação

$$\underbrace{\int_a^b f(x)\,dx}_{I} \approx \underbrace{\frac{h}{3}\left[f(x_1) + 2\sum_{i=2}^{n} f(x_{2i-1}) + 4\sum_{i=1}^{n} f(x_{2i}) + f(x_{n+1})\right]}_{S}, \quad (10.59)$$

onde $h = 1/(8n)$ e $x_i = (i-1)h$, $i = 1, 2, \ldots, n$. Na Tabela 10.3, temos as aproximações computadas com diversos números de subintervalos, bem como, seus erros absolutos.

Tabela 10.3: Resultados referentes ao Exemplo 10.2.3.

n	S	$\lvert I - S\rvert$
1	3,20914e−1	4,9e−3
10	3,16061e−1	3,4e−7
100	3,16060e−1	3,4e−11
1000	3,16060e−1	4,2e−15

Podemos fazer estas computações com o auxílio do seguinte código `GNU Octave`:

```
f = @(x) x*exp(-x^2);
a=0;
b=1;
n=1000;
h=(b-a)/(2*n);
s=f(a);
for i=2:n
  x=a+(2*i-2)*h;
  s+=2*f(x);
endfor
for i=1:n
  x=a+(2*i-1)*h;
  s+=4*f(x);
endfor
s+=f(b);
s*=h/3;
printf("%1.5E %1.1E\n",s,abs((1-e^(-1))/2-s))
```

Exercícios

Exercício 10.2.1. Aproxime

$$\int_{-1}^{0} \frac{\operatorname{sen}(x+2) - e^{-x^2}}{x^2 + \ln(x+2)}\, dx \tag{10.60}$$

usando a:

a) regra composta do ponto médio com 10 subintervalos.

b) regra composta do trapézio com 10 subintervalos.

c) regra composta de Simpson com 10 subintervalos.

Exercício 10.2.2. Considere a seguinte tabela de pontos

i	1	2	3	4	5	6
x_i	2,0	2,1	2,2	2,3	2,4	2,5
y_i	1,86	1,90	2,01	2,16	2,23	2,31

Assumindo que $y = f(x)$, e usando o máximo de subintervalos possíveis, calcule:

a) $\int_{2,0}^{2,4} f(x)\, dx$ usando a regra do ponto médio composta.

b) $\int_{2,0}^{2,5} f(x)\, dx$ usando a regra do trapézio composta.

c) $\int_{2,0}^{2,4} f(x)\, dx$ usando a regra de Simpson composta.

10.3 Quadratura de Romberg

A quadratura de Romberg é construída por sucessivas extrapolações de Richardson da regra do trapézio composta. Sejam $h_k = (b-a)/(2k)$, $x_i = a + (i-1)h_k$ e

$$R_{k,1} := \frac{h_k}{2}\left[f(a) + 2\sum_{i=2}^{2k} f(x_i) + f(b)\right] \tag{10.61}$$

a regra do trapézio composta com $2k$ subintervalos de

$$I := \int_a^b f(x)\, dx. \tag{10.62}$$

Por sorte, o erro de truncamento de aproximar I por $R_{k,1}$ tem a seguinte forma

$$I - R_{k,1} = \sum_{i=1}^{\infty} k_i h_k^{2i}, \tag{10.63}$$

o que nos permite aplicar a extrapolação de Richardson para obter aproximações de mais alta ordem.

Mais precisamente, para obtermos uma aproximação de I com erro de truncamento da ordem h^{2n}, $h = (b-a)$, computamos $R_{k,1}$ para $k = 1, 2, \ldots, n$. Então, usamos das sucessivas extrapolações de Richardson

$$R_{k,j} := R_{k,j-1} + \frac{R_{k,j-1} - R_{k-1,j-1}}{4^{j-1} - 1}, \tag{10.64}$$

$j = 2, 3, \ldots, n$, de forma a computarmos $R_{n,n}$, a qual fornece a aproximação desejada.

Exemplo 10.3.1. Consideremos o problema de aproximar a integral de $f(x) = xe^{-x^2}$ no intervalo $[0, 1]$. Para obtermos uma quadratura de Romberg de ordem 4, calculamos

$$R_{1,1} := \frac{1}{2}[f(0) + f(1)] = 1{,}83940\text{e}{-1} \tag{10.65}$$

$$R_{2,1} := \frac{1}{4}[f(0) + 2f(1/2) + f(1)] = 2{,}86670\text{e}{-1}. \tag{10.66}$$

Então, calculando

$$R_{2,2} = R_{2,1} + \frac{R_{2,1} - R_{1,1}}{3} = 3{,}20914\text{e}{-1}, \tag{10.67}$$

a qual é a aproximação desejada.

Tabela 10.4: Resultados referentes ao Exemplo 10.3.1.

k	$R_{k,1}$	$R_{k,2}$	$R_{k,3}$	$R_{k,4}$
1	1,83940e−1			
2	2,86670e−1	3,20914e−1		
3	3,08883e−1	3,16287e−1	3,15978e−1	
4	3,14276e−1	3,16074e−1	3,16059e−1	3,16061e−1

Na Tabela 10.4, temos os valores de aproximações computadas pela quadratura de Romberg até ordem 8.

Podemos fazer estas computações com o auxílio do seguinte código `GNU Octave`:

```
#integral
f = @(x) x*exp(-x^2);
a=0;
b=1;

#ordem 2n
n=4;

R = zeros(n,n);
#R(k,1)
for k=1:n
  h = (b-a)/(2^(k-1));
  R(k,1) = f(a);
  for i=2:2^(k-1)
    x = a + (i-1)*h;
    R(k,1) += 2*f(x);
  endfor
  R(k,1) += f(b);
  R(k,1) *= h/2;
endfor
#extrapola
for j=2:n
  for k=j:n
    R(k,j) = R(k,j-1) + (R(k,j-1)-R(k-1,j-1))/(4^(j-1)-1);
  endfor
endfor
#sol.
for i = 1:n
  printf("%1.5E ",R(i,:))
  printf("\n")
end
```

Exercícios

Exercício 10.3.1. Aproxime

$$\int_{-1}^{0} \frac{\operatorname{sen}(x+2) - e^{-x^2}}{x^2 + \ln(x+2)}\, dx \tag{10.68}$$

usando a quadratura de Romberg de ordem 4.

10.4 Grau de exatidão

O grau de exatidão é uma medida de exatidão de uma quadratura numérica. Mais precisamente, dizemos que uma dada quadratura numérica de nodos e pesos $\{(x_i, w_i)\}_{i=1}^{n}$ tem grau de exatidão m, quando

$$\int_a^b p(x)\, dx = \sum_{i=1}^{n} p(x_i) w_i \tag{10.69}$$

para todo polinômio $p(x)$ de grau menor m. Ou ainda, conforme descrito na definição a seguir.

Definição 10.4.1. Dizemos que uma dada quadratura numérica de pontos e nodos $\{x_i, w_i\}_{i=1}^{n}$ tem **grau de exatidão** m, quando

$$\int_a^b x^k\, dx = \sum_{i=1}^{n} x_i^k w_i,\ \forall k \leq m. \tag{10.70}$$

Exemplo 10.4.1. Determinemos o grau de exatidão da regra do ponto médio. Para tanto, verificamos para quais k vale

$$\int_a^b x^k\, dx = (b-a)\left(\frac{a+b}{2}\right)^k. \tag{10.71}$$

Vejamos:

- $k = 0$:

$$\int_a^b x^0\, dx = x|_a^b = b - a, \tag{10.72}$$

$$(b-a)\left(\frac{a+b}{2}\right)^0 = b - a. \tag{10.73}$$

- $k = 1$:

$$\int_a^b x^1\,dx = \frac{x^2}{2}\bigg|_a^b = \frac{b^2}{2} - \frac{a^2}{2}, \tag{10.74}$$

$$(b-a)\left(\frac{a+b}{2}\right)^1 = (b-a)\frac{(a+b)}{2} = \frac{b^2}{2} - \frac{a^2}{2}. \tag{10.75}$$

- $k = 2$:

$$\int_a^b x^2\,dx = \frac{x^3}{3}\bigg|_a^b = \frac{b^3}{3} - \frac{a^3}{3}, \tag{10.76}$$

$$(b-a)\left(\frac{a+b}{2}\right)^2 \neq \frac{b^3}{3} - \frac{a^3}{3}. \tag{10.77}$$

Ou seja, a regra do ponto média tem grau de exatidão 1.

Exemplo 10.4.2. Determinemos o grau de exatidão da regra de Simpson. Para tanto, verificamos para quais k vale

$$\int_a^b x^k\,dx = \frac{(b-a)}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right)^k. \tag{10.78}$$

Vejamos:

- $k = 0$:

$$\int_a^b x^0\,dx = x|_a^b = b - a, \tag{10.79}$$

$$\frac{(b-a)}{6}\left(a^0 + 4\left(\frac{a+b}{2}\right)^0 + b^0\right) = b - a. \tag{10.80}$$

- $k = 1$:

$$\int_a^b x^1\,dx = \frac{x^2}{2}\bigg|_a^b = \frac{b^2}{2} - \frac{a^2}{2}, \tag{10.81}$$

$$\frac{(b-a)}{6}\left(a^1 + 4\left(\frac{a+b}{2}\right)^1 + b^1\right) = \frac{(b-a)}{2}(a+b) \tag{10.82}$$

$$= \frac{b^2}{2} - \frac{a^2}{2}. \tag{10.83}$$

- $k = 2$:

$$\int_a^b x^2\,dx = \frac{x^3}{3}\bigg|_a^b = \frac{b^3}{3} - \frac{a^3}{3}, \tag{10.84}$$

$$\frac{(b-a)}{6}\left(a^2 + 4\left(\frac{a+b}{2}\right)^2 + b^2\right) = \frac{(b-a)}{3}(a^2 + ab + b^2) \tag{10.85}$$

$$= \frac{b^3}{3} - \frac{a^3}{3}. \tag{10.86}$$

- $k = 3$:

$$\int_a^b x^3\,dx = \frac{x^4}{4}\bigg|_a^b = \frac{b^4}{4} - \frac{a^4}{4}, \tag{10.87}$$

$$\frac{(b-a)}{6}\left(a^3 + 4\left(\frac{a+b}{2}\right)^3 + b^3\right) \tag{10.88}$$

$$= \frac{(b-a)}{6}\left[\frac{3a^3}{2} + \frac{3b}{2}a^2 + \frac{3a}{2}b^2 + \frac{3b^3}{2}\right] \tag{10.89}$$

$$= \frac{b^4}{4} - \frac{a^4}{4}. \tag{10.90}$$

- $k = 4$:

$$\int_a^b x^4\,dx = \frac{x^5}{5}\bigg|_a^b = \frac{b^5}{5} - \frac{a^5}{5}, \tag{10.91}$$

$$\frac{(b-a)}{6}\left(a^4 + 4\left(\frac{a+b}{2}\right)^4 + b^4\right) \neq \frac{b^5}{5} - \frac{a^5}{5}. \tag{10.92}$$

Ou seja, a regra de Simpson tem grau de exatidão 3.

Exercícios

Exercício 10.4.1. Determine o grau de exatidão da regra do trapézio.

Exercício 10.4.2. Determine o nodo e o peso da quadratura numérica de um único nodo e de grau de exatidão 1 para o intervalo de integração $[-1, 1]$.

10.5 Quadratura Gauss-Legendre

Quadraturas gaussianas são quadraturas numéricas de máximo grau de exatidão. Especificamente, quadraturas de Gauss-Legendre são quadraturas gaussianas para integrais da forma

$$\int_{-1}^{1} f(x)\,dx. \tag{10.93}$$

Consideremos o problema de determinar a quadratura de Gauss-Legendre de apenas um ponto. Começamos por exigir o grau de exatidão 0, o que nos leva a

$$w_1 x_1^0 = \int_{-1}^{1} x^0\,dx \Rightarrow w_1 = x|_{-1}^{1} = 2. \tag{10.94}$$

Agora, exigindo o grau de exatidão 1, obtemos

$$w_1 x_1^1 = \int_{-1}^{1} x^1\,dx \Rightarrow 2x_1 = \left.\frac{x^2}{2}\right|_{-1}^{1} = 0 \tag{10.95}$$

$$\Rightarrow x_1 = 0. \tag{10.96}$$

Com isso, concluímos que a quadratura de apenas um nodo de maior grau de exatidão para tais integrais é a de nodo $x_1 = 0$ e $w_1 = 2$. A qual é, por acaso, a regra do ponto médio.

Observamos, também, que cada grau de exatidão nos fornece uma condição para determinarmos os nodos e os pesos da desejada quadratura. Mais precisamente, seguindo o raciocínio anterior, para determinarmos a quadratura de n pontos com maior grau de exatidão possível para integrais no intervalo $[-1, 1]$, acabaremos tendo que resolver um sistema de equações

$$\sum_{i=1}^{n} x_i^k w_i = \int_{-1}^{1} x^k\,dx,\ k = 0,1,2,\ldots,2n-1. \tag{10.97}$$

Isto é, como teremos $2n$ incógnitas (n nodos e n pesos) a determinar, poderemos exigir o grau de exatidão máximo de $2n-1$.

O sistema (10.97) é um sistema não linear para os nodos e a determinação de soluções para n grande não é uma tarefa trivial. Alternativamente, veremos que os pontos da quadratura de Gauss-Legendre de n nodos são as

raízes do polinômio de Legendre de grau n. Por definição, o polinômio de Legendre de grau n, denotado por $P_n(x)$, satisfaz a seguinte propriedade de ortogonalidade

$$\int_{-1}^{1} p(x)P_n(x)\,dx = 0, \tag{10.98}$$

para todo polinômio $p(x)$ de grau menor que n. Com isso, estabelecemos o seguinte resultado.

Teorema 10.5.1. *A quadratura de Gauss-Legendre de n nodos tem as raízes do polinômio de Legendre de grau n como seus nodos e seus pesos são dados por*

$$w_i = \int_{-1}^{1} \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}\,dx. \tag{10.99}$$

Demonstração. Sejam $x_1, x_2, \ldots, x_n$ as raízes do polinômio de Legendre de grau n. Queremos mostrar que

$$\int_{-1}^{1} p(x)\,dx = \sum_{i=1}^{n} p(x_i)w_i, \tag{10.100}$$

para todo polinômio $p(x)$ de grau menor ou igual $2n-1$. Primeiramente, suponhamos que $p(x)$ seja um polinômio de grau menor que n. Então, tomando sua representação por polinômio de Lagrange nos nodos x_i, $i = 1, 2, \ldots, n$, temos

$$\int_{-1}^{1} p(x)\,dx = \int_{-1}^{1} \sum_{i=1}^{n} p(x_i) \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}\,dx \tag{10.101}$$

$$= \sum_{i=1}^{n} p(x_i) \int_{-1}^{1} \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x - x_j}{x_i - x_j}\,dx \tag{10.102}$$

$$= \sum_{i=1}^{n} p(x_i)w_i. \tag{10.103}$$

Isto mostra o resultado para polinômios $p(x)$ de grau menor que n. Agora, suponhamos que $p(x)$ é um polinômio de grau maior ou igual que n e menor

ou igual a $2n-1$. Dividindo $p(x)$ pelo polinômio de Legendre de grau n, $P_n(x)$, obtemos

$$p(x) = q(x)P_n(x) + r(x), \tag{10.104}$$

onde $q(x)$ e $r(x)$ são polinômio de grau menor que n. Ainda, nas raízes $x_1, x_2, \ldots, x_n$ temos $p(x_i) = r(x_i)$ e da ortogonalidade dos polinômios de Legendre (veja, equação (10.98)), temos

$$\int_{-1}^{1} p(x)\,dx = \int_{-1}^{1} q(x)P_n(x) + r(x)\,dx \tag{10.105}$$

$$= \int_{-1}^{1} r(x)\,dx. \tag{10.106}$$

Agora, do resultado anterior aplicado a $r(x)$, temos

$$\int_{-1}^{1} p(x)\,dx = \sum_{i=1}^{n} r(x_i)w_i = \sum_{i=1}^{n} p(x_i)w_i. \tag{10.107}$$

Isto complete o resultado para polinômios de grau menor ou igual a $2n-1$. □

Exemplo 10.5.1. Consideremos a quadratura de Gauss-Legendre de 2 nodos. Do teorema anterior (Teorema 10.5.1, seus nodos são as raízes do polinômio de Legendre de grau 2

$$P_2(x) = \frac{3}{2}x^2 - \frac{1}{2}, \tag{10.108}$$

as quais são

$$x_1 = -\frac{\sqrt{3}}{3}, \quad x_2 = \frac{\sqrt{3}}{3}. \tag{10.109}$$

Os pesos são, então

$$w_1 = \int_{-1}^{1} \frac{x - x_1}{x_2 - x_1}\,dx \tag{10.110}$$

$$= \frac{\sqrt{3}}{2}\left[\frac{x^2}{2} + \frac{\sqrt{3}}{3}x\right]_{-1}^{1} \tag{10.111}$$

$$= 1 \tag{10.112}$$

e

$$w_2 = \int_{-1}^{1} \frac{x - x_2}{x_1 - x_2}\, dx \tag{10.113}$$

$$= -\frac{\sqrt{3}}{2}\left[\frac{x^2}{2} - \frac{\sqrt{3}}{3}x\right]_{-1}^{1} \tag{10.114}$$

$$= 1 \tag{10.115}$$

Ou seja, a quadratura de Gauss-Legendre de 2 pontos tem o seguinte conjunto de nodos e pesos $\{(x_1 = -\sqrt{3}/3, w_1 = 1), (x_2 = \sqrt{3}/3, w_2 = 1)\}$. Esta, por sua vez, é exata para polinômios de grau menor ou igual a 3. De fato, verificando para potência de x^k temos:

- $k = 0$:

$$\int_{-1}^{1} x^0\, dx = 2 \tag{10.116}$$

$$x_1^0 w_1 + x_2^0 w_2 = \left(-\frac{\sqrt{3}}{3}\right)^0 + \left(\frac{\sqrt{3}}{3}\right)^0 = 2. \tag{10.117}$$

- $k = 1$:

$$\int_{-1}^{1} x^1\, dx = 0 \tag{10.118}$$

$$x_1^1 w_1 + x_2^1 w_2 = \left(-\frac{\sqrt{3}}{3}\right)^1 + \left(\frac{\sqrt{3}}{3}\right)^1 = 0. \tag{10.119}$$

- $k = 2$:

$$\int_{-1}^{1} x^2\, dx = \frac{2}{3} \tag{10.120}$$

$$x_1^2 w_1 + x_2^2 w_2 = \left(-\frac{\sqrt{3}}{3}\right)^2 + \left(\frac{\sqrt{3}}{3}\right)^2 = \frac{2}{3}. \tag{10.121}$$

- $k = 3$:

$$\int_{-1}^{1} x^3\, dx = 0 \tag{10.122}$$

$$x_1^3 w_1 + x_2^3 w_2 = \left(-\frac{\sqrt{3}}{3}\right)^3 + \left(\frac{\sqrt{3}}{3}\right)^3 = 0. \tag{10.123}$$

- $k = 4$:

$$\int_{-1}^{1} x^4\,dx = \frac{2}{5} \tag{10.124}$$

$$x_1^4 w_1 + x_2^4 w_2 = \left(-\frac{\sqrt{3}}{3}\right)^4 + \left(\frac{\sqrt{3}}{3}\right)^4 = \frac{2}{9}. \tag{10.125}$$

Tabela 10.5: Conjunto de nodos e pesos da quadratura de Gauss-Legendre.

n	x_i	w_i
1	0	2
2	$\pm\frac{\sqrt{3}}{3}$	1
3	0	$\frac{8}{9}$
	$\pm\sqrt{\frac{3}{5}}$	$\frac{5}{9}$
4	$\pm\sqrt{\frac{3}{7} - \frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18+\sqrt{30}}{36}$
	$\pm\sqrt{\frac{3}{7} + \frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18-\sqrt{30}}{36}$
5	0	$\frac{128}{225}$
	$\pm\frac{1}{3}\sqrt{5 - 2\sqrt{\frac{10}{7}}}$	$\frac{322+13\sqrt{70}}{900}$
	$\pm\frac{1}{3}\sqrt{5 + 2\sqrt{\frac{10}{7}}}$	$\frac{322-13\sqrt{70}}{900}$

Observação 10.5.1. O conjunto de nodos e pesos da quadratura de Gauss-Legendre para $n = 1, 2, 3, 4, 5$ são apresentados na Tabela 10.5[2]. Alternativamente, a quadratura de Gauss-Legendre com n pontos tem seus nodos iguais

[2]Disponível em https://en.wikipedia.org/w/index.php?title=Gaussian_quadrature&oldid=837460315.

as raízes de $P_n(x)$ (o polinômio de Legendre de gaus n), e os pesos dados por (10.99) ou [5, Cap.4, Sec. 4.6]:

$$w_i = \frac{2}{(1-x_i^2)\left[P_n'(x_i)\right]^2}, \quad i = 1, 2, \ldots, n. \tag{10.126}$$

Assim sendo, no `GNU Octave` podemos encontrar os nodos e pesos da quadratura de Gauss-Legendre com o seguinte código:

```
pkg load miscellaneous
n=6;
Pn=legendrepoly(n);
dPn=polyder(Pn);
x=roots(Pn);
w=2./((1-x.^2).*(polyval(dPn,x)).^2);
printf("i xx_i w_i\n")
for i=1:n
  printf("%d %1.7E %1.7E\n",i,x(i),w(i))
endfor
```

Exemplo 10.5.2. Considere o problema de obter uma aproximação para $I = \int_{-1}^{1} \cos(x)\,dx$ usando a quadratura de Gauss-Legendre. Calculemos algumas aproximações com $n = 1$, 2 e 3 pontos:

- $n = 1$:

$$\int_{-1}^{1} \cos(x)\,dx \approx 2\cos 0 = 2. \tag{10.127}$$

- $n = 2$:

$$\int_{-1}^{1} xe^{-x^2}\,dx \approx \cos(-\sqrt{3}/3) + \cos(-\sqrt{3}/3) = 1{,}67582. \tag{10.128}$$

- $n = 3$:

$$\begin{aligned}\int_{-1}^{1} xe^{-x^2}\,dx \approx{}& \frac{8}{9}\cos 0 + \frac{5}{9}\cos(-\sqrt{3/5}) \\ &+ \frac{5}{9}\cos(\sqrt{3/5}) = 1{,}68300.\end{aligned} \tag{10.129}$$

n	$\tilde{I}$	$\lvert I-\tilde{I}\rvert$
1	2,00000	3,2e−01
2	1,67582	7,1e−03
3	1,68300	6,2e−05
4	1,68294	2,8e−07
5	1,68294	7,9e−10

Tabela 10.6: Resultados referentes ao Exemplo 10.5.2.

Na Tabela 10.6, temos as aproximações de I com a quadratura de Gauss-Legendre de $n = 1$, 2, 3, 4 e 5 pontos (detonado por $\tilde{I}$, bem como, o erro absoluto com respeito ao valor analítico da integral.

Os cálculos, aqui, apresentados podem ser realizados no `GNU Octave` com o seguinte código:

```
f = @(x) cos(x);

#int. anlitc.
ia = sin(1)-sin(-1);

#GL-1
x=0;
w=2;
s=w*f(x);
printf("%1.5E %1.1\E\n",s,abs(s-ia))

#GL-2
x=[sqrt(3)/3];
w=[1];
s=w(1)*f(x(1));
s+=w(1)*f(-x(1));
printf("%1.5E %1.1\E\n",s,abs(s-ia))

#GL-3
x=[0 sqrt(3/5)];
w=[8/9 5/9];
s=w(1)*f(x(1)) + w(2)*f(x(2));
```

```
s+=w(2)*f(-x(2));
printf("%1.5E %1.1\E\n",s,abs(s-ia))

#GL-4
x=[sqrt(3/7-2/7*sqrt(6/5)) sqrt(3/7+2/7*sqrt(6/5))];
w=[(18+sqrt(30))/36 (18-sqrt(30))/36];
s=w(1)*f(x(1)) + w(2)*f(x(2));
s+=w(1)*f(-x(1)) + w(2)*f(-x(2));
printf("%1.5E %1.1\E\n",s,abs(s-ia))

#GL-5
x=[0 1/3*sqrt(5-2*sqrt(10/7)) 1/3*sqrt(5+2*sqrt(10/7))];
w=[128/225 (322+13*sqrt(70))/900 (322-13*sqrt(70))/900];
s=w(1)*f(x(1)) + w(2)*f(x(2)) + w(3)*f(x(3));
s+=w(2)*f(-x(2)) + w(3)*f(-x(3));
printf("%1.5E %1.1\E\n",s,abs(s-ia))
```

10.5.1 Intervalos de integração arbitrários

Observamos que a quadratura de Gauss-Legendre foi desenvolvida para aproximar integrais definidas no intervalo $[-1, 1]$. Por sorte, uma integral definida em um intervalo arbitrário $[a, b]$ pode ser reescrita como uma integral no intervalo $[-1, 1]$ através de uma mudança de variável apropriada. Mais precisamente, assumindo a mudança de variável

$$x = \frac{b-a}{2}(u+1) + a \tag{10.130}$$

temos

$$dx = \frac{b-a}{2}du \tag{10.131}$$

e, portanto,

$$\int_a^b f(x)\,dx = \int_{-1}^{1} f\left(\frac{b-a}{2}(u+1)+a\right)\cdot\frac{b-a}{2}du. \tag{10.132}$$

Portanto, para computarmos $\int_a^b f(x)\,dx$ podemos aplicar a quadratura de Gauss-Legendre na integral definida no $[-1, 1]$ dada conforme acima.

Exemplo 10.5.3. Usemos a quadratura de Gauss-Legendre com 2 pontos para aproximar a integral

$$\int_0^1 xe^{-x^2}\,dx. \tag{10.133}$$

Fazendo a mudança de variável $x = u/2 + 1/2$, temos

$$\int_0^1 xe^{-x^2}\,dx = \int_{-1}^1 \left(\frac{u}{2}+\frac{1}{2}\right) e^{-\left(\frac{u}{2}+\frac{1}{2}\right)^2}\,du. \tag{10.134}$$

Então, aplicando a quadratura temos

$$\int_0^1 xe^{-x^2}\,dx = \left(-\frac{\sqrt{3}}{6}+\frac{1}{2}\right) e^{-\left(-\frac{\sqrt{3}}{6}+\frac{1}{2}\right)^2} + \left(\frac{\sqrt{3}}{6}+\frac{1}{2}\right) e^{-\left(\frac{\sqrt{3}}{6}+\frac{1}{2}\right)^2} \tag{10.135}$$

$$= 3{,}12754\text{e}{-1}. \tag{10.136}$$

No `GNU Octave`, pode usar o seguinte código:

```
f = @(x) x*exp(-x^2);
a=0;
b=1;
F = @(u) (b-a)/2*f((b-a)/2*(u+1)+a);
x=sqrt(3)/3;
w=1;
s=w*F(-x)+w*F(x);
printf("%1.5E\n",s)
```

Exercícios

Exercício 10.5.1. Aproxime

$$\int_{-1}^1 \frac{\operatorname{sen}(x+2) - e^{-x^2}}{x^2 + \ln(x+2)}\,dx \tag{10.137}$$

usando a quadratura de Gauss-Legendre com:

a) $n = 1$ ponto.

b) $n = 2$ pontos.

c) $n = 3$ pontos.

d) $n = 4$ pontos.

e) $n = 5$ pontos.

Exercício 10.5.2. Aproxime

$$\int_0^1 \frac{\operatorname{sen}(x+2) - e^{-x^2}}{x^2 + \ln(x+2)}\, dx \tag{10.138}$$

usando a quadratura de Gauss-Legendre com:

a) $n = 1$ ponto.

b) $n = 2$ pontos.

c) $n = 3$ pontos.

d) $n = 4$ pontos.

e) $n = 5$ pontos.

Exercício 10.5.3. Aproxime

$$\int_{-1}^1 \frac{\operatorname{sen}(x+2) - e^{-x^2}}{x^2 + \ln(x+2)}\, dx \tag{10.139}$$

usando a quadratura de Gauss-Legendre com:

a) $n = 5$ ponto.

b) $n = 10$ pontos.

c) $n = 20$ pontos.

10.6 Quadraturas gaussianas com pesos

A quadratura gaussiana estudada na seção anterior (Seção 10.5) é um caso particular de quadraturas de máximo grau de exatidão para integrais da forma

$$\int_a^b f(x)w(x)\, dx, \tag{10.140}$$

onde $w(x)$ é positiva e contínua, chamada de função peso. Como anteriormente, os nodos x_i, $i = 1, 2, \ldots, n$, da quadratura gaussiana de n pontos são as raízes do polinômio $p_n(x)$ que é ortogonal a todos os polinômios de grau menor que n. Aqui, isto significa

$$\int_a^b q(x)p_n(x)w(x)\,dx = 0, \tag{10.141}$$

para todo polinômio $q(x)$ de grau menor que n.

10.6.1 Quadratura de Gauss-Chebyshev

Quadraturas de Gauss-Chebyshev são quadraturas gaussianas para integrais da forma

$$\int_{-1}^1 f(x)(1-x^2)^{-1/2}\,dx. \tag{10.142}$$

Neste caso, na quadratura gaussiana de n pontos os nodos x_i são as raízes do n-ésimo polinômio de Chebyshev $T_n(x)$. Pode-se mostrar (veja, por exemplo, [3, Cap. 7, Sec. 4.1]) que o conjunto de pontos desta quadratura são dados por

$$x_i = \cos\left(\frac{2i-1}{2n}\pi\right), \tag{10.143}$$

$$w_i = \frac{\pi}{n}. \tag{10.144}$$

Exemplo 10.6.1. Considere o problema de aproximar a integral

$$\int_{-1}^1 \frac{e^{-x^2}}{\sqrt{1-x^2}}\,dx. \tag{10.145}$$

Usando a quadratura de Gauss-Chebyshev de n pontos temos:

- $n = 1$:

$$\int_{-1}^1 \frac{e^{-x^2}}{\sqrt{1-x^2}}\,dx \approx \pi e^{-\cos(\pi/2)^2} = \pi. \tag{10.146}$$

- $n = 2$:

$$\int_{-1}^1 \frac{e^{-x^2}}{\sqrt{1-x^2}}\,dx \approx \frac{\pi}{2}e^{-\cos(\pi/4)^2} + \frac{\pi}{2}e^{-\cos(3\pi/4)^2} \tag{10.147}$$

$$= 1{,}90547. \tag{10.148}$$

- $n = 3$:

$$\int_{-1}^{1} \frac{e^{-x^2}}{\sqrt{1-x^2}}\,dx \approx \frac{\pi}{3}e^{-\cos(\pi/6)^2} + \frac{\pi}{3}e^{-\cos(\pi/2)^2} + \frac{\pi}{3}e^{-\cos(5\pi/6)^2} \tag{10.149}$$

$$= 2{,}03652. \tag{10.150}$$

n	$\tilde{I}$
1	3,14159
2	1,90547
3	2,03652
4	2,02581
5	2,02647
6	2,02644
10	2,02644

Tabela 10.7: Resultados referentes ao Exemplo 10.6.1.

Na Tabela 10.7, temos as aproximações $\tilde{I}$ da integral computadas com a quadratura de Gauss-Chebyshev com diferentes números de pontos. Estes resultados podem ser computados no `GNU Octave` com o seguinte código:

```
f = @(x) exp(-x^2);
n=5;
s=0;
for i=1:n
  x=cos((2*i-1)*pi/(2*n));
  w=pi/n;
  s+=f(x)*w;
endfor
printf("%1.5E\n",s)
```

10.6.2 Quadratura de Gauss-Laguerre

Quadraturas de Gauss-Laguerre são quadraturas gaussianas para integrais da forma

$$\int_0^{\infty} f(x)e^{-x}\,dx. \tag{10.151}$$

Neste caso, na quadratura gaussiana de n pontos os nodos x_i são as raízes do n-ésimo polinômio de Laguerre $L_n(x)$ e os pesos por

$$w_i = -\frac{1}{n[L_n'(x_i)]^2},\ i = 1, 2, \ldots, n. \tag{10.152}$$

Na Tabela 10.8, temos os pontos da quadratura de Gauss-Laguerre para diversos valores de n.

Tabela 10.8: Pontos da quadratura de Gauss-Laguerre.

n	x_i	w_i
1	1,0000000e+00	1,0000000e+00
2	3,4142136e+00	1,4644661e−01
	5,8578644e−01	8,5355339e−01
3	6,2899451e+00	1,0389257e−02
	2,2942804e+00	2,7851773e−01
	4,1577456e−01	7,1109301e−01
4	9,3950709e+00	5,3929471e−04
	4,5366203e+00	3,8887909e−02
	1,7457611e+00	3,5741869e−01
	3,2254769e−01	6,0315410e−01
5	1,2640801e+01	2,3369972e−05
	7,0858100e+00	3,6117587e−03
	3,5964258e+00	7,5942450e−02
	1,4134031e+00	3,9866681e−01
	2,6356032e−01	5,2175561e−01

No `GNU Octave`, os pontos da quadratura de Gauss-Laguerre podem ser obtido com o seguinte código:

```
pkg load miscellaneous
n=2;
Ln=laguerrepoly(n);
dLn=polyder(Ln);
x=roots(Ln);
w=1./(x.*(polyval(dLn,x)).^2);
printf("i xx_i w_i\n")
for i=1:n
```

```
  printf("%1.7E %1.7E\n",x(i),w(i))
endfor
```

Exemplo 10.6.2. Na Tabela 10.9, temos as aproximações $\tilde{I}$ da integral $I = \int_0^\infty \text{sen}(x)e^{-x}\,dx$ obtidas pela quadratura de Gauss-Laguerre com diferentes pontos n.

n	$\tilde{I}$
1	8,41471e−01
2	4,32459e−01
3	4,96030e−01
4	5,04879e−01
5	4,98903e−01

Tabela 10.9: Resultados referentes ao Exemplo 10.6.1.

Os resultados obtidos neste exemplo podem ser computados no `GNU Octave` com o seguinte código:

```
f = @(x) sin(x);

n=1;
xw = [1.0000000E+00 1.0000000E+00];
s = f(xw(1,1))*xw(1,2);
printf("%d %1.5E\n",n,s)

n=2;
xw = [3.4142136E+00 1.4644661E-01; ...
      5.8578644E-01 8.5355339E-01];
s=0;
for i=1:n
  s+=f(xw(i,1))*xw(i,2);
endfor
printf("%d %1.5E\n",n,s)

n=3;
xw = [6.2899451E+00 1.0389257E-02;...
      2.2942804E+00 2.7851773E-01;...
      4.1577456E-01 7.1109301E-01];
```

```
s=0;
for i=1:n
  s+=f(xw(i,1))*xw(i,2);
endfor
printf("%d %1.5E\n",n,s)

n=4;
xw = [9.3950709E+00 5.3929471E-04;...
      4.5366203E+00 3.8887909E-02;...
      1.7457611E+00 3.5741869E-01;...
      3.2254769E-01 6.0315410E-01];
s=0;
for i=1:n
  s+=f(xw(i,1))*xw(i,2);
endfor
printf("%d %1.5E\n",n,s)

n=5;
xw = [1.2640801E+01 2.3369972E-05;...
      7.0858100E+00 3.6117587E-03;...
      3.5964258E+00 7.5942450E-02;...
      1.4134031E+00 3.9866681E-01;...
      2.6356032E-01 5.2175561E-01];
s=0;
for i=1:n
  s+=f(xw(i,1))*xw(i,2);
endfor
printf("%d %1.5E\n",n,s)
```

10.6.3 Quadratura de Gauss-Hermite

Quadraturas de Gauss-Hermite são quadraturas gaussianas para integrais da forma

$$\int_{-\infty}^{\infty} f(x)e^{-x^2}\,dx. \tag{10.153}$$

Seus nodos x_i, $i = 1, 2, \ldots, n$ são as raízes do n-ésimo polinômio de Hermite e os pesos são dados por

$$w_i = \frac{2^{n+1} n! \sqrt{\pi}}{[H'_n(x_i)]^2}. \tag{10.154}$$

Na Tabela 10.10, temos os pontos da quadratura de Gauss-Hermite para diversos valores de n.

Tabela 10.10: Pontos da quadratura de Gauss-Hermite.

n	x_i	w_i
1	0,0000000e+00	1,7724539e+00
2	−7,0710678e−01	8,8622693e−01
	7,0710678e−01	8,8622693e−01
3	−1,2247449e+00	2,9540898e−01
	1,2247449e+00	2,9540898e−01
	0,0000000e+00	1,1816359e+00
4	−1,6506801e+00	8,1312835e−02
	1,6506801e+00	8,1312835e−02
	−5,2464762e−01	8,0491409e−01
	5,2464762e−01	8,0491409e−01
5	−2,0201829e+00	1,9953242e−02
	2,0201829e+00	1,9953242e−02
	−9,5857246e−01	3,9361932e−01
	9,5857246e−01	3,9361932e−01
	0,0000000e+00	9,4530872e−01

No `GNU Octave`, os pontos da quadratura de Gauss-Hermite podem ser obtido com o seguinte código:

```
pkg load miscellaneous
n=2;
Hn=hermitepoly(n);
dHn=polyder(Hn);
x=roots(Hn);
w=2^(n+1)*factorial(n)*sqrt(pi)./((polyval(dHn,x)).^2);
printf("i xx_i w_i\n")
for i=1:n
```

```
  printf("%1.7E %1.7E\n",x(i),w(i))
endfor
```

Exemplo 10.6.3. Na Tabela 10.11, temos as aproximações $\tilde{I}$ da integral $I = \int_{-\infty}^{\infty} x\,\mathrm{sen}(x)e^{-x^2}\,dx$ obtidas pela quadratura de Gauss-Hermite com diferentes pontos n.

n	$\tilde{I}$
1	0,00000e+00
2	8,14199e−01
3	6,80706e−01
4	6,90650e−01
5	6,90178e−01

Tabela 10.11: Resultados referentes ao Exemplo 10.6.3.

Os resultados obtidos neste exemplo podem ser computados no `GNU Octave` com o seguinte código:

```
f = @(x) x*sin(x);

n=1;
xw = [0.0000000E+00 1.7724539E+00];
s = f(xw(1,1))*xw(1,2);
printf("%d %1.5E\n",n,s)

n=2;
xw = [-7.0710678E-01 8.8622693E-01;...
       7.0710678E-01 8.8622693E-01];
s=0;
for i=1:n
  s+=f(xw(i,1))*xw(i,2);
endfor
printf("%d %1.5E\n",n,s)

n=3;
xw = [-1.2247449E+00 2.9540898E-01;...
       1.2247449E+00 2.9540898E-01;...
       0.0000000E+00 1.1816359E+00];
```

```
s=0;
for i=1:n
  s+=f(xw(i,1))*xw(i,2);
endfor
printf("%d %1.5E\n",n,s)

n=4;
xw = [-1.6506801E+00 8.1312835E-02;...
       1.6506801E+00 8.1312835E-02;...
      -5.2464762E-01 8.0491409E-01;...
       5.2464762E-01 8.0491409E-01];
s=0;
for i=1:n
  s+=f(xw(i,1))*xw(i,2);
endfor
printf("%d %1.5E\n",n,s)

n=5;
xw = [-2.0201829E+00 1.9953242E-02;...
       2.0201829E+00 1.9953242E-02;...
      -9.5857246E-01 3.9361932E-01;...
       9.5857246E-01 3.9361932E-01;...
       0.0000000E+00 9.4530872E-01];
s=0;
for i=1:n
  s+=f(xw(i,1))*xw(i,2);
endfor
printf("%d %1.5E\n",n,s)
```

Exercícios

Exercício 10.6.1. Aproxime

$$\int_{-1}^{1} \frac{\operatorname{sen}(x+2) - e^{-x^2}}{\sqrt{1-x^2}}\, dx \tag{10.155}$$

usando a quadratura de Gauss-Chebyshev com:

a) $n = 1$ ponto.

b) $n = 2$ pontos.

c) $n = 3$ pontos.

d) $n = 4$ pontos.

e) $n = 5$ pontos.

Exercício 10.6.2. Aproxime

$$\int_0^\infty \left(\text{sen}(x+2) - e^{-x^2}\right) e^{-x}\, dx \tag{10.156}$$

usando a quadratura de Gauss-Laguerre com:

a) $n = 3$ pontos.

b) $n = 4$ pontos.

c) $n = 5$ pontos.

Exercício 10.6.3. Aproxime

$$\int_{-\infty}^\infty \text{sen}(x+2)e^{-x^2} - e^{-2x^2}\, dx \tag{10.157}$$

usando a quadratura de Gauss-Hermite com:

a) $n = 3$ pontos.

b) $n = 4$ pontos.

c) $n = 5$ pontos.

10.7 Método de Monte Carlo

O método de Monte Carlo é uma técnica não determinística para a aproximação de integrais. Mais especificamente, o método compreende a aproximação

$$\int_a^b f(x)\, dx \approx \frac{(b-a)}{n} \sum_{i=1}^{n} f(x_i), \tag{10.158}$$

onde $x_1, x_2, \ldots, x_n$ são pontos de uma sequência aleatória em $[a, b]$. Aqui, não vamos entrar em detalhes sobre a escolha desta sequência e, sem mais justificativas, assumiremos uma sequência de pontos uniformemente distribuídos no intervalo de integração.

Exemplo 10.7.1. Na tabela 10.12 temos aproximações $\tilde{I}$ computadas para

$$I = \int_0^1 xe^{-x^2}\,dx \tag{10.159}$$

usando o método de Monte Carlo com diferentes números de pontos n. Aqui, os pontos foram gerados no `GNU Octave` pela sequência *quasi*-randômica obtida da função `rand` inicializada com `seed=0`.

n	$\tilde{I}$	$\lvert I - \tilde{I}\rvert$
10	2,53304e−01	6,3e−02
100	3,03149e−01	1,3e−02
1000	3,08415e−01	7,6e−03
10000	3,16385e−01	3,2e−04
100000	3,15564e−01	5,0e−04

Tabela 10.12: Resultados referentes ao Exemplo 10.7.1.

Os resultados presentes na Tabela 10.12 podem ser computados no `GNU Octave` com o seguinte código:

```
#inic. gerador randômico
rand("seed",0)
#fun. obj.
f = @(x) x*exp(-x^2);
a=0;
b=1;
#num. de amostras
n=100000;
#calc. aprox.
s=0;
for i=1:n
  x=a + (b-a)*rand();
  s+=f(x);
endfor
```

```
s*=(b-a)/n;
#sol. analítica
ia=0.5-exp(-1)/2;
printf("%1.5E %1.1E\n",s,abs(ia-s))
```

Exercícios

Exercício 10.7.1. Use o método de Monte Carlo para obter uma aproximação de

$$\int_{-1}^{1} \frac{\operatorname{sen}(x+2) - e^{-x^2}}{x^2 + \ln(x+2)}\, dx \tag{10.160}$$

com precisão de 10^{-2}.

Capítulo 11

Problema de valor inicial

Neste capítulo, discutimos sobre técnicas numéricas para aproximar a solução de equações diferenciais ordinárias com valor inicial, i.e. problemas da forma

$$y'(t) = f(t,y(t)), \quad t > t_0, \tag{11.1}$$
$$y(t_0) = y_0. \tag{11.2}$$

11.1 Método de Euler

Dado um problema de valor inicial

$$y'(t) = f(t,y(t)), \quad t > t_0, \tag{11.3}$$
$$y(t_0) = y_0, \tag{11.4}$$

temos que $f(t,y)$ é a derivada da solução $y(t)$ no tempo t. Então, aproximando a derivada pela razão fundamental de passo $h > 0$, obtemos

$$\frac{y(t+h) - y(t)}{h} \approx f(t,y) \tag{11.5}$$
$$\Rightarrow y(t+h) \approx y(t) + hf(t,y(t)). \tag{11.6}$$

Ou seja, se conhecermos a solução y no tempo t, então (11.6) nos fornece uma aproximação da solução y no tempo $t+h$. Observemos que isto poder ser usado de forma iterativa. Da condição inicial $y(t_0) = y_0$, computamos

uma aproximação de y no tempo $t_0 + h$. Usando esta no lugar de $y(t_0 + h)$, (11.6) com $t_0 + h$ no lugar de t nos fornece uma aproximação para $y(t_0 + 2h)$ e, assim, sucessivamente.

Mais especificamente, denotando $y^{(i)}$ a aproximação de $y(t^{(i)})$ com $t^{(i)} = t_0 + (i-1)h$, $i = 1, 2, \ldots, n$, o **método de Euler** consiste na iteração

$$y^{(1)} = y_0, \tag{11.7}$$
$$y^{(i+1)} = y^{(i)} + hf(t^{(i)}, y^{(i)}), \tag{11.8}$$

com $i = 1, 2, \ldots, n$. O número de iteradas n e o tamanho do passo $h > 0$, determinam os tempos discretos $t^{(i)}$ nos quais a solução y será aproximada.

Exemplo 11.1.1. Consideremos o seguinte problema de valor inicial

$$y' - y = \text{sen}(t), t > 0 \tag{11.9}$$
$$y(0) = \frac{1}{2}. \tag{11.10}$$

A solução analítica deste é

$$y(t) = e^t - \frac{1}{2}\text{sen}(t) - \frac{1}{2}\cos(t). \tag{11.11}$$

No tempo, $t_f = 1$, temos $y(t_f) = e^{t_f} - \text{sen}(1)/2 - \cos(1)/2 = 2{,}02740$. Agora, computarmos uma aproximação para este problema pelo método de Euler, reescrevemos (11.35) na forma

$$y' = y + \text{sen}(t) =: f(t, y). \tag{11.12}$$

Então, escolhendo $h = 0{,}1$, a iteração do método de Euler (11.7)-(11.8) nos fornece o método de Euler com passo $h = 0{,}1$

$$y^{(1)} = 0{,}5 \tag{11.13}$$
$$\begin{aligned} y^{(2)} &= y^{(1)} + hf(t^{(1)}, y^{(1)}) \\ &= 0{,}5 + 0{,}1[0{,}5 + \text{sen}(0)] \\ &= 0{,}55 \end{aligned} \tag{11.14}$$
$$\begin{aligned} y^{(3)} &= y^{(2)} + hf(t^{(2)}, y^{(2)}) \\ &= 0{,}55 + 0{,}1[0{,}55 + \text{sen}(0{,}1)] \\ &= 6{,}14983\text{e}{-}01 \end{aligned} \tag{11.15}$$

$$\vdots$$

$$y^{(11)} = 1{,}85259 \tag{11.16}$$

Na Figura 11.1, temos os esboços das soluções analítica e numérica.

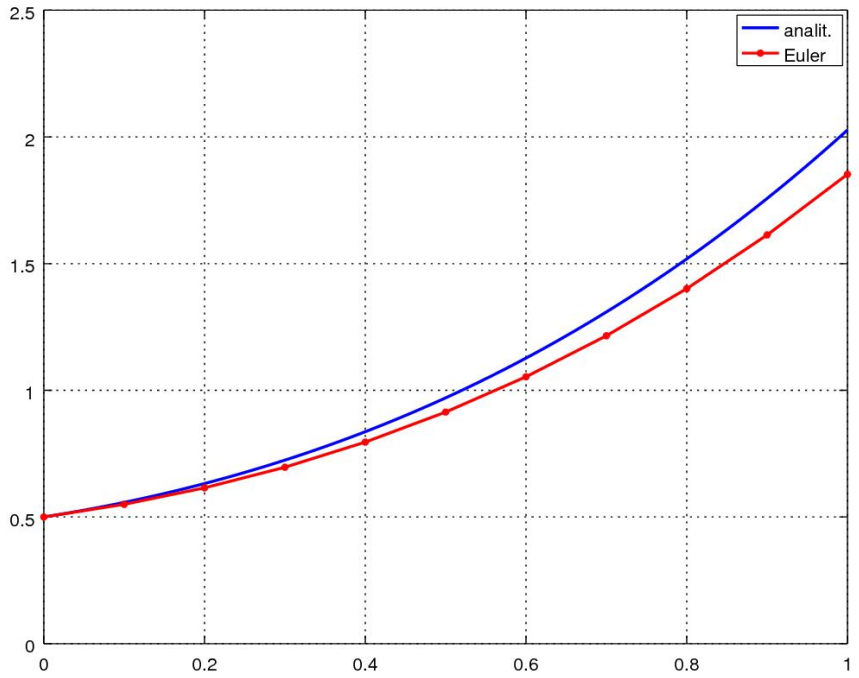

Figura 11.1: Esboço das soluções referente ao Exemplo 11.1.1.

As aproximações obtidas neste exemplo podem ser computadas no `GNU Octave` com o seguinte código:

```
f = @(t,y) y+sin(t);

h=0.1;
n=11;
t=zeros(n,1);
y=zeros(n,1);
```

```
t(1)=0;
y(1)=0.5;

for i=1:n-1
  t(i+1) = t(i)+h;
  y(i+1)=y(i)+h*f(t(i),y(i));
endfor

printf("%1.5E %1.5E\n",t(n),y(n))
tt=linspace(0,1);
ya = @(t) exp(t)-sin(t)/2-cos(t)/2;
plot(tt,ya(tt),'b-',...
     t,y,'r.-');grid
legend("analit.","Euler")
```

11.1.1 Análise de consistência e convergência

O método de Euler com passo h aplicado ao problema de valor inicial (11.3)-(11.4), pode ser escrito da seguinte forma

$$\tilde{y}(t^{(1)};h) = y_0, \tag{11.17}$$

$$\tilde{y}(t^{(i+1)};h) = \tilde{y}(t^{(i)};h) + h\Phi(t^{(i)},\tilde{y}(t^{(i)});h), \tag{11.18}$$

onde $\tilde{y}(t^{(i)})$ representa a aproximação da solução exata y no tempo $t^{(i)} = t_0 + (i-1)h$, $i = 1, 2, \ldots$. Métodos que podem ser escritos desta forma, são chamados de métodos de passo simples (ou único). No caso específico do método de Euler, temos

$$\Phi(t,y;h) := f(t,y(t)). \tag{11.19}$$

Agora, considerando a solução exata $y(t)$ de (11.3)-(11.4), introduzimos

$$\Delta(t,y;h) := \begin{cases} \frac{y(t+h)-y(t)}{h} & ,h \neq 0, \\ f(t,y(t)) & ,h = 0, \end{cases} \tag{11.20}$$

Com isso, vamos analisar o chamado **erro de discretização local**

$$\tau(t,y;h) := \Delta(t,y;h) - \Phi(t,y;h), \tag{11.21}$$

a qual estabelece uma medida quantitativa com que a solução exata $y(t)$ no tempo $t+h$ satisfaz a iteração de Euler.

Definição 11.1.1. (Consistência) Um método de passo simples (11.17)-(11.18) é dito consistente quando

$$\lim_{h\to 0} \tau(t,y;h) = 0, \tag{11.22}$$

ou, equivalentemente, quando

$$\lim_{h\to 0} \Phi(t,y;h) = f(t,y). \tag{11.23}$$

Observação 11.1.1. Da Definição 11.1.1, temos que o método de Euler é consistente.

A **ordem do erro de discretização local** de um método de passo simples (11.17)-(11.18) é dita ser p, quando

$$\tau(t,y;h) = O(h^p). \tag{11.24}$$

Para determinarmos a ordem do método de Euler, tomamos a expansão em série de Taylor da solução exata $y(t)$ em torno de t, i.e.

$$y(t+h) = y(t) + hy'(t) + \frac{h^2}{2}y''(t) + \frac{h^3}{6}y'''(t+\theta h),\ 0<\theta<1. \tag{11.25}$$

Como $y(t) = f(t,y(t))$ e assumindo a devida suavidade de f, temos

$$\begin{aligned} y''(t) &= \frac{d}{dt}f(t,y(t)) && (11.26)\\ &= f_t(t,y) + f_y(t,y)y' && (11.27)\\ &= f_t(t,y) + f_y(t,y)f(t,y). && (11.28)\end{aligned}$$

Então,

$$\Delta(t,y;h) = f(t,y(t)) + \frac{h}{2}[f_t(t,y) + f_y(t,y)f(t,y)] + O(h^2). \tag{11.29}$$

Portanto, para o método de Euler temos

$$\begin{aligned} \tau(t,y;h) &:= \Delta(t,y;h) - \Phi(t,y;h) && (11.30)\\ &= \frac{h}{2}[f_t(t,y) + f_y(t,y)f(t,y)] + O(h^2) && (11.31)\end{aligned}$$

$$= O(h). \tag{11.32}$$

Isto mostra que o método de Euler é um método de ordem 1.

A análise acima trata apenas da consistência do método de Euler. Para analisarmos a convergência de métodos de passo simples, definimos o **erro de discretização global**

$$e(t; h_n) := \tilde{y}(t; h_n) - y(t), \quad h_n := \frac{t - t_0}{n}. \tag{11.33}$$

E, com isso, dizemos que o método é **convergente** quando

$$\lim_{n\to\infty} e(t,h_n) = 0, \tag{11.34}$$

bem como, dizemos que o método tem erro de discretização global de ordem h^p quando $e(t,h_n) = O(h^p)$.

Observação 11.1.2. Pode-se mostrar que, assumindo a devida suavidade de f, que a ordem do erro de discretização global de um método de passo simples é igual a sua ordem do erro de discretização local (veja, [6, Cap. 7, Seç. 7.2]). Portanto, o método de Euler é convergente e é de ordem 1.

Exemplo 11.1.2. Consideremos o seguinte problema de valor inicial

$$y' - y = \text{sen}(t), t > 0 \tag{11.35}$$

$$y(0) = \frac{1}{2}. \tag{11.36}$$

Na Tabela 11.1, temos as aproximações $\tilde{y}(1)$ de $y(1)$ computadas pelo método de Euler com diferentes passos h.

h	$\tilde{y}(1)$	$\lvert\tilde{y}(1) - y(1)\rvert$
10^{-1}	1,85259	1,7e−01
10^{-2}	2,00853	1,9e−02
10^{-3}	2,02549	1,9e−03
10^{-5}	2,02735	4,8e−05
10^{-7}	2.02739	1,9e−07

Tabela 11.1: Resultados referentes ao Exemplo 11.1.2

Os resultados mostrados na Tabela 11.1 podem ser computados no `GNU Octave` com o auxílio do seguinte código:

```
f = @(t,y) y+sin(t);

h=1e-2;
n=round(1/h+1);
t=zeros(n,1);
y=zeros(n,1);

t(1)=0;
y(1)=0.5;

for i=1:n-1
  t(i+1) = t(i)+h;
  y(i+1)=y(i)+h*f(t(i),y(i));
endfor

ya = @(t) exp(t)-sin(t)/2-cos(t)/2;
printf("%1.5E %1.1E\n",y(n),abs(y(n)-ya(1)))
```

Exercícios

Exercício 11.1.1. Considere o seguinte problema de valor inicial

$$y' + e^{-y^2+1} = 2, \quad t > 1, \tag{11.37}$$

$$y(1) = -1. \tag{11.38}$$

Use o método de Euler com passo $h = 0{,}1$ para computar o valor aproximado de $y(2)$.

11.2 Métodos de Runge-Kutta

Os métodos de Runge-Kutta de s-estágios são métodos de passo simples da seguinte forma

$$y^{(i+1)} = y^{(i)} + h(c_1 k_1 + \cdots + c_s k_s) \tag{11.39}$$

onde

$$k_1 := f(t^{(i)}, y^{(i)}), \tag{11.40}$$

$$k_2 := f(t^{(i)} + \alpha_2 h, y^{(i)} + h\beta_{21}k_1), \tag{11.41}$$
$$k_3 := f(t^{(i)} + \alpha_3 h, y^{(i)} + h(\beta_{31}k_1 + \beta_{32}k_2)), \tag{11.42}$$
$$\vdots \tag{11.43}$$
$$k_s := f(t^{(i)} + \alpha_s h, y^{(i)} + h(\beta_{s1}k_1 + \cdots + \beta_{s,s-1}k_{s-1})), \tag{11.44}$$

$t^{(i)} = t_0 + (i-1)h$ e $y^{(1)} = y_0$.

Na sequência, discutimos alguns dos métodos de Runge-Kutta usualmente utilizados. Pode-se encontrar uma lista mais completa em [3, Cap. 8, Seç. 3.2].

11.2.1 Métodos de Runge-Kutta de ordem 2

Precisamos apenas de 2 estágios para obtermos métodos de Runge-Kutta de ordem 2. Portanto, assumimos

$$y^{(i+1)} = y^{(i)} + h\left[c_1 f(t^{(i)}, y^{(i)}) + c_2 f(t^{(i)} + \alpha_2 h, y^{(i)} + h\beta_{21} f(t^{(i)}, y^{(i)}))\right]. \tag{11.45}$$

Neste caso, o erro de discretização local é dado por

$$\tau(t,y;h) = \Delta(t,y;h) - \Phi(t,y;h), \tag{11.46}$$

onde, da equação (11.29) temos

$$\Delta(t,y;h) = f(t,y(t)) + \frac{h}{2}[f_t(t,y) + f_y(t,y)f(t,y)] + O(h^2) \tag{11.47}$$

e de (11.45)

$$\Phi(t,y;h) = c_1 f(t,y) + c_2 f(t + \alpha_2 h, y + h\beta_{21} f(t,y)) \tag{11.48}$$

Agora, tomando a expansão de série de Taylor em torno de t de $\Phi(t,y;h)$, temos

$$\Phi(t,y;h) = (c_1 + c_2)f(t,y) + c_2 h[\alpha_2 f_t(t,y) + \beta_{21} f_y(t,y) f(t,y)) + O(h^2). \tag{11.49}$$

Então, por comparação de (11.47) e (11.49), temos

$$c_1 + c_2 = 1 \tag{11.50}$$

$$c_2\alpha_2 = \frac{1}{2} \tag{11.51}$$

$$c_2\beta_{21} = \frac{1}{2}. \tag{11.52}$$

Assim sendo, temos mais de uma solução possível.

Método do ponto médio

O método do ponto médio é um método de Runge-Kutta de ordem 2 proveniente da escolha de coeficientes

$$c_1 = 0, \quad c_2 = 1, \quad \alpha_2 = \frac{1}{2}, \quad \beta_{21} = \frac{1}{2}. \tag{11.53}$$

Logo, a iteração do método do ponto médio é

$$y^{(1)} = y_0 \tag{11.54}$$

$$y^{(i+1)} = y^{(i)} + hf\left(t^{(i)} + \frac{h}{2}, y^{(i)} + \frac{h}{2}f(t^{(i)}, y^{(i)})\right). \tag{11.55}$$

Exemplo 11.2.1. Consideremos o seguinte problema de valor inicial

$$y' - y = \text{sen}(t), t > 0 \tag{11.56}$$

$$y(0) = \frac{1}{2}. \tag{11.57}$$

Na Tabela 11.2, temos as aproximações $\tilde{y}(1)$ de $y(1)$ computadas pelo método do ponto médio com diferentes passos h.

h	$\tilde{y}(1)$	$\lvert\tilde{y}(1) - y(1)\rvert$
10^{-1}	2,02175	5,6e−03
10^{-2}	2,02733	6,0e−05
10^{-3}	2,02739	6,1e−07
10^{-4}	2,02740	6,1e−09
10^{-5}	2,02737	2,9e−05

Tabela 11.2: Resultados referentes ao Exemplo 11.2.1.

Os resultados mostrados na Tabela 11.2 podem ser computados no `GNU Octave` com o auxílio do seguinte código:

```
f = @(t,y) y+sin(t);

h=1e-1;
n=round(1/h+1);
t=zeros(n,1);
y=zeros(n,1);

t(1)=0;
y(1)=0.5;

for i=1:n-1
  t(i+1) = t(i)+h;
  y(i+1)=y(i)+h*f(t(i)+h/2,y(i)+h/2*f(t(i),y(i)));
endfor

ya = @(t) exp(t)-sin(t)/2-cos(t)/2;
printf("%1.5E %1.1E\n",y(n),abs(y(n)-ya(1)))
```

Método de Euler modificado

O método de Euler modificado é um método de Runge-Kutta de ordem 2 proveniente da escolha de coeficientes

$$c_1 = \frac{1}{2}, \quad c_2 = \frac{1}{2}, \quad \alpha_2 = 1, \quad \beta_{21} = 1. \tag{11.58}$$

Logo, a iteração do método de Euler modificado é

$$y^{(1)} = y_0 \tag{11.59}$$

$$y^{(i+1)} = y^{(i)} + \frac{h}{2}\left[f(t^{(i)},y^{(i)}) + f(t^{(i)}+h,y^{(i)}+hf(t^{(i)},y^{(i)})\right]. \tag{11.60}$$

Exemplo 11.2.2. Consideremos o seguinte problema de valor inicial

$$y' - y = \text{sen}(t), t > 0 \tag{11.61}$$

$$y(0) = \frac{1}{2}. \tag{11.62}$$

Na Tabela 11.3, temos as aproximações $\tilde{y}(1)$ de $y(1)$ computadas pelo método de Euler modificado com diferentes passos h.

h	$\tilde{y}(1)$	$\lvert\tilde{y}(1) - y(1)\rvert$
10^{-1}	2,02096	6,4e−03
10^{-2}	2,02733	6,9e−05
10^{-3}	2,02739	6,9e−07
10^{-4}	2,02740	6,9e−09
10^{-5}	2.02737	2,9e−05

Tabela 11.3: Resultados referentes ao Exemplo 11.2.2

Os resultados mostrados na Tabela 11.3 podem ser computados no `GNU Octave` com o auxílio do seguinte código:

```
f = @(t,y) y+sin(t);

h=1e-1;
n=round(1/h+1);
t=zeros(n,1);
y=zeros(n,1);

t(1)=0;
y(1)=0.5;

for i=1:n-1
  t(i+1) = t(i)+h;
  y(i+1)=y(i)+h*f(t(i),y(i));
  y(i+1)=y(i)+h/2*(f(t(i),y(i))+f(t(i+1),y(i+1)));
endfor

ya = @(t) exp(t)-sin(t)/2-cos(t)/2;
printf("%1.5E %1.1E\n",y(n),abs(y(n)-ya(1)))
```

11.2.2 Método de Runge-Kutta de ordem 4

Um dos métodos de Runge-Kutta mais empregados é o seguinte método de ordem 4:

$$y^{(i+1)} = y^{(i)} + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{11.63}$$

onde

$$k_1 := f(t^{(i)},y^{(i)}), \tag{11.64}$$
$$k_2 := f(t^{(i)} + h/2,y^{(i)} + hk_1/2), \tag{11.65}$$
$$k_3 := f(t^{(i)} + h/2,y^{(i)} + hk_2/2), \tag{11.66}$$
$$k_4 := f(t^{(i)} + h,y^{(i)} + hk_3), \tag{11.67}$$

$t^{(i)} = t_0 + (i-1)h$ e $y^{(1)} = y_0$.

Exemplo 11.2.3. Consideremos o seguinte problema de valor inicial

$$y' - y = \text{sen}(t), t > 0 \tag{11.68}$$
$$y(0) = \frac{1}{2}. \tag{11.69}$$

Na Tabela 11.4, temos as aproximações $\tilde{y}(1)$ de $y(1)$ computadas pelo método de Runge-Kutta de quarta ordem com diferentes passos h.

h	$\tilde{y}(1)$	$\lvert\tilde{y}(1) - y(1)\rvert$
10^{-1}	2,02739	2,8e−06
10^{-2}	2,02740	3,1e−10
10^{-3}	2,02740	3,0e−14
10^{-4}	2,02740	4,4e−14

Tabela 11.4: Resultados referentes ao Exemplo 11.2.3

Os resultados mostrados na Tabela 11.4 podem ser computados no `GNU Octave` com o auxílio do seguinte código:

```
f = @(t,y) y+sin(t);

h=1e-4;
n=round(1/h+1);
t=zeros(n,1);
y=zeros(n,1);

t(1)=0;
y(1)=0.5;
```

```
for i=1:n-1
  t(i+1) = t(i)+h;
  k1 = h*f(t(i),y(i));
  k2 = h*f(t(i)+h/2,y(i)+k1/2);
  k3 = h*f(t(i)+h/2,y(i)+k2/2);
  k4 = h*f(t(i)+h,y(i)+k3);
  y(i+1)=y(i)+(k1+2*k2+2*k3+k4)/6;
endfor

ya = @(t) exp(t)-sin(t)/2-cos(t)/2;
printf("%1.5E %1.1E\n",y(n),abs(y(n)-ya(1)))
```

Exercícios

Exercício 11.2.1. Considere o seguinte problema de valor inicial

$$y' + e^{-y^2+1} = 2, \quad t > 1, \tag{11.70}$$

$$y(1) = -1. \tag{11.71}$$

Use os seguintes métodos de Runge-Kutta com passo $h = 0{,}1$ para computar o valor aproximado de $y(2)$:

a) método do ponto médio.

b) método de Euler modificado.

c) método de Runge-Kutta de ordem 4.

11.3 Método adaptativo com controle de erro

Consideremos um problema de valor inicia

$$y'(t) = f(t,y(t)), \quad t > t_0, \tag{11.72}$$

$$y(t_0) = y_0. \tag{11.73}$$

e um método de passo simples

$$y^{(1)} = y_0, \tag{11.74}$$

$$y^{(i+1)}(h^{(i+1)}) = y^{(i)} + h^{(i+1)}\Phi(t^{(i)},y^{(i)};h^{(i+1)}), \tag{11.75}$$

com $t^{(i)} = t_0 + (i-1)h^{(i)}$. Nesta seção, discutiremos uma estimava para o maior valor de $h^{(i+1)}$ tal que o erro de discretização global $e(t^{(i+1)}; h^{(i+1)})$ seja controlado por uma dada tolerância TOL, i.e.

$$|e(t^{(i+1)}; h^{(i+1)})| := |y^{(i+1)}(h^{(i+1)}) - y(t^{(i+1)})| \approx TOL. \tag{11.76}$$

Para um método de ordem h^p, pode-se mostrar que (veja, [3, Cap. 7, Seç. 7.2])

$$y^{(i+1)}(h^{(i+1)}) = y(t^{(i+1)}) + e_p(t^{(i+1)})(h^{(i+1)})^p, \tag{11.77}$$

onde $e(t^{(i+1)})$ é uma função apropriada. Então, assumindo que $e(t^{(i)}; h^{(i)}) = 0$, temos

$$e_p(t^{(i+1)}) = h^{(i+1)} e'_p(t^{(i)}) \tag{11.78}$$

e, portanto, para termos (11.76) impomos que

$$|(h^{(i+1)})^{p+1} e'_p(t^{(i)})| = TOL. \tag{11.79}$$

Daí, se obtermos uma aproximação para $e'_p(t^{(i)})$ teremos uma aproximação para o passo $h^{(i+1)}$.

Para estimarmos $e_p(t^{(i+1)})$, observamos que de (11.77) temos

$$y^{(i+1)}\left(\frac{h^{(i+1)}}{2}\right) = y(t^{(i+1)}) + e_p(t^{(i+1)})\frac{(h^{(i+1)})^p}{2^p} \tag{11.80}$$

e, então, subtraindo esta de (11.77) temos

$$y^{(i+1)}(h^{(i+1)}) - y^{(i+1)}\left(\frac{h^{(i+1)}}{2}\right) = e_p(t^{(i+1)})\left(\frac{h^{(i+1)}}{2}\right)^p (2^p - 1), \tag{11.81}$$

donde

$$e_p(t^{(i+1)})\left(\frac{h^{(i+1)}}{2}\right)^p = \frac{y^{(i+1)}(h^{(i+1)}) - y^{(i+1)}\left(\frac{h^{(i+1)}}{2}\right)}{2^p - 1}. \tag{11.82}$$

Daí, de (11.78), obtemos

$$e'_p(t^{(i)})h^{(i+1)}\left(\frac{h^{(i+1)}}{2}\right)^p = \frac{y^{(i+1)}(h^{(i+1)}) - y^{(i+1)}\left(\frac{h^{(i+1)}}{2}\right)}{2^p - 1}, \tag{11.83}$$

o que nos fornece a seguinte aproximação de $e'_p(t^{(i)})$

$$e'_p(t^{(i)}) = \frac{1}{(h^{(i+1)})^{p+1}} \frac{2^p}{2^p - 1} \left[y^{(i+1)}(h^{(i+1)}) - y^{(i+1)} \left(\frac{h^{(i+1)}}{2} \right) \right]. \quad (11.84)$$

Assim sendo, de (11.79) temos que o passo $h^{(i+1)}$ apropriado é tal que

$$\frac{2^p}{2^p - 1} \left| y^{(i+1)}(h^{(i+1)}) - y^{(i+1)} \left(\frac{h^{(i+1)}}{2} \right) \right| \approx TOL. \quad (11.85)$$

Com base nesta estimativa podemos propor o seguinte método de passo adaptativo. Partindo de uma escolha arbitrária de h, computamos $y^{(i+1)}(h)$ e $y^{(i+1)}(h/2)$ de $y^{(i)}$. Então, enquanto

$$\frac{2^p}{2^p - 1} \left| y^{(i+1)}(h) - y^{(i+1)} \left(\frac{h}{2} \right) \right| > TOL, \quad (11.86)$$

tomamos sucessivas divisões de h por 2, até satisfazermos (11.85). Obtido o h que satisfaz (11.85), temos computado $y^{(i+1)}$ com $h^{(i+1)} = h$.

Exemplo 11.3.1. Consideremos o seguinte problema de valor inicial

$$y' - y = \text{sen}(t), t > 0 \quad (11.87)$$

$$y(0) = \frac{1}{2}. \quad (11.88)$$

A Figura 11.2 mostra a comparação entre $y(t)$ e a solução numérica obtida da aplicação do método de Euler com passo adaptativo. No método, utilizamos o passo inicial $h^{(1)} = 0{,}1$ e tolerância $TOL = 10^{-4}$. Ao compararmos esta figura com a Figura (11.1) fica evidente o controle do erro.

O algoritmo utilizado neste exemplo pode ser implementado no `GNU Octave` com o seguinte código:

```
f = @(t,y) y+sin(t);

TOL=1e-4;
h=1e-1;
tf=1;
```

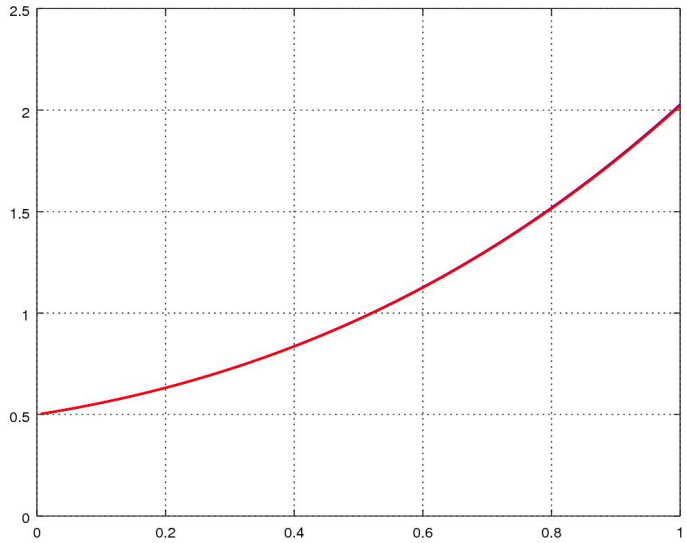

Figura 11.2: Resultados referentes ao Exemplo 11.3.1.

```
t0=0;
y0=0.5;

t=[];
y=[];

c=1;
do

  h = min(h,tf-t0);

  do
    #passo h
    y1=y0+h*f(t0,y0);
    #passo h/2
    y2=y0+h/2*f(t0,y0);
    y2=y2+h/2*f(t0+h/2,y2);
    #verifica TOL
```

```
    est = 2*abs(y1-y2);
    if (est > TOL)
      h/=2;
      if (h<1e-8)
        error("h muito pequeno")
      endif
    else
      t0+=h;
      y0=y2;

      t(c)=t0;
      y(c)=y0;
      c+=1;
    endif
  until ((est <= TOL))

until (abs(t0-tf)<1e-14)

ya = @(t) exp(t)-sin(t)/2-cos(t)/2;
printf("%1.1E %1.5E %1.1E\n",t0,y0,abs(y0-ya(1)))

plot(t,ya(t),'b-',t,y,'r-');grid
```

Exercícios

Exercício 11.3.1. Considere o seguinte problema de valor inicial

$$y' + e^{-y^2+1} = 2, \quad t > 1, \tag{11.89}$$

$$y(1) = -1. \tag{11.90}$$

Use o método de Euler com passo adaptativo para computar o valor aproximado de $y(2)$. Para tanto, utilize o passo inicial $h = 0{,}1$ e a tolerância de $TOL = 10^{-4}$.

11.4 Métodos de passo múltiplo

Dado um problema de valor inicial

$$y'(t) = f(t,y(t)), \quad t > t_0, \tag{11.91}$$
$$y(t_0) = y_0. \tag{11.92}$$

temos

$$y(t) = y(t_0) + \int_{t_0}^{t} f(s,y(s))\, ds. \tag{11.93}$$

De forma mais geral, consideramos uma partição uniforme no tempo $\{t_0 = t^{(1)} < t^{(2)} < \cdots < t^{(i)} < \cdots < t^{(n)} = t_f\}$, onde t_f é um determinado tempo para o qual queremos computar uma aproximação para $y(t_f)$. Também, denotamos o passo no tempo por $h = (t_f - t_0)/n$. Com isso, a solução $y(t)$ satisfaz

$$y\left(t^{(i+k)}\right) = y\left(t^{(i-j)}\right) + \int_{t^{(i-j)}}^{t^{(i+k)}} f(s,y(s))\, ds. \tag{11.94}$$

A ideia é, então, aproximar a integral acima por uma quadratura numérica.

Seguindo as regras de Newton-Cotes (veja, Cap. 10 Seç. 10.1), escolhemos os nodos da quadratura como $x_l = t^{(i-l+1)}$, $l = 1, 2, \ldots, m$, e, então

$$\int_{t^{(i-j)}}^{t^{i+k}} f(x,y(x))\, dx \approx \sum_{l=1}^{m} f\left(x_l, y(x_l)\right) w_l, \tag{11.95}$$

e

$$w_l = \int_{t^{(i-j)}}^{t^{(i+k)}} \prod_{\substack{p=1 \\ p\neq l}}^{m} \frac{x - x_p}{x_l - x_p}\, dx. \tag{11.96}$$

Agora, fazendo a mudança de variável $u = (x - t^{(i)})/h$, obtemos

$$w_l = h \int_{-j}^{k} \prod_{\substack{p=1 \\ p\neq l}}^{m} \frac{u + p - 1}{-l + p}\, du \tag{11.97}$$

Assim sendo, temos o seguinte esquema numérico

$$y^{(i+k)} = y^{(i-j)} + h \sum_{l=1}^{m} c_l f(t^{(i-l+1)}, y^{(i-l+1)}), \tag{11.98}$$

onde

$$c_l = \int_{-j}^{k} \prod_{\substack{p=1 \\ p \neq l}}^{m} \frac{s+p-1}{-l+p}\, ds. \tag{11.99}$$

Diferentes escolhas de j, k e m não fornecem diferentes métodos. Observamos, ainda, que a ordem de um tal método de passo múltiplo é determinada pela ordem de truncamento da quadratura numérica usada (veja, por exemplo, [2, Cap. 5, Seç. 5.6]).

11.4.1 Métodos de Adams-Bashforth

Métodos de Adams-Bashforth são métodos de passo múltiplo obtidos ao escolhermos $j = 0$ e $k = 1$ no esquema numérico (11.98). Com isso, ao escolhermos m obtemos um método de ordem $O(h^m)$ [2, Cap. 5, Seç. 5.6].

Método de Adams-Bashforth de ordem 2

Tomando $m = 2$ em (11.99), temos

$$c_1 = \int_0^1 s + 1\, ds = \frac{3}{2} \tag{11.100}$$

e

$$c_2 = \int_0^1 -s\, ds = -\frac{1}{2}. \tag{11.101}$$

Então, de (11.98) temos a iteração do **método de Adams-Bashforth de 2 passos**:

$$y^{(1)} = y_0, \tag{11.102}$$

$$y^{(i+1)} = y^{(i)} + \frac{h}{2}\left[3f(t^{(i)},y^{(i)}) - f(t^{(i-1)},y^{(i-1)})\right], \tag{11.103}$$

com $t^{(i)} = t_0 + (i-1)h$.

Exemplo 11.4.1. Consideremos o seguinte problema de valor inicial

$$y' - y = \text{sen}(t), t > 0 \tag{11.104}$$

$$y(0) = \frac{1}{2}. \tag{11.105}$$

Na Tabela 11.5, temos as aproximações $\tilde{y}(1)$ de $y(1)$ computadas pelo método de Adams-Bashforth de 2 passos. Como este método é de ordem 2, escolhemos inicializá-lo pelo método do ponto médio, de forma a mantermos a consistência.

h	$\tilde{y}(1)$	$\lvert\tilde{y}(1) - y(1)\rvert$
10^{-1}	2,01582	1,2e−02
10^{-2}	2,02727	1,3e−04
10^{-3}	2,02739	1,3e−06
10^{-4}	2,02740	1,3e−08
10^{-5}	2,02740	1,3e−10

Tabela 11.5: Resultados referentes ao Exemplo 11.4.1

Os resultados mostrados na Tabela 11.5 podem ser computados no `GNU Octave` com o auxílio do seguinte código:

```
f = @(t,y) y+sin(t);

h=1e-1;
n=round(1/h+1);
t=zeros(n,1);
y=zeros(n,1);

#c.i.
t(1)=0;
y(1)=0.5;

#inicializacao
t(2)=t(1)+h;
y(2)=y(1)+h*f(t(1)+h/2,y(1)+h/2*f(t(1),y(1)));

#iteracoes
for i=2:n-1
  t(i+1) = t(i)+h;
  y(i+1)=y(i) + ...
        h/2*(3*f(t(i),y(i))-f(t(i-1),y(i-1)));
```

```
endfor

ya = @(t) exp(t)-sin(t)/2-cos(t)/2;
printf("%f %1.5E %1.1E\n",t(n),y(n),abs(y(n)-ya(1)))
```

Método de Adams-Bashforth de ordem 3

Tomando $m = 3$ em (11.99) obtemos, de (11.98), a iteração do **método de Adams-Bashforth de** 3 **passos**:

$$y^{(1)} = y_0, \tag{11.106}$$

$$y^{(i+1)} = y^{(i)} + \frac{h}{12}\left[23f(t^{(i)},y^{(i)}) - 16f(t^{(i-1)},y^{(i-1)}) + 5f(t^{(i-2)},y^{(i-2)})\right], \tag{11.107}$$

com $t^{(i)} = t_0 + (i-1)h$.

Exemplo 11.4.2. Consideremos o seguinte problema de valor inicial

$$y' - y = \text{sen}(t), t > 0 \tag{11.108}$$

$$y(0) = \frac{1}{2}. \tag{11.109}$$

Na Tabela 11.6, temos as aproximações $\tilde{y}(1)$ de $y(1)$ computadas pelo método de Adams-Bashforth de 3 passos. Como este método é de ordem 3, escolhemos inicializá-lo pelo método de Runge-Kutta de ordem 4, de forma a garantirmos a consistência.

h	$\tilde{y}(1)$	$\lvert\tilde{y}(1) - y(1)\rvert$
10^{-1}	2,02696	4,3e−04
10^{-2}	2,02739	5,9e−07
10^{-3}	2,02740	6,1e−10
10^{-4}	2,02740	6,6e−13

Tabela 11.6: Resultados referentes ao Exemplo 11.4.2

Os resultados mostrados na Tabela 11.6 podem ser computados no `GNU Octave` com o auxílio do seguinte código:

```
f = @(t,y) y+sin(t);

h=1e-1;
n=round(1/h+1);
t=zeros(n,1);
y=zeros(n,1);

#c.i.
t(1)=0;
y(1)=0.5;

#inicializacao
for i=1:2
  t(i+1)=t(i)+h;
  k1=h*f(t(i),y(i));
  k2=h*f(t(i)+h/2,y(i)+k1/2);
  k3=h*f(t(i)+h/2,y(i)+k2/2);
  k4=h*f(t(i)+h,y(i)+k3);
  y(i+1)=y(i)+(k1+2*k2+2*k3+k4)/6;
endfor

#iteracoes
for i=3:n-1
  t(i+1) = t(i)+h;
  y(i+1)=y(i) + ...
        h/12*(23*f(t(i),y(i)) ...
        -16*f(t(i-1),y(i-1)) ...
        +5*f(t(i-2),y(i-2)));
endfor

ya = @(t) exp(t)-sin(t)/2-cos(t)/2;
printf("%f %1.5E %1.1E\n",t(n),y(n),abs(y(n)-ya(1)))
```

Método de Adams-Bashforth de ordem 4

Tomando $m = 4$ em (11.99) obtemos, de (11.98), a iteração do **método de Adams-Bashforth de** 4 **passos**:

$$y^{(1)} = y_0, \tag{11.110}$$

$$y^{(i+1)} = y^{(i)} + \frac{h}{24}\left[55f(t^{(i)},y^{(i)}) - 59f(t^{(i-1)},y^{(i-1)}) + 37f(t^{(i-2)},y^{(i-2)}) - 9f(t^{(i-3)},y^{(i-3)})\right], \quad (11.111)$$

com $t^{(i)} = t_0 + (i-1)h$.

Exemplo 11.4.3. Consideremos o seguinte problema de valor inicial

$$y' - y = \text{sen}(t), t > 0 \quad (11.112)$$

$$y(0) = \frac{1}{2}. \quad (11.113)$$

Na Tabela 11.7, temos as aproximações $\tilde{y}(1)$ de $y(1)$ computadas pelo método de Adams-Bashforth de 4 passos. Como este método é de ordem 3, escolhemos inicializá-lo pelo método de Runge-Kutta de ordem 4, de forma a mantermos a consistência.

h	$\tilde{y}(1)$	$\lvert\tilde{y}(1) - y(1)\rvert$
10^{-1}	2,02735	5,0e−05
10^{-2}	2,02740	7,7e−09
10^{-3}	2,02740	7,9e−13

Tabela 11.7: Resultados referentes ao Exemplo 11.4.3

Os resultados mostrados na Tabela 11.7 podem ser computados no `GNU Octave` com o auxílio do seguinte código:

```
f = @(t,y) y+sin(t);

h=1e-1;
n=round(1/h+1);
t=zeros(n,1);
y=zeros(n,1);

#c.i.
t(1)=0;
y(1)=0.5;

#inicializacao
```

```
for i=1:3
  t(i+1)=t(i)+h;
  k1=h*f(t(i),y(i));
  k2=h*f(t(i)+h/2,y(i)+k1/2);
  k3=h*f(t(i)+h/2,y(i)+k2/2);
  k4=h*f(t(i)+h,y(i)+k3);
  y(i+1)=y(i)+(k1+2*k2+2*k3+k4)/6;
endfor

#iteracoes
for i=4:n-1
  t(i+1) = t(i)+h;
  y(i+1)=y(i) + ...
        h/24*(55*f(t(i),y(i)) ...
        -59*f(t(i-1),y(i-1)) ...
        +37*f(t(i-2),y(i-2)) ...
        -9*f(t(i-3),y(i-3)));
endfor

ya = @(t) exp(t)-sin(t)/2-cos(t)/2;
printf("%f %1.5E %1.1E\n",t(n),y(n),abs(y(n)-ya(1)))
```

Exercícios

Exercício 11.4.1. Considere o seguinte problema de valor inicial

$$y' + e^{-y^2+1} = 2, \quad t > 1, \tag{11.114}$$

$$y(1) = -1. \tag{11.115}$$

Inicializando pelo método de Euler, use os seguintes métodos de passo múltiplo com $h = 0{,}1$ para computar o valor aproximado de $y(2)$:

a) método de Adams-Bashforth de ordem 2.

b) método de Adams-Bashforth de ordem 3.

c) método de Adams-Bashforth de ordem 4.

Capítulo 12

Problema de valor de contorno

Neste capítulo, discutimos sobre a aplicação do método de diferenças finitas para aproximar a solução de problemas de valores de contorno da forma

$$\alpha(x)u'' + \beta(x)u' + \gamma(x)u = f(x), \quad c_1 < x < c_2, \tag{12.1}$$
$$\eta_1 u'(c_1) + \theta_1 u(c_1) = g_1 \tag{12.2}$$
$$\eta_2 u'(c_2) + \theta_2 u(c_2) = g_2 \tag{12.3}$$

onde a incógnita $u = u(x)$ e os são dados os coeficientes $\alpha(x) \neq 0$, $\beta(x)$, $\gamma(x)$ e a função $f(x)$. Nas condições de contorno, são dados os coeficientes η_1 e θ_1 não simultaneamente nulos, bem como, os coeficientes η_2 e θ_2, também, não simultaneamente nulos.

12.1 Método de diferenças finitas

Consideramos o seguinte problema linear de valor de contorno

$$\alpha(x)u'' + \beta(x)u' + \gamma(x)u = f(x), \quad c_1 < x < c_2, \tag{12.4}$$
$$\eta_1 u'(c_1) + \theta_1 u(c_1) = g_1 \tag{12.5}$$
$$\eta_2 u'(c_2) + \theta_2 u(c_2) = g_2 \tag{12.6}$$

onde a incógnita $u = u(x)$ e os são dados os coeficientes $\alpha(x) \neq 0$, $\beta(x)$, $\gamma(x)$ e a função $f(x)$. Nas condições de contorno, são dados os coeficientes η_1 e θ_1 não simultaneamente nulos, bem como, os coeficientes η_2 e θ_2, também, não simultaneamente nulos.

A aproximação pelo método de diferenças finitas de (12.4)-(12.6) surge da substituição das derivadas por fórmulas de diferenças finitas. Isto requer a a prévia discretização do domínio do problema. Mais precisamente, a aplicação do método de diferenças finitas envolve três procedimentos básicos: 1. discretização do domínio, 2. discretização das equações, 3. resolução do problema discreto.

1. Discretização do domínio

A discretização do domínio refere-se ao particionamento do mesmo em pontos espaçados uniformemente ou não. Aqui, para mantermos a simplicidade, vamos considerar apenas o caso de um particionamento uniforme. Desta forma, escolhemos o número n de pontos da partição e, então, o passo é dado por

$$h = \frac{c_2 - c_1}{n-1}, \tag{12.7}$$

e os pontos da partição podem ser indexados da seguinte forma

$$x_i = c_1 + (i-1)h. \tag{12.8}$$

2. Discretização das equações

Começando pela equação (12.4), no ponto $x = x_i$ temos

$$\alpha(x_i)u''(x_i) + \beta(x_i)u'(x_i) + \gamma(x_i)u(x_i) = f(x_i) \tag{12.9}$$

para $i = 2, 3, \ldots, n-1$. Podemos substituir a segunda derivada de u pela fórmula de diferenças finitas central de ordem h^2, i.e.

$$u''(x_i) = \underbrace{\frac{u(x_i - h) - 2u(x_i) + u(x_i + h)}{h^2}}_{D^2_{0,h^2}u(x_i)} + O(h^2). \tag{12.10}$$

A primeira derivada de u também pode ser substituída pela fórmula de diferenças finitas central de ordem h^2, i.e.

$$u'(x_i) = \underbrace{\frac{u(x_i + h) - u(x_i - h)}{2h}}_{D_{0,h^2}u(x_i)} + O(h^2). \tag{12.11}$$

Agora, denotando $u_i \approx u(x_i)$, temos $u_{i-1} \approx u(x_i - h)$ e $u_{i+1} \approx u(x_i + h)$. Então, substituindo as derivadas pelas fórmulas de diferenças finitas acima na equação (12.9), obtemos

$$\alpha(x_i)\left(\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2}\right) + \beta(x_i)\left(\frac{u_{i+1} - u_{i-1}}{2h}\right) + \gamma(x_i)u_i + O(h^2) = f(x_i), \tag{12.12}$$

para $i = 2, 3, \ldots, n-1$. Rearranjando os termos e desconsiderando o termo do erro de truncamento, obtemos o seguinte sistema discreto de equações lineares

$$\left(\frac{\alpha(x_i)}{h^2} - \frac{\beta(x_i)}{2h}\right)u_{i-1} + \left(\gamma(x_i) - \frac{2\alpha(x_i)}{h^2}\right)u_i + \left(\frac{\alpha(x_i)}{h^2} + \frac{\beta(x_i)}{2h}\right)u_{i+1} = f(x_i), \tag{12.13}$$

para $i = 2, 3, \ldots, n-1$. Observe que este sistema consiste em $n-2$ equações envolvendo as n incógnitas u_i, $i = 1, 2, \ldots, n$. Para fechá-lo, usamos as condições de contorno.

Usando a fórmula de diferenças finitas progressiva de ordem h^2 para a derivada $u'(c_1)$ temos

$$u'(c_1) = \frac{-3u(c_1) + 4u(c_1 + h) - u(c_1 + 2h}{2h} + O(h^2). \tag{12.14}$$

Então, observando que c_1 corresponde ao ponto x_1 na partição do domínio, temos $u_1 \approx u(c_1)$, $u_2 = u(c_1 + h)$ e $u_3 = u(c_1 + 2h)$ e, portanto de (12.5) temos

$$\eta_1\left(\frac{-3u_1 + 4u_2 - u_3}{2h}\right) + \theta_1 u_1 + O(h^2) = g_1. \tag{12.15}$$

Então, desconsiderando o termo do erro de truncamento, obtemos a seguinte equação discreta

$$\left(\theta_1 - \frac{3\eta_1}{2h}\right)u_1 + \frac{2\eta_1}{h}u_2 - \frac{\eta_1}{2h}u_3 = g_1. \tag{12.16}$$

Procedendo de forma análoga para a condição de contorno (12.6), usamos a fórmula de diferenças finitas regressiva de ordem h^2 para a derivada $u'(c_2)$,

i.e.

$$u'(c_2) = \frac{3u(c_2) - 4u(c_2 - h) + u(c_2 - 2h)}{2h} + O(h^2). \tag{12.17}$$

Aqui, temos $u_n \approx u(c_2)$, $u_{n-1} \approx u(c_2 - h)$ e $u_{n-2} \approx u(c_2 - 2h)$, e de (12.6) obtemos

$$\eta_2 \left(\frac{3u_n - 4u_{n-1} + u_{n-2}}{2h} \right) + \theta_2 u_n + O(h^2) = g_2. \tag{12.18}$$

Então, desconsiderando o termo do erro de truncamento, obtemos

$$\frac{\eta_2}{2h} u_{n-2} - \frac{2\eta_2}{h} u_{n-1} + \left(\theta_2 + \frac{3\eta_2}{2h} \right) u_n = g_2. \tag{12.19}$$

Por fim, as equações (12.16)-(12.19) formam o seguinte problema discretizado pelo método de diferenças finitas

$$\left(\theta_1 - \frac{3\eta_1}{2h} \right) u_1 + \frac{2\eta_1}{h} u_2 - \frac{\eta_1}{2h} u_3 = g_1. \tag{12.20}$$

$$\begin{aligned} &\left(\frac{\alpha(x_i)}{h^2} - \frac{\beta(x_i)}{2h} \right) u_{i-1} + \left(\gamma(x_i) - \frac{2\alpha(x_i)}{h^2} \right) u_i \\ &+ \left(\frac{\alpha(x_i)}{h^2} + \frac{\beta(x_i)}{2h} \right) u_{i+1} = f(x_i),\ i = 2, \ldots, n-1, \end{aligned} \tag{12.21}$$

$$\frac{\eta_2}{2h} u_{n-2} - \frac{2\eta_2}{h} u_{n-1} + \left(\theta_2 + \frac{3\eta_2}{2h} \right) u_n = g_2. \tag{12.22}$$

3. Resolução do problema discreto

O problema discreto (12.20)-(12.22) consiste em um sistema linear de n equações com n incógnitas. Na forma matricial temos

$$A\tilde{u} = b \tag{12.23}$$

onde $\tilde{u} = (u_1, u_2, \ldots, u_n)$ é o vetor das incógnitas, b é o vetor dos termos contantes $b = (g_1, f(x_2), f(x_3), \ldots, f(x_{n-1}), g_2)$ e A é a matriz dos coeficientes. Observamos que os coeficientes não nulos da matriz A são:

$$a_{11} = \left(\theta_1 - \frac{3\eta_1}{2h} \right), \tag{12.24}$$

$$a_{12} = \frac{2\eta_1}{h}, \tag{12.25}$$
$$a_{13} = -\frac{\eta_1}{2h}, \tag{12.26}$$

$$a_{i,i-1} = \left(\frac{\alpha(x_i)}{h^2} - \frac{\beta(x_i)}{2h}\right), \ i = 2, \ldots, n-1, \tag{12.27}$$
$$a_{i,i} = \left(\gamma(x_i) - \frac{2\alpha(x_i)}{h^2}\right), \ i = 2, \ldots, n-1, \tag{12.28}$$
$$a_{i,i+1} = \left(\frac{\alpha(x_i)}{h^2} + \frac{\beta(x_i)}{2h}\right), \ i = 2, \ldots, n-1, \tag{12.29}$$

$$a_{n,n-2} = \frac{\eta_2}{2h}, \tag{12.30}$$
$$a_{n,n-1} = -\frac{2\eta_2}{h}, \tag{12.31}$$
$$a_{n,n} = \left(\theta_2 + \frac{3\eta_2}{2h}\right). \tag{12.32}$$

Com isso em mente, a matriz A tem a seguinte estrutura

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & & & & \\ a_{21} & a_{22} & a_{23} & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & a_{i,i-1} & a_{i,i} & a_{i,i+1} & & \\ & & \ddots & \ddots & \ddots & & \\ & & & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} & \\ & & & a_{n,n-2} & a_{n,n-1} & a_{n,n} \end{bmatrix}. \tag{12.33}$$

A resolução do sistema discreto se resume, então, a resolver o sistema $A\tilde{u} = b$, o que pode ser feito por qualquer método numérica apropriada.

Exemplo 12.1.1. Consideremos o seguinte problema de valor de contorno

$$-u'' = \text{sen}(x), \quad 0 \leq x \leq 2, \tag{12.34}$$
$$u(0) = 0, \tag{12.35}$$
$$u(2) = \text{sen}(2). \tag{12.36}$$

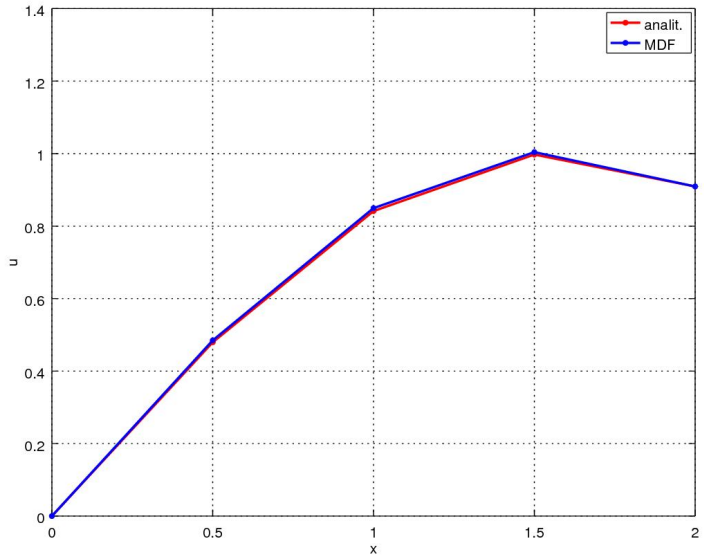

Figura 12.1: Resultado referente ao Exemplo 12.1.1.

A solução analítica deste problema é $u(x) = \text{sen}(x)$. Agora, usando a abordagem pelo método de diferenças finitas abordado nesta seção, obtemos o seguinte problema discreto

$$u_1 = 0, \tag{12.37}$$

$$-\frac{1}{h^2}u_{i-1} + \frac{2}{h^2}u_i - \frac{1}{h^2}u_{i+1} = \text{sen}(x_i),\ i = 2, \ldots, n-1, \tag{12.38}$$

$$u_n = \text{sen}(2), \tag{12.39}$$

onde $h = \pi/(n-1)$ e $x_i = (i-1)h$.

h	n	$\|\tilde{u} - u\|_{L^2}$
10^{-1}	21	1,0e−03
10^{-2}	201	3,3e−05
10^{-3}	2001	1,0e−06

Tabela 12.1: Resultados referentes ao Exemplo 12.1.1.

Resolvendo este sistema com $h = 0{,}5$ obtemos a solução numérica apresentada na Figura 12.1. Ainda, na Tabela 12.1 temos a comparação na

norma L^2 da solução numérica $\tilde{u} = (u_1, u_2, \ldots, u_n)$ com a solução analítica $u(x) = \text{sen}(x)$ para diferentes escolhas de h.

No `GNU Octave`, podemos computar os resultados discutidos neste exemplo com o seguinte código:

```
#param
n = 5;
h = 2/(n-1);

#fonte
f = @(x) sin(x);

#nodos
x = linspace(0,2,n)';

#sist. MDF
A = zeros(n,n);
b = zeros(n,1);

A(1,1) = 1;
b(1)=0;
for i=2:n-1
  A(i,i-1)=-1/h^2;
  A(i,i)=2/h^2;
  A(i,i+1)=-1/h^2;
  b(i)=sin(x(i));
endfor
A(n,n)=1;
b(n)=sin(2);

#sol MDF
u = A\b;

#sol. analic.
ua = @(x) sin(x);

#grafico comparativo
plot(x,ua(x),'r.-',...
```

```
     x,u,'b.-');grid
legend("analit.","MDF")
xlabel("x")
ylabel("u")

#erro na norma L2
printf("%1.1E %1.1E\n", h,norm(u-ua(x)))
```

Exercícios

Exercício 12.1.1. Considere o seguinte problema de valor inicial

$$-u'' + u' = f(x), \ -1 < x < 1, \tag{12.40}$$
$$u(-1) = 0, \tag{12.41}$$
$$u'(1) = 0, \tag{12.42}$$

onde

$$f(x) = \begin{cases} 1 & , x \leq 0 \\ 0 & , x > 0 \end{cases} \tag{12.43}$$

Use uma aproximação adequada pelo método de diferenças finitas para obter o valor aproximado de $u(0)$ com precisão de 2 dígitos significativos.

Capítulo 13

Equações Diferenciais Parciais

Neste capítulo, discutimos alguns tópicos fundamentais da aplicação do método de diferenças finitas para a simulação (aproximação da solução) de equações diferenciais parciais.

13.1 Equação de Poisson

A equação de Poisson em um domínio retangular $D = (x_{\text{ini}}, x_{\text{fin}}) \times (y_{\text{ini}}, y_{\text{fin}})$ com condições de contorno de Dirichlet homogêneas refere-se o seguinte problema

$$u_{xx} + u_{yy} = f(x,y),\ (x,y) \in D, \tag{13.1}$$
$$u(x_{\text{ini}}, y) = 0,\ y_{\text{ini}} \leq y \leq y_{\text{fin}}, \tag{13.2}$$
$$u(x_{\text{fin}}, y) = 0,\ y_{\text{ini}} \leq y \leq y_{\text{fin}}, \tag{13.3}$$
$$u(x, y_{\text{ini}}) = 0,\ x_{\text{ini}} \leq x \leq x_{\text{fin}}, \tag{13.4}$$
$$u(x, y_{\text{fin}}) = 0,,\ x_{\text{ini}} \leq x \leq x_{\text{fin}}, \tag{13.5}$$

onde $u = u(x,y)$ é a incógnita.

A aplicação do método de diferenças finitas para resolver este problema consiste dos mesmos passos usados para resolver problemas de valores de contorno (veja Capítulo 12), a saber: 1. construção da malha, 2. discretização das equações, 3. resolução do problema discreto.

1. Construção da malha

Tratando-se do domínio retangular $\overline{D} = [x_{\text{ini}}, x_{\text{fin}}] \times [y_{\text{ini}}, y_{\text{fin}}]$, podemos construir uma malha do produto cartesiano de partições uniformes dos intervalos $[x_{\text{ini}}, x_{\text{fin}}]$ e $[y_{\text{ini}}, y_{\text{fin}}]$. Mais explicitamente, tomamos

$$x_i := x_{\text{ini}} + (i-1)h_x, \quad h_x = \frac{x_{\text{fin}} - x_{\text{ini}}}{n_x - 1}, \tag{13.6}$$

$$y_j := y_{\text{ini}} + (j-1)h_y, \quad h_y = \frac{y_{\text{fin}} - y_{\text{ini}}}{n_y - 1}, \tag{13.7}$$

onde $i = 1, 2, \ldots, n_x$, $j = 1, 2, \ldots, n_y$, sendo n_x e n_y o número de subintervalos escolhidos para as partições em x e y, respectivamente.

O produto cartesiano das partições em x e y nos fornece uma partição do domínio $\overline{D}$ da forma

$$P(\overline{D}) = \{(x_1, y_1), (x_1, y_2), \ldots, (x_i, y_j), \ldots, (x_{n_x}, y_{n_y})\}, \tag{13.8}$$

cujos nodos (x_i, y_j) podem ser indexados (enumerados) por $k = j + (i-1)n_x$. Por simplicidade, no decorrer do texto, assumiremos $n_x = n_y =: n$ e, por conseguinte, $h_x = h_y = h$.

2. Discretização das equações

Usando a fórmula de diferenças finitas central de ordem h^2 para a segunda derivada, temos

$$u_{xx}(x,y) = \frac{u(x+h,y) - 2u(x,y) + u(x-h,y)}{h^2} + O(h^2), \tag{13.9}$$

$$u_{yy}(x,y) = \frac{u(x,y+h) - 2u(x,y) + u(x,y-h)}{h^2} + O(h^2). \tag{13.10}$$

Daí, denotando $u_{ij} \approx u(x_i, y_j)$ temos

$$u_{xx}(x_i,y_j) = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + O(h^2), \tag{13.11}$$

$$u_{yy}(x_i,y_j) = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} + O(h^2). \tag{13.12}$$

Então, da equação 13.1 temos

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} + O(h^2) = f(x_i,y_j). \tag{13.13}$$

Adora, denotando $u_k := u_{j+(i-1)n}$, desprezando o termo do erro de truncamento e rearranjando os termos nesta última equação temos

$$\frac{1}{h^2}u_{k-n} + \frac{1}{h^2}u_{k-1} - \frac{4}{h^2}u_k + \frac{1}{h^2}u_{k+1} + \frac{1}{h^2}u_{k+n} = f(x_i,y_j), \tag{13.14}$$

para $i,j = 2,3,\ldots,n-1$. Isto é, esta última expressão nos fornece um sistema de $(n-2)^2$ equações para n^2 incógnitas u_k.

Para fechar o sistema, usamos as condições de contorno (13.2)-(13.5):

$$u_{1,j} = 0, \quad u_{n,j} = 0, \tag{13.15}$$
$$u_{i,1} = 0, \quad u_{i,n} = 0, \tag{13.16}$$

$i,j = 1,2,\ldots,n$.

Com isso, o problema discreto obtido da aplicação do método de diferenças finitas consiste no sistema linear de n^2 equações (13.14)-e(13.16) para as n^2 incógnitas u_k, $k = 1,2,\ldots,n^2$.

3. Resolução do problema discreto

O problema discreto (13.14)-(13.16) pode ser escrito na forma matricial

$$A\tilde{u} = b, \tag{13.17}$$

onde o vetor da incógnitas é $\tilde{u} = (u_1, u_2, \ldots, u_{n^2})$ e o vetor dos termos contantes b é tal que

$$i = 1,n,\ j = 1,2,\ldots,n: \ b_k = 0, \tag{13.18}$$
$$i = 1,2,\ldots,n,\ j = 1,n: \ b_k = 0, \tag{13.19}$$
$$i,j = 2,3,\ldots,n-1: \ b_k = f(x_i,y_j). \tag{13.20}$$

Além disso, a matriz dos coeficientes A é tal que

$$i = 1,n,\ j = 1,2,\ldots,n: \ a_{k,k} = 1, \tag{13.21}$$
$$i = 1,2,\ldots,n,\ j = 1,n: \ a_{k,k} = 1, \tag{13.22}$$
$$i,j = 2,3,\ldots,n-1: \ a(k,k-n) = \frac{1}{h^2}, \tag{13.23}$$
$$a(k,k-1) = \frac{1}{h^2}, \tag{13.24}$$

$$a(k,k) = -\frac{4}{h^2}, \tag{13.25}$$

$$a(k,k+1) = \frac{1}{h^2}, \tag{13.26}$$

$$a(k,k+n) = \frac{1}{h^2}. \tag{13.27}$$

Assim sendo, basta empregarmos um método apropriado para resolver o sistema linear (13.17) para obter a solução aproximada de u nos nodos (x_i, y_j).

Exemplo 13.1.1. Consideremos o seguinte problema

$$u_{xx} + u_{yy} = -\operatorname{sen}(x)\operatorname{sen}(y),\ (x,y) \in (0,\pi)\times(0,\pi), \tag{13.28}$$

$$u(0,y) = 0,\ y \in [0,\pi], \tag{13.29}$$

$$u(\pi,y) = 0,\ y \in [0,\pi], \tag{13.30}$$

$$u(x,0) = 0,\ x \in [0,\pi], \tag{13.31}$$

$$u(x,\pi) = 0,\ x \in [0,\pi]. \tag{13.32}$$

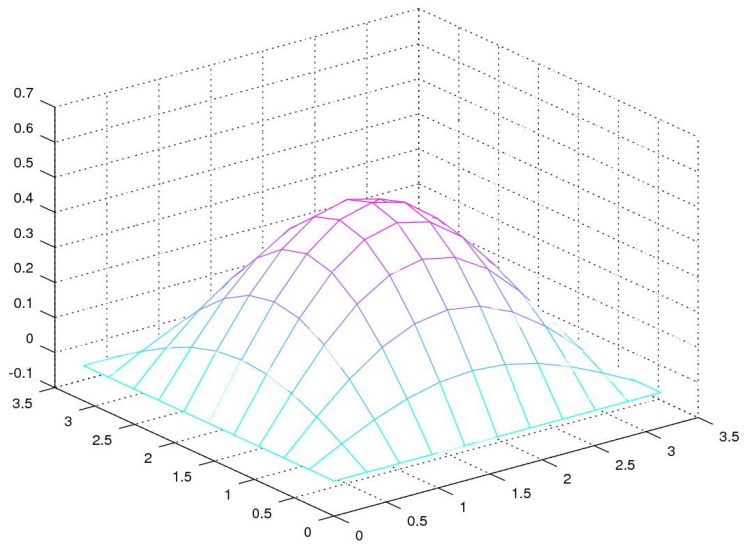

Figura 13.1: Resultado referente ao Exemplo 13.1.1.

A Figura 13.1 mostra um esboço do gráfico da solução aproximada obtida pelo método de diferenças finitas apresentado acima (equações (13.14)-(13.16)) com $n = 11$, i.e. $h = \pi/10$.

n	$\|\tilde{u}-u\|_{L^2}$
6	4,2e−2
11	2,1e−2
21	1,0e−2
41	5,1e−3
81	2,6e−3

Tabela 13.1: Resultados referentes ao Exemplo 13.1.1.

Na Tabela 13.1 temos a norma L^2 da diferença entre a solução aproximada $\tilde{u}$ e a solução analítica $u(x,y) = 0{,}5\,\mathrm{sen}(x)\,\mathrm{sen}(y)$ nos pontos de malha computados com diferentes escolhas de n.

Os resultados obtidos neste exemplo podem ser obtidos no `GNU Octave` com o seguinte código:

```
#params
n=11;
h=pi/(n-1);

#fonte
f = @(x,y) -sin(x).*sin(y);

#malha
x = linspace(0,pi,n);
y = linspace(0,pi,n);

#sistema MDF
A = sparse(n*n,n*n);
b = zeros(n*n,1);

#cc x=0 e x=pi
for i=[1,n]
  for j=1:n
    k = i + (j-1)*n;
    A(k,k)=1;
    b(k) = 0;
  endfor
```

```
endfor

#cc y=0, y=pi
for j=[1,n]
  for i=1:n
    k = i + (j-1)*n;
    A(k,k)=1;
    b(k) = 0;
  endfor
endfor

#nodos internos
for i=2:n-1
  for j=2:n-1
    k = i + (j-1)*n;
    A(k,k-n) = 1/h^2;
    A(k,k-1) = 1/h^2;
    A(k,k) = -4/h^2;
    A(k,k+1) = 1/h^2;
    A(k,k+n) = 1/h^2;

    b(k) = f(x(i),y(j));
  endfor
endfor

u = A\b;

#visu
z = zeros(n,n);
for i=1:n
  for j=1:n
    k = i + (j-1)*n;
    z(i,j) = u(k);
  endfor
endfor
colormap("cool")
mesh(x,y,z)
```

```
ua = zeros(n*n,1);
for i=1:n
  for j=1:n
    k=i+(j-1)*n;
    ua(k) = 0.5*sin(x(i))*sin(y(j));
  endfor
endfor
printf("%d %1.5E %1.1E\n",n,h,norm(u-ua))
```

Exercícios

Exercício 13.1.1.

$$-(u_{xx} + u_{yy}) = f(x),\ (x,y) \in (0,1)^2, \tag{13.33}$$
$$u(0,y) = 0,\ y \in [0,1], \tag{13.34}$$
$$u(1,y) = 0,\ y \in [0,1], \tag{13.35}$$
$$\left.\frac{\partial u}{\partial y}\right|_{y=0} = 0,\ x \in [0,1], \tag{13.36}$$
$$u(x,1) = 0,\ x \in [0,1]. \tag{13.37}$$

onde

$$f(x) = \begin{cases} 1 & ,x \leq 0{,}5 \\ 0 & ,x > 0{,}5 \end{cases} \tag{13.38}$$

Use uma aproximação adequada pelo método de diferenças finitas para obter o valor aproximado de $u(0{,}5,\ 0{,}5)$ com precisão de 2 dígitos significativos.

13.2 Equação do calor

A equação do calor definida em $D = (x_{\text{ini}}, x_{\text{fin}})$ com condição inicial dada e condições de contorno de Dirichlet homogêneas refere-se o seguinte problema

$$u_t - \alpha u_{xx} = f(t,x),\ t > t_0,\ x \in D, \tag{13.39}$$
$$u(t_0,x) = u_0(x),\ x \in D, \tag{13.40}$$
$$u(t, x_{\text{ini}}) = 0,\ t > t_0, \tag{13.41}$$

$$u(t, x_{\text{fin}}) = 0,\ t > t_0 \tag{13.42}$$

onde $u = u(t,x)$ é a incógnita.

O problema acima é um problema de valor inicial com condições de contorno. Uma das estratégias numéricas de solução é o chamado método de Rothe, o qual trata separadamente as discretizações espacial e temporal. Aqui, vamos começar pela discretização espacial e, então, trataremos a discretização temporal.

Discretização espacial

Na discretização espacial, aplicaremos o método de diferenças finitas. Começamos considerando uma partição do domínio $P(\overline{D}) = \{x_1, x_2, \ldots, x_n\}$ com pontos $x_i = x_{\text{ini}} + (i-1)h$ igualmente espaçados por $h = (x_{\text{fin}} - x_{\text{ini}})$. Então, denotando $u_i = u_i(t) \approx u(t,x_i)$ e usando da fórmula de diferenças finitas central de ordem h^2 para as derivadas segundas na equação (13.39), temos

$$\frac{\mathrm{d}}{\mathrm{d}t}u_i - \alpha\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} = f(t,x_i), \tag{13.43}$$

para $i = 2, 3, \ldots, n-1$. Agora, das condições de contorno, temos $u_1 = 0$ e $u_n = 0$, donde obtemos o seguinte sistema de equações diferenciais ordinárias

$$\frac{\mathrm{d}}{\mathrm{d}t}u_2 = -\frac{2\alpha}{h^2}u_2 + \frac{\alpha}{h^2}u_3 + f(t,x_2), \tag{13.44}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}u_i = \frac{\alpha}{h^2}u_{i-1} - \frac{2\alpha}{h^2}u_i + \frac{\alpha}{h^2}u_{i+1} + f(t,x_i), \tag{13.45}$$

$$\frac{\mathrm{d}}{\mathrm{d}t}u_{n-1} = \frac{\alpha}{h^2}u_{n-2} - \frac{2\alpha}{h^2}u_{n-1} + f(t,x_{n-1}), \tag{13.46}$$

$$\tag{13.47}$$

onde $i = 3, 4, \ldots, n-2$ e com condições iniciais dadas por (13.40), i.e.

$$u_j(t_0) = u_0(x),\ j = 2, 3, \ldots, n-1. \tag{13.48}$$

Ainda, observamos que o sistema (13.44) pode ser escrito de forma mais compacta como

$$\frac{\mathrm{d}\tilde{u}}{\mathrm{d}t} = A\tilde{u} + \tilde{f}, \tag{13.49}$$

onde $\tilde{u}(t) = (u_2(t), u_3(t), \ldots, u_{n-1}(t))$, $\tilde{f}(t) = (f(t,x_2), f(t,x_3), \ldots, f(t,x_{n-1}))$ e A é uma matriz $(n-2) \times (n-2)$ da forma

$$A = \begin{bmatrix} -\frac{2\alpha}{h^2} & \frac{\alpha}{h^2} & 0 & 0 & 0 & \cdots & 0 & 0 \\ \frac{\alpha}{h^2} & -\frac{2\alpha}{h^2} & \frac{\alpha}{h^2} & 0 & 0 & \cdots & 0 & 0 \\ 0 & \frac{\alpha}{h^2} & -\frac{2\alpha}{h^2} & \frac{\alpha}{h^2} & 0 & \cdots & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \cdots & 0 & 0 \\ 0 & 0 & & 0 & 0 & \cdots & \frac{\alpha}{h^2} & -\frac{2\alpha}{h^2} \end{bmatrix}. \tag{13.50}$$

Discretização temporal

Aqui, vamos usar o método de Euler (veja, 11.1) para aproximar a solução de (13.50)-(13.48). Para tando, escolhemos um passo de tempo $h_t > 0$ e denotamos $t^{(k)} = t_0 + (k-1)h_t$, $\tilde{u}^{(k)} \approx \tilde{u}(t^{(k)})$ e $\tilde{f}^{(k)} = \tilde{f}(t^{(k)})$. Com isso, a iteração do método de Euler nos fornece

$$\tilde{u}^{(1)} = \tilde{u}_0 \tag{13.51}$$
$$\tilde{u}^{(k+1)} = \tilde{u}^{(k)} + h_t \left(A\tilde{u}^{(k)} + \tilde{f}^{(k)} \right), \tag{13.52}$$

com $k = 1, 2, \ldots$. Equivalentemente, escrevemos

$$\tilde{u}^{(1)} = \tilde{u}_0 \tag{13.53}$$
$$\tilde{u}^{(k+1)} = (I - h_t A)\, \tilde{u}^{(k)} + h_t \tilde{f}^{(k)}. \tag{13.54}$$

Observação 13.2.1. O esquema numérico acima é **condicionalmente estável**. Pode-se mostrar a seguinte condição de estabilidade [2, Cap. 12, Seç. 2]:

$$\alpha \frac{h_t}{h^2} \leq \frac{1}{2}. \tag{13.55}$$

Exemplo 13.2.1. Consideremos o seguinte problema

$$u_t - u_{xx} = \text{sen}(x),\ t > 0,\ 0 \leq x \leq \pi, \tag{13.56}$$
$$u(0,x) = 0,\ 0 < x < \pi, \tag{13.57}$$
$$u(t,0) = 0,\ t > 0 \tag{13.58}$$
$$u(t,\pi) = 0,\ t > 0. \tag{13.59}$$

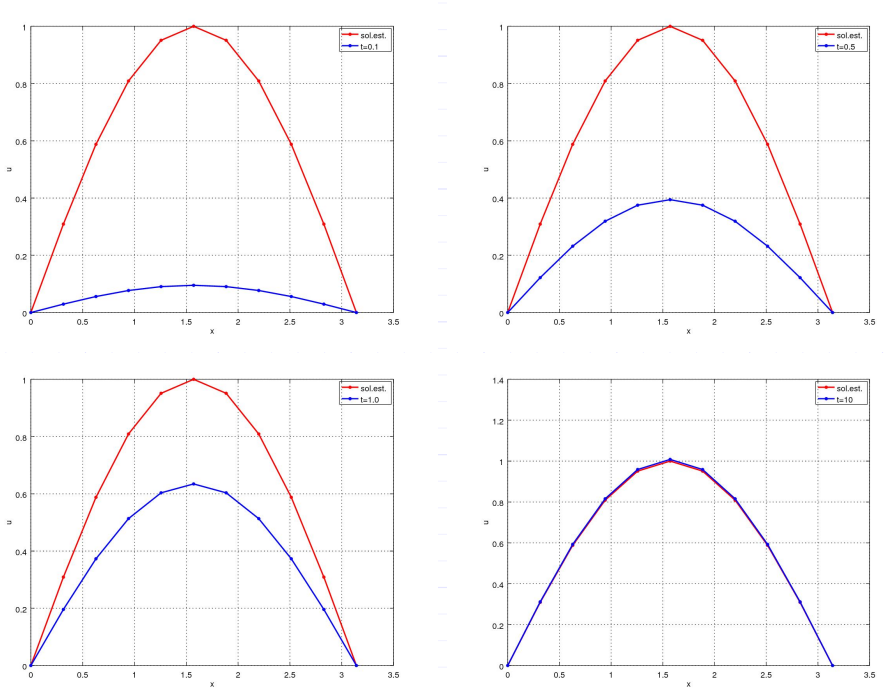

Figura 13.2: Resultados referentes ao Exemplo 13.2.1.

Este problema tem solução estacionário $u(x) = \text{sen}(x)$. Na Figura 13.2, temos o esboço das soluções numéricas em diferentes tempos t usando o esquema numérico acima com $h = 10^{-1}$ e $h_t = 10^{-3}$.

No `GNU Octave`, podemos computar os resultados discutidos neste exemplo com o seguinte código:

```
#params
n=11;
h=pi/(n-1);

tf=1;
ht=10^-3;
nt=round(tf/ht)+1;

#fonte
f = @(x) sin(x);
```

```
#malha
t=[0:ht:(nt-1)*ht]';
x=[0:h:(n-1)*h]';

#matriz MDF
A = sparse(n-2,n-2);
A(1,1)=-2/h^2;
A(1,2)=1/h^2;
for i=2:n-3
  A(i,i-1)=1/h^2;
  A(i,i)=-2/h^2;
  A(i,i+1)=1/h^2;
endfor
A(n-2,n-3)=1/h^2;
A(n-2,n-2)=-2/h^2;

#c.i.
u=zeros(n,1);

#iter. de Euler
for k=1:nt-1
  u(2:n-1)=u(2:n-1)+ht*(A*u(2:n-1)+f(x(2:n-1)));
endfor

#visu
uest = @(x) sin(x);
plot(x,uest(x),'r.-',...
     x,u,'b.-');grid
xlabel('x');
ylabel('u');
legend('sol.est.','sol.num.');
```

Exercícios

Exercício 13.2.1. Considere o seguinte problema

$$u_t - u_{xx} = f(x),\ t > 0,\ 0 \leq x \leq 1, \tag{13.60}$$

$$u(0,x) = 0,\ 0 < x < 1, \tag{13.61}$$
$$u(t,0) = 1,\ t > 0 \tag{13.62}$$
$$u(t,1) = 0,\ t > 0. \tag{13.63}$$

com

$$f(x) = \begin{cases} 1 & ,x \leq 0{,}5, \\ 0 & ,x > 0{,}5 \end{cases} \tag{13.64}$$

Use o método de diferenças finitas para obter uma aproximação de $u(1,0.5)$ com dois dígitos significativos de precisão.

13.3 Equação da onda

A equação da onda definida em $D := (x_{\text{ini}}, x_{\text{fin}})$ com condições iniciais dadas e condições de contorno de Dirichlet homogêneas refere-se o seguinte problema

$$u_{tt} - \alpha u_{xx} = 0,\ t > t_0,\ x \in D, \tag{13.65}$$
$$u(x,t_0) = f(x),\ x \in D, \tag{13.66}$$
$$\frac{\partial u}{\partial t}(x,t_0) = g(x),\ x \in D, \tag{13.67}$$
$$u(x_{\text{ini}},t) = 0,\ t > t_0, \tag{13.68}$$
$$u(x_{\text{fin}},t) = 0,\ t > t_0 \tag{13.69}$$

onde $u = u(x,t)$ é a incógnita.

Aqui, para aplicarmos o método de diferenças finitas, vamos escolher os tempos $t^{(j)} = t_0 + (j-1)h_t$, $j = 1, 2, \ldots, n_t$, com passo temporal $h_t > 0$, e os pontos $x_i = x_{\text{ini}} + (i-1)h_x$, $i = 1, 2, \ldots, n_x$, com passo no espaço espacial $h_x = (x_{\text{fin}} - x_{\text{ini}})/(n_x - 1)$.

Da escolha das discretizações temporal e espacial, podemos usar a fórmula de diferenças finitas de ordem 2 para discretizarmos a equação (13.65). Para tanto, denotamos $u_{i,j} \approx u(x_i,t_j)$ e de (13.65) temos

$$\frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h_t^2} - \alpha \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h_x^2} = 0, \tag{13.70}$$

para $j = 2, 3, \ldots, n_t - 1$ e $i = 2, 3, \ldots, n_x - 1$. Rearranjando os termos, temos

$$u_{i,j+1} = \alpha \frac{h_t^2}{h_x^2} u_{i-1,j} + 2\left(1 - \alpha \frac{h_t^2}{h_x^2}\right) u_{i,j} + \alpha \frac{h_t^2}{h_x^2} u_{i+1,j} - u_{i,j-1}, \tag{13.71}$$

para $j = 2, 3, \ldots, n_t - 1$ e $i = 2, 3, \ldots, n_x - 1$.

Agora, das condições de contorno (13.68) e (13.69), temos $u_{1,j} = u_{n_x,j} = 0$, $j = 2, 3, \ldots, n_t$. Com isso, o sistema (13.71) torna-se

$$u_{2,j+1} = 2\left(1 - \alpha \frac{h_t^2}{h_x^2}\right) u_{2,j} + \alpha \frac{h_t^2}{h_x^2} u_{3,j} - u_{2,j-1}, \tag{13.72}$$

$$u_{i,j+1} = \alpha \frac{h_t^2}{h_x^2} u_{i-1,j} + 2\left(1 - \alpha \frac{h_t^2}{h_x^2}\right) u_{i,j} + \alpha \frac{h_t^2}{h_x^2} u_{i+1,j} - u_{i,j-1}, \tag{13.73}$$

$$u_{n_x-1,j+1} = \alpha \frac{h_t^2}{h_x^2} u_{n_x-2,j} + 2\left(1 - \alpha \frac{h_t^2}{h_x^2}\right) u_{n_x-1,j} - u_{i,j-1}, \tag{13.74}$$

$$\tag{13.75}$$

para $i = 3, 4, \ldots, n_x$ e $j = 2, 3, \ldots, n_t$. Este sistema de equações pode ser escrita na seguinte forma matricial

$$\begin{bmatrix} u_{2,j+1} \\ u_{3,j+1} \\ \vdots \\ u_{n_x-1,j+1} \end{bmatrix} = \begin{bmatrix} 2(1-\lambda) & \lambda & 0 & \cdots & 0 \\ \lambda & 2(1-\lambda) & \lambda & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \cdots \\ 0 & 0 & \cdots & \lambda & 2(1-\lambda) \end{bmatrix} \begin{bmatrix} u_{2,j} \\ u_{3,j} \\ \vdots \\ u_{n_x-1,j} \end{bmatrix} - \begin{bmatrix} u_{2,j-1} \\ u_{3,j-1} \\ \vdots \\ u_{n_x-1,j-1} \end{bmatrix}, \tag{13.76}$$

para $j = 2, 3, \cdots, n_t - 1$, onde $\lambda := \alpha h_t^2 / h_x^2$.

Esta última equação (13.76) nos permite computar iterativamente a aproximação $u_{i,j+1}$ a partir das aproximações $u_{i,j}$ e $u_{i,j-1}$. Para inicializar as iterações, precisamos de $u_{i,1}$ e $u_{i,2}$, $i = 2, 3, \ldots, n_x$. A primeira é dada pela condição inicial (13.66), da qual temos

$$u_{i,1} = f(x_i), \ i = 2, 3, \ldots, n_t. \tag{13.77}$$

Agora, usando a fórmula de diferenças finitas progressiva de ordem 1 na condições inicial (13.67), obtemos

$$u_{i,2} = u_{i,1} + h_t g(x_i),\ i = 2, 3, \ldots, n_t. \tag{13.78}$$

Com tudo isso, observamos que as equações (13.77), (13.78) e (13.76), nesta ordem, nos fornece um algoritmo iterativo no tempo para computar as aproximações da solução u.

Observação 13.3.1. Pode-se mostrar a seguinte condição de estabilidade

$$\alpha \frac{h_t^2}{h_x^2} \leq 1. \tag{13.79}$$

Exemplo 13.3.1. Consideremos o seguinte problema

$$u_{tt} - u_{xx} = 0,\ t > 0,\ 0 < x < 1, \tag{13.80}$$
$$u(0,x) = x(1-x),\ 0 < x < 1, \tag{13.81}$$
$$u_t(0,x) = 0,\ 0 < x < 1, \tag{13.82}$$
$$u(t,0) = 0,\ t > 0 \tag{13.83}$$
$$u(t,\pi) = 0,\ t > 0. \tag{13.84}$$

Na Figura 13.3, temos o esboço das soluções numéricas em diferentes tempos t usando o esquema numérico acima com $h_t = 10^{-2}$ e $h_x = 10^{-1}$.

No `GNU Octave`, podemos computar os resultados discutidos neste exemplo com o seguinte código:

```
#params
nx=11;
hx=1/(nx-1);

tf=1;
ht=10^-2;
nt=round(tf/ht)+1;

lambda = ht^2/hx^2;
```

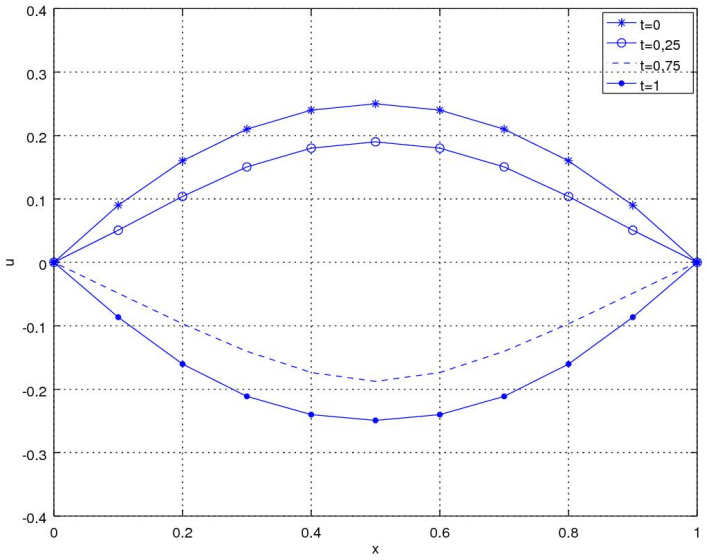

Figura 13.3: Resultados referentes ao Exemplo 13.3.1.

```
#malha
t=[0:ht:(nt-1)*ht]';
x=[0:hx:(nx-1)*hx]';

#u
u0=zeros(nx,1);
u1=zeros(nx,1);
u=zeros(nx,1);

#c.i. 1
for i=2:nx-1
  u0(i)=x(i)*(1-x(i));
endfor

#c.i. 2
u1=zeros(nx,1);
for i=2:nx-1
  u1(i)=u0(i)+ht*0;
```

```
endfor

#matriz MDF
A = sparse(nx-2,nx-2);
A(1,1)=2*(1-lambda);
A(1,2)=lambda;
for i=2:nx-3
  A(i,i-1)=lambda;
  A(i,i)=2*(1-lambda);
  A(i,i+1)=lambda;
endfor
A(nx-2,nx-3)=lambda;
A(nx-2,nx-2)=2*(1-lambda);

#iteracoes
for k=2:nt-1
  u(2:nx-1)=A*u1(2:nx-1) - u0(2:nx-1);
  u0=u1;
  u1=u;
endfor

#visu
plot(x,u1,'b-');grid
xlabel('x');
ylabel('u');
```

Exercício

Exercício 13.3.1. Considere o seguinte problema

$$u_{tt} - u_{xx} = 0,\ t > 0,\ 0 < x < 1, \tag{13.85}$$
$$u(0,x) = x(1-x),\ 0 < x < 1, \tag{13.86}$$
$$u_t(0,x) = 1,\ 0 < x < 1, \tag{13.87}$$
$$u(t,0) = 0,\ t > 0 \tag{13.88}$$
$$u(t,\pi) = 0,\ t > 0. \tag{13.89}$$

Use o método de diferenças finitas para obter uma aproximação de $u(0{,}75,\ 1)$ com dois dígitos significativos de precisão.

Resposta dos Exercícios

Exercício 1.1.1. Código. a) 45,15625; b) 11,25; c) 55981; d) $0,\overline{3}$; e) $0,\overline{3}$

Exercício 1.1.2. Código. a) $(101101,1)_2$; b) $(0,1\overline{03})_4$

Exercício 1.1.3. Código. a) $(231,022)_4$; b) $(1011,01)_2$

Exercício 1.2.1. Código. a) [11110000]; b) [10001111]
c) [00000100]; d) [00000111]

Exercício 1.2.2. Código. a) [0000000000100000];
b) [0000000000111111];

Exercício 1.2.3. [10111 ... 11] ~ -3

Exercício 1.2.4. Código. a) [000...010|111...10|1];
b) [000...01|000...01|0]

Exercício 1.2.5. Código. a) 7; b) `NaN`

Exercício 1.3.1. Código. a) $2{,}72 \times 10^3$; b) $2{,}718 \times 10^3$

Exercício 1.4.1. Código. a) $1{,}593 \times 10^{-3}$; b) $2{,}818 \times 10^{-1}$;

Exercício 1.4.2. Código. a) 0,051%; b) 0,01%;

Exercício 1.4.3. Código. a) 3; b) 3

Exercício 1.4.4. $1/6! \approx 1{,}4 \times 10^{-3}$.

Exercício 1.5.1. Código. $2{,}002083 \times 10^{-3}$

Exercício 1.5.2. Código. $5{,}75403 \times 10^{-3}$

Exercício 1.5.3. Código. $5{,}0 \times 10^{-10}$

Exercício 2.1.1. Código. $9{,}179688 \times 10^{-1}$

Exercício 2.1.2. Código. 9,15833e−1

Exercício 2.1.3. Código. $5{,}770508 \times 10^{-1}$

Exercício 2.2.1. Código. $9{,}158079 \times 10^{-1}$

Exercício 2.2.2. Código. $5{,}76984 \times 10^{-1}$

Exercício 2.3.1. Código. $\alpha = 0{,}6$; $7{,}3909 \times 10^{-1}$

Exercício 2.4.1. Código. $9{,}15811 \times 10^{-1}$

Exercício 2.5.1. Código. $9{,}15811 \times 10^{-1}$

Exercício 2.5.2. Código. $5{,}7700 \times 10^{-1}$

Exercício 2.6.1. Código. $9{,}1581 \times 10^{-1}$

Exercício 2.6.2. Código. $5{,}7700 \times 10^{-1}$

Exercício 2.7.1. Código. 2

Exercício 3.1.1. Código.

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & -3 \\ 0 & 0 & 1 & 0 & 0 & -2 \\ 0 & 0 & 0 & 1 & 0 & 2 \\ 0 & 0 & 0 & 0 & 1 & -4 \end{bmatrix}$$

Exercício 3.1.2. Código.

$$\begin{bmatrix} 1.0000 & 0.0000 & 0.0000 & -3.9435\text{e}{-1} \\ -0.0000 & 1.0000 & -0.0000 & -2.3179\text{e}{-1} \\ 0.0000 & 0.0000 & 1.0000 & 1.2120\text{e}{+0} \end{bmatrix}$$

Exercício 3.2.1. Código. a) 2,2383; b) 2,0323e+1; c) 3,5128e+1

Exercício 3.3.1. Código.

$$\begin{bmatrix} 1{,}0000 & 0{,}0000 & 0{,}0000 & 6{,}2588\text{e}{-1} \\ 0{,}0000 & 1{,}0000 & 0{,}0000 & -1{,}6777\text{e}{+0} \\ 0{,}0000 & 0{,}0000 & 1{,}0000 & 6{,}2589\text{e}{+4} \end{bmatrix}$$

Exercício 3.4.1. Código. $x_1 = -3$, $x_2 = -1$, $x_3 = 1$

Exercício 4.1.1. Código. $x^{(5)} = (-1{,}00256,\ 2{,}95365,\ -1{,}95347,\ 0{,}97913)$; $\|Ax^{(5)} - b\| = 0{,}42244$

Exercício 4.1.2. Código. $x^{(5)} = (-1{,}00423,\ 3{,}00316,\ -2{,}00401,\ 0{,}99852)$; $\|Ax^{(5)} - b\| = 0{,}025883$

Exercício 5.1.1. Código. $x_1 = 1{,}7519\text{e}{+0}$, $x_2 = -2{,}6202\text{e}{-1}$, $x_3 = -1{,}8983\text{e}{+0}$

Exercício 6.1.1. Código. $0{,}58\bar{3}x^2 - 1{,}25x + 0{,}1\bar{6} + 1$.

Exercício 6.2.1. Código. $0{,}54x^3 - 0{,}15x^2 + 1{,}3x + 1$.

Exercício 6.3.1. Código. $0{,}54x^3 - 0{,}15x^2 + 1{,}3x + 1$.

Exercício 7.1.1. Código. $c_1 = -1{,}3259$, $c_2 = 8{,}66071\text{e}{-2}$, $\|r(c)\|_2 = 1{,}01390$.

Exercício 7.1.2. Código. $c_1 = -4{,}50361\text{e}{-1}$, $c_2 = -2{,}78350\text{e}{-1}$, $c_3 = 1{,}46291$, $c_4 = 2{,}09648$, $\|r(c)\|_2 = 5{,}71346$

Exercício 7.1.3. Código. $c_1 = -2{,}76842$, $c_2 = -7{,}17935\text{e}{-1}$, $c_3 = 1{,}37014\text{e}{-1}$, $\|r(c)\|_2 = 2{,}48880\text{e}{+1}$

Exercício 7.1.4. Código. $c_1 = 2{,}10131\text{e}{+0}$, $c_2 = -9{,}73859\text{e}{-1}$, $c_3 = 1.25521\text{e}{+0}$

Exercício 7.2.1. Código. a) $c_1 = 2{,}69971\text{e}{+0}$, $c_2 = -1{,}44723\text{e}{+0}$, $c_3 = 1.24333\text{e}{+0}$; b) divergente.

Exercício 7.2.2. Código. a) $c_1 = 2{,}69971\text{e}{+0}$, $c_2 = -1{,}44723\text{e}{+0}$, $c_3 = 1.24333\text{e}{+0}$; b) $c_1 = 2{,}69971\text{e}{+0}$, $c_2 = -1{,}44723\text{e}{+0}$, $c_3 = 1.24333\text{e}{+0}$

Exercício 8.1.1. Código. a) $D_{+,h}f(2{,}5) = 1{,}05949$; b) $D_{-,h}f(2{,}5) = 1{,}05877$; c) $D_{0,h^2}f(2{,}5) = 1{,}05913$;

Exercício 8.1.2. Código.

i	1	2	3	4	5	6
dy/dx	4,0e−1	7,5e−1	1,3e+0	1,1e+0	7,5e−1	8,0e−1

Exercício 8.2.1. Código. a) 7,25162e−2; b) 7.24701e−2; c) 7,24696e−2; d) 7,24696e−2; $h = 10^{-2}$;

Exercício 8.2.2. Código. 4,0;

Exercício 8.3.1. Código. 1,05913

Exercício 8.3.2. Código.

a) $\frac{1}{12h}\left[3f(x-4h) - 16f(x-3h) + 36f(x-2h) - 48f(x-h) + 25f(x)\right]$
b) $\frac{1}{12h}\left[-f(x-3h) + 6f(x-2h) - 18f(x-h) + 10f(x) + 3f(x+h)\right]$
c) $\frac{1}{12h}\left[f(x-2h) - 8f(x-h) + 8f(x+h) - f(x+2h)\right]$
d) $\frac{1}{12h}\left[-3f(x-h) - 10f(x) + 18f(x+h) - 6f(x+2h) + f(x+3h)\right]$
d) $\frac{1}{12h}\left[-25f(x) + 48f(x+h) - 36f(x+2h) + 16f(x+3h) - 3f(x+4h)\right]$

Exercício 8.3.3. Código.

i	1	2	3	4	5	6
dy/dx	1,7500e−1	7,2500e−1	1,4250e+0	1,1250e+0	4,2500e−1	1,6750e+0

Exercício 9.1.2. Código. a) 1,05919; b) 1,05916; c) 1,05913

Exercício 10.1.1. Código. a) 3,33647e−1; b) 1,71368e−1; c) 2,79554e−1

Exercício 10.1.2. Código. a) 4,02000e−1; b) 1,04250E+0; c) 8,08667e−1

Exercício 10.1.3. Use um procedimento semelhante aquele usado para determinar a ordem do erro de truncamento da regra de Simpson.

Exercício 10.1.4.

$$\int_a^b f(x)\,dx = \frac{3h}{2}\left[f\left(a + \frac{1}{3}(b-a)\right)\right. \tag{10.44}$$

$$\left. + f\left(a + \frac{2}{3}(b-a)\right)\right] + O(h^3),\ h = \frac{(b-a)}{3} \tag{10.45}$$

Exercício 10.2.1. Código. a) 2,69264e−1; b) 2,68282e−1; c) 2,68937e−1

Exercício 10.2.2. Código. a) 8,12000e−1; b) 1,03850; c) 8,11667e−1

Exercício 10.3.1. Código. 2,68953e−1

Exercício 10.4.1. 1

Exercício 10.4.2. $x_1 = 0$, $w_1 = 2$

Exercício 10.5.1. Código. a) −2,61712e−1; b) 2,55351e−1; c) 8,97510e−2; d) 1,27411e−1; e) 1.21016e−1.

Exercício 10.5.2. Código. a) −1,54617e−1; b) −1,50216e−1; c) −1,47026e−1; d) −1,47190e−1; e) −1,47193e−1.

Exercício 10.5.3. Código. a) 1,21016e−1; b) 1,21744e−1; c) 1,21744e−1

Exercício 10.6.1. Código. a) $-2{,}84951E{-}01$; b) 2,66274e−01; c) 1,49496e−01; d) 1,60085e−01; e) 1,59427e−01.

Exercício 10.6.2. Código. a) −1,03618e−1; b) −5,56446e−2; c) −4,19168e−2

Exercício 10.6.3. Código. a) −1,31347; b) −1,23313; c) −1,26007

Exercício 10.7.1. Código. 1,2e−1

Exercício 11.1.1. Código. −5,87722e−1

Exercício 11.2.1. Código. a) −6,00654e−1; b) −6,00703e−1; c) −5,99608e−1

Exercício 11.3.1. Código. −5.99240e−1

Exercício 11.4.1. Código. a) −6,00696e−1; b) −5,96694e−1; c) −5,96161e−1

Exercício 12.1.1. Código. 7,2e−1

Exercício 13.1.1. Código. 2,9e−2

Exercício 13.2.1. Código. 5,6e−1

Exercício 13.3.1. Código. 6,3e−2

Bibliografia

[1] A. Björk. *Numerical methods for least squares problems*. SIAM, 1996.

[2] R. Burden, J. Faires, and A. Burden. *Análise Numérica*. CENGAGE Learning, 10. edition, 2015.

[3] E. Isaacson and H. Keller. *Analysis of numerical methods*. Dover, 1994.

[4] J. Nocedal and S. Wright. *Numerical optimization*. Springer, 2006.

[5] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical recipes*. Cambridge University Press, 3. edition, 2007.

[6] J. Stoer and R. Bulirsch. *Introduction to numerical analysis*. Springer-Verlag, 2. edition, 1993.

Índice