# Matemática numérica

Pedro H A Konzen

9 de agosto de 2018

# Licença

# Prefácio

Nestas notas de aula são abordados temas introdutórios de matemática numérica. Como ferramenta computacional de apoio didático, faço uso de códigos em `GNU Octave` (compatíveis com `MATLAB`). Ressalto que os códigos apresentados são implementações ingênuas com intuito de ressaltar os aspectos fundamentais dos métodos numéricos discutidos no texto.
Agradeço aos(às) estudantes que assiduamente ou esporadicamente contribuem com correções, sugestões e críticas em prol do desenvolvimento deste material didático.

Pedro H A Konzen

# Sumário

# 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

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

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

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

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

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

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 $b_2$.

[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.

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

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:

$$\begin{aligned} I &= 26 && (1.14)\\ &= 6 \times 4 + 2 \times 4^0 && (1.15)\\ &= (1 \times 4 + 2) \times 4 + 2 \times 4^0 && (1.16)\\ &= 1 \times 4^2 + 2 \times 4 + 2 \times 4^0 && (1.17)\\ &= (122)_4. && (1.18) \end{aligned}$$

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 $(11010{,}101)_2 = (122{,}22)_4$.

### 1.1.2 Exercícios

**E 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) $(0{,}\overline{1})_4$

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

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

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

**E 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
```

## 1.3 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

$$M := (1{,}m_1 m_2 m_3 \ldots m_{52})_2, \tag{1.47}$$

$$c := (c_{10} \ldots c_2 c_1 c_0)_2. \tag{1.48}$$

**Exemplo 1.3.1.** 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 $-3{,}25$.

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 exemplo, $x = 0{,}1$ é representado pelo registro

[0101100110011001100110011001100110011001100110011001100110111111100]

o qual é o número $fl(x) \approx 0{,}1000000000000000055511151231 2$. Desta forma, dados $x = 0{,}1$ e $y = 0{,}3$, a computação de $x + y$ em ponto flutuante passa, primeiro, por obtermos as representações $fl(x)$ e $fl(y)$, então computamos a soma $z = fl(x) + fl(y)$ e, por fim, obtemos a representação $fl(z)$, a qual será o valor computado de $x + y$.
Em construção ...

# Capítulo 2

# Aproximação por mínimos quadrados

## 2.1 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

$$f(x;c) = \sum_{j=1}^{m} c_j f_j(x) \tag{2.1}$$

$$= c_1 f_1(x) + c_2 f_2(x) + c_3 f_3(x) + \cdots + c_m f_m(x) \tag{2.2}$$

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

$$\min_c \left\{ E := \sum_{i=1}^{n} (y_i - f(x_i;c))^2 \right\}. \tag{2.3}$$

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

$$\min_c \{E := \|r(c)\|_2^2\}. \tag{2.4}$$

### 2.1.1 Método das equações normais

A fim de resolver o problema de mínimos quadrados (2.4), observamos que o erro quadrático

$$E = \|r(c)\|_2^2 \tag{2.5}$$
$$= \sum_{i=1}^{n} r_i(c)^2 \tag{2.6}$$
$$= \sum_{i=1}^{n} \left(y_i - f(x_i; c)\right)^2 \tag{2.7}$$
$$= \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{m} c_j f_j(x_i)\right)^2 \tag{2.8}$$
$$= \|y - Ac\|_2^2, \tag{2.9}$$

onde $y = (y_1, y_2, \ldots, y_n)$ e

$$A := \begin{bmatrix} f_1(x_1) & f_2(x_1) & \cdots & f_m(x_1) \\ f_1(x_2) & f_2(x_2) & \cdots & f_m(x_2) \\ \vdots & \vdots & \vdots & \vdots \\ f_1(x_n) & f_2(x_n) & \cdots & f_m(x_n) \end{bmatrix}. \tag{2.10}$$

Os parâmetros $c_j$ que minimizam o erro $E$ são solução do seguinte sistema de equações

$$\frac{\partial E}{\partial c_j} = 2 \sum_{i=0}^{n} r_i(c) \frac{\partial}{\partial c_j} r_i(c) = 0, \tag{2.11}$$

onde $j = 1, 2, \ldots, m$. Ou, em uma notação mais apropriada,

$$\nabla_c E = 0 \Leftrightarrow A^T r(c) = 0 \tag{2.12}$$
$$\Leftrightarrow A^T (y - Ac) = 0 \tag{2.13}$$
$$\Leftrightarrow A^T A c = A^T y. \tag{2.14}$$

Portanto, o problema linear de mínimos quadrados se resume em resolver as chamadas **equações normais**

$$A^T A c = A^T y. \tag{2.15}$$

Logo, o problema linear de mínimos quadrados (2.4) reduz-se a resolver o sistema linear (2.15) para $c$. Isto nos leva a questão de verificar se $A^T A$ é invertível. De sorte, da disciplina de álgebra linear temos o seguinte teorema.

**Teorema 2.1.1.** *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

$$c = (A^TA)^{-1}A^Ty. \tag{2.16}$$

**Exemplo 2.1.1.** (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

$$p(x) = p_1x^2 + p_2x + p_n \tag{2.17}$$

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

$$A^TAp = A^Ty, \tag{2.18}$$

onde $y = (y_1, y_2, y_3)$ e

$$A := \begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \\ x_4^2 & x_4 & 1 \end{bmatrix}. \tag{2.19}$$

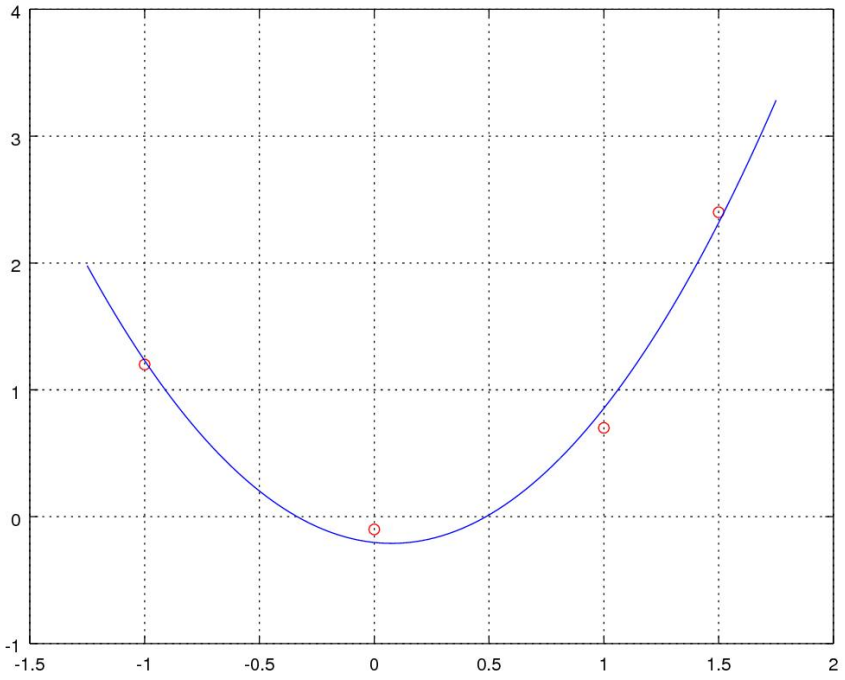


Figura 2.1: Esboço do polinômio ajustado no Exemplo 2.1.1.

Emfim, resolvendo as equações normais (2.18), obtemos

$$p(x) = 1{,}25x^2 - 0{,}188x - 0{,}203. \tag{2.20}$$

A Figura 2.2 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 2.1.2.** (Ajuste de curvas) Consideremos o mesmo conjunto de pontos do exemplo anterior (Exemplo 2.1.1). Aqui, vamos ajustar uma curva da forma

$$f(x) = c_1 \operatorname{sen}(x) + c_2 \cos(x) + c_3 \tag{2.21}$$

no sentido de mínimos quadrados. Para tanto, formamos a matriz

$$A := \begin{bmatrix} \operatorname{sen}(x_1) & \cos(x_1) & 1 \\ \operatorname{sen}(x_2) & \cos(x_2) & 1 \\ \operatorname{sen}(x_3) & \cos(x_3) & 1 \\ \operatorname{sen}(x_4) & \cos(x_4) & 1 \end{bmatrix} \tag{2.22}$$

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 **??** 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)];
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
```

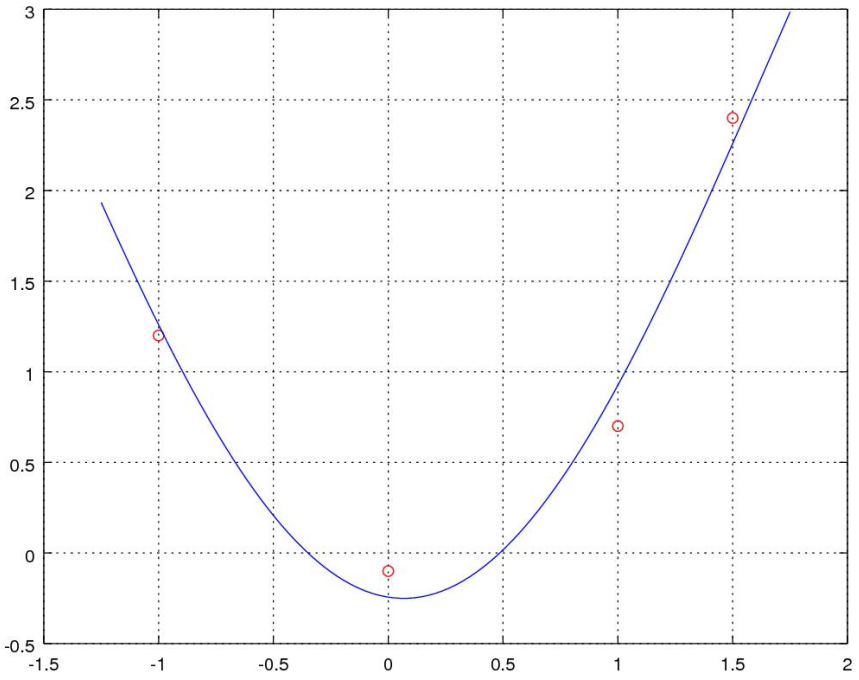


Figura 2.2: Esboço da curva ajustada no Exemplo 2.1.2.

**Exemplo 2.1.3.** (Um problema não linear) Consideremos o problema de ajustar, no sentido de mínimos quadrados, à função

$$f(x) = c_1 e^{c_2 x} \tag{2.23}$$

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

$$y = c_1 e^{c_2 x} \Rightarrow \ln y = \ln c_1 e^{c_2 x} \tag{2.24}$$

$$\Rightarrow \ln y = \ln c_1 + c_2 x. \tag{2.25}$$

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

$$A := \begin{bmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ 1 & x_4 \end{bmatrix} \tag{2.26}$$

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 2.3 mostra um esboço da curva $f(x) = c_1 e^{c_2 x}$ ajustada (linha azul) aos pontos dados (círculos vermelhos).

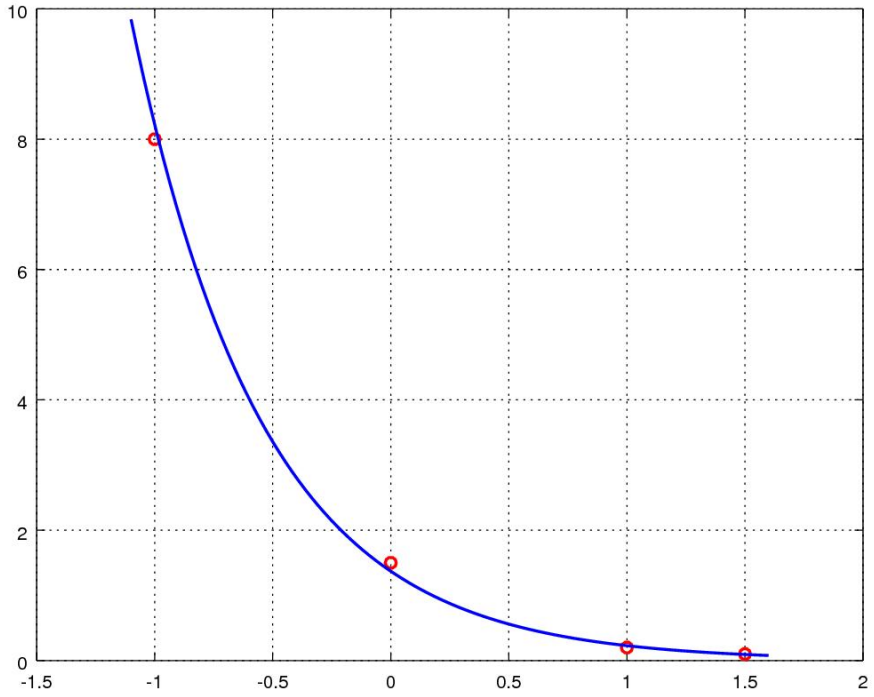


Figura 2.3: Esboço da curva ajustada no Exemplo 2.1.3.

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);

#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

**E 2.1.1.** 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.

**E 2.1.2.** 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. $\|r(c)\|_2$.

**E 2.1.3.** 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. $\|r(c)\|_2$.

**E 2.1.4.** 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 |

## 2.2 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

$$\min_{\{c_1,c_2,\ldots,c_m\}} \left[ E := \sum_{i=1}^{n} \left(y_i - f(x_i; c)\right)^2 \right]. \tag{2.27}$$

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

$$E = \|r(c)\|_2^2. \tag{2.28}$$

Tais parâmetros são solução do seguinte sistema de equações

$$\frac{\partial E}{\partial c_j} = 2 \sum_{i=1}^{n} r_i(c) \frac{\partial}{\partial c_j} r_i(c) = 0 \tag{2.29}$$

ou, equivalentemente, da equação

$$\nabla E = 0 \Leftrightarrow J_R^T(c) r(c) = 0, \tag{2.30}$$

onde

$$J_R(c) := \begin{bmatrix} \frac{\partial r_1}{\partial c_1} & \frac{\partial r_1}{\partial c_2} & \cdots & \frac{\partial r_1}{\partial c_m} \\ \frac{\partial r_2}{\partial c_1} & \frac{\partial r_2}{\partial c_2} & \cdots & \frac{\partial r_2}{\partial c_m} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{\partial r_n}{\partial c_1} & \frac{\partial r_n}{\partial c_2} & \cdots & \frac{\partial r_n}{\partial c_m} \end{bmatrix} \tag{2.31}$$

é a jacobiana do resíduo $r$ em relação aos parâmetros $c$.
Podemos usar o método de Newton para resolver (2.30). Para tanto, escolhemos uma aproximação inicial para $c^{(1)} = (c_1^{(1)}, c_2^{(1)}, \ldots, c_m^{(1)})$ e iteramos

$$H_R(c^{(k)})\delta^{(k)} = -J_R^T(c)r(c) \tag{2.32}$$

$$c^{(k+1)} = c^{(k)} + \delta^{(k)}, \tag{2.33}$$

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

$$h_{p,q} := \sum_{i=1}^{n} \left\{ \frac{\partial r_i}{\partial c_q}\frac{\partial r_i}{\partial c_p} + r_i \frac{\partial^2 r_i}{\partial c_q \partial c_p} \right\}. \tag{2.34}$$

**Exemplo 2.2.1.** Consideremos o problema de ajustar, no sentido de mínimos quadrados, a função

$$f(x; c) = c_1 e^{c_2 x} \tag{2.35}$$

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 (2.32)-(2.33). Para tanto, para cada $i = 1, 2, 3, 4$, precisamos das seguintes derivadas parciais do resíduo $r_i(c) :=$

$y_i - c_1 e^{c_2 x_i}$:

$$\frac{\partial}{\partial c_1} r_i(c) = -e^{c_2 x_i}, \tag{2.36}$$

$$\frac{\partial}{\partial c_2} r_i(c) = -c_1 x_i e^{c_2 x_i}, \tag{2.37}$$

$$\frac{\partial^2}{\partial c_1^2} r_i(c) = 0, \tag{2.38}$$

$$\frac{\partial^2}{\partial c_1 \partial c_2} r_i(c) = \frac{\partial^2}{\partial c_2 \partial c_1} r_i(c) = -x_i e^{c_2 x_i}, \tag{2.39}$$

$$\frac{\partial^2}{\partial c_2^2} r_i(c) = -c_1 x_i^2 e^{c_2 x_i}. \tag{2.40}$$

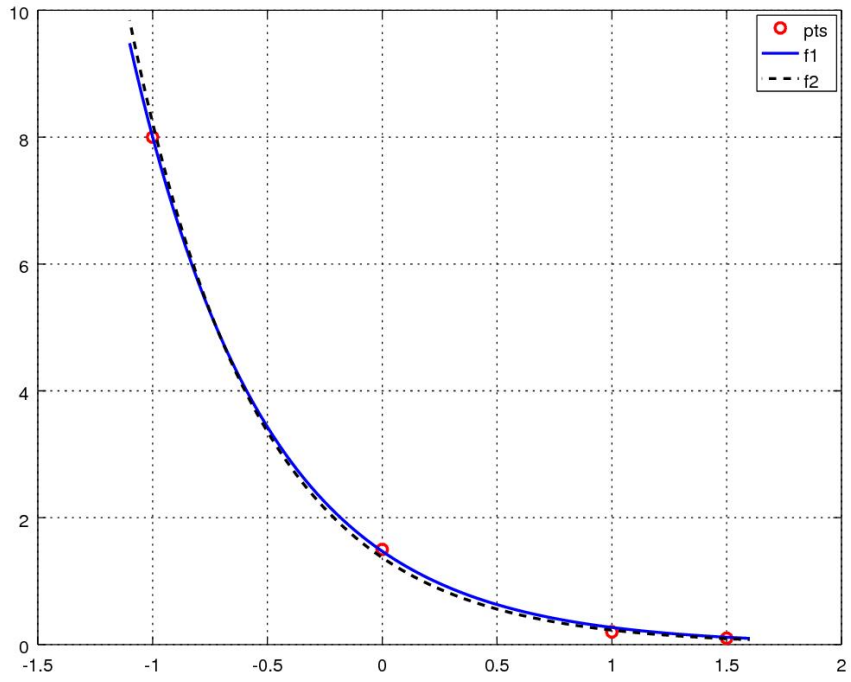


Figura 2.4: Esboço da curva ajustada no Exemplo 2.2.1.

Com isso e tomando $c^{(1)} = (1{,}4,\ -1{,}8)$ (motivado do Exemplo 2.1.3), computamos as iterações de Newton (2.32)-(2.33). Iterando até a precisão de

$TOL = 10^{-4}$, obtemos a solução $c_1 = 1{,}471$ e $c_2 = -1{,}6938$. Na Figura 2.4 vemos uma comparação entre a curva aqui ajustada $(-)$ e aquela obtida no Exemplo 2.1.3 $(--)$.
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);

#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 2.2.1) difere da previamente encontrada no Exemplo 2.1.3. 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.

### 2.2.1 Método de Gauss-Newton

O método de Gauss-Newton é uma técnica iterativa que aproxima o problema não linear de mínimos quadrados (2.27) por uma sequência de problemas 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.

$$f(x; c^{(k+1)}) \approx f(x; c^{(k)}) + \nabla_c f(x; c^{(k)})(c^{(k+1)} - c^{(k)}), \tag{2.41}$$

onde

$$\nabla_c f(x; c) = \left[ \frac{\partial}{\partial c_1} f(x; c) \;\; \frac{\partial}{\partial c_2} f(x; c) \;\; \cdots \;\; \frac{\partial}{\partial c_m} f(x; c) \right]. \tag{2.42}$$

O método consiste em obter a solução do problema não linear (2.27) pelo limite dos seguintes problemas lineares de mínimos quadrados

$$\min_{\delta^{(k)}} \left[ \tilde{E} := \sum_{i=1}^{n} (y_i - f(x_i, c^{(k)}) - \nabla_c f(x_i; c^{(k)}) \delta^{(k)})^2 \right] \tag{2.43}$$

$$c^{(k+1)} = c^{(k)} + \delta^{(k)}. \tag{2.44}$$

Agora, usando a notação de resíduo $r(c) = y - f(x; c)$, observamos que (2.50) consiste no problema linear de mínimos quadrados

$$\min_{\delta^{(k)}} \|r(c^{(k)}) + J_R(c^{(k)}) \delta^{(k)}\|_2^2, \tag{2.45}$$

o qual é equivalente a resolver as equações normais

$$J_R^T(c^{(n)}) J_R(c^{(n)}) \delta^{(n)} = -J_R^T(c) r(c). \tag{2.46}$$

Com isso, dada uma aproximação inicial $c^{(1)}$, a **iteração do método de Gauss-Newton** consiste em

$$J_R^T(c^{(k)}) J_R(c^{(k)}) \delta^{(k)} = -J_R^T(c) r(c) \tag{2.47}$$

$$c^{(k+1)} = c^{(k)} + \delta^{(k)}. \tag{2.48}$$

**Exemplo 2.2.2.** A aplicação da iteração de Gauss-Newton ao problema de mínimos quadrados discutido no Exemplo 2.2.1 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.

### 2.2.2 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

$$\min_{\delta^{(k)}} \{ \| r(c^{(k)}) + J_R(c^{(k)}) \delta^{(k)} \|_2^2 + \mu^{(k)} \| \delta^{(k)} \|_2^2 \} \tag{2.49}$$

ou, equivalentemente,

$$\min_{\delta^{(k)}} \left\| \begin{bmatrix} r(c^{(k)}) \\ 0 \end{bmatrix} + \begin{bmatrix} J_R(c^{(k)}) \\ \mu^{(k)} I \end{bmatrix} \delta^{(k)} \right\|_2^2 \tag{2.50}$$

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 2.2.1.** Quando $\mu^{(k)} \equiv 0$ para todo $n$, o método de Levenberg-Marquardt é equivalente ao método de Gauss-Newton.

**Exemplo 2.2.3.** Consideremos o problema de mínimos quadrados discutido no Exemplo 2.2.1. 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

**E 2.2.1.** 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 $c_3 = -1$.

**E 2.2.2.** Resolva o exercício anterior (Exercício 2.2.1) usando o método de Levenberg-Marquardt com amortecimento constante $\mu = 0{,}2$.

# Capítulo 3

# Derivação

Neste capítulo, discutimos os métodos fundamentais de derivação numérica de funções.

## 3.1 Derivadas de primeira ordem

A derivada de uma função $f$ num ponto $x$ é, por definição,

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}. \tag{3.1}$$

Assim sendo e assumindo $h > 0$[1] próximo de zero, temos que $f'(x)$ pode ser aproximada pela razão fundamental, i.e.

$$f'(x) \approx \underbrace{\frac{f(x+h) - f(x)}{h}}_{D_h f(x)}. \tag{3.2}$$

Analisando a Figura 3.1 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 $(x+h,f(x+h))$.

**Exemplo 3.1.1.** 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 (3.2),

[1] Para fixar notação, assumiremos $h > 0$ ao longo deste capítulo.

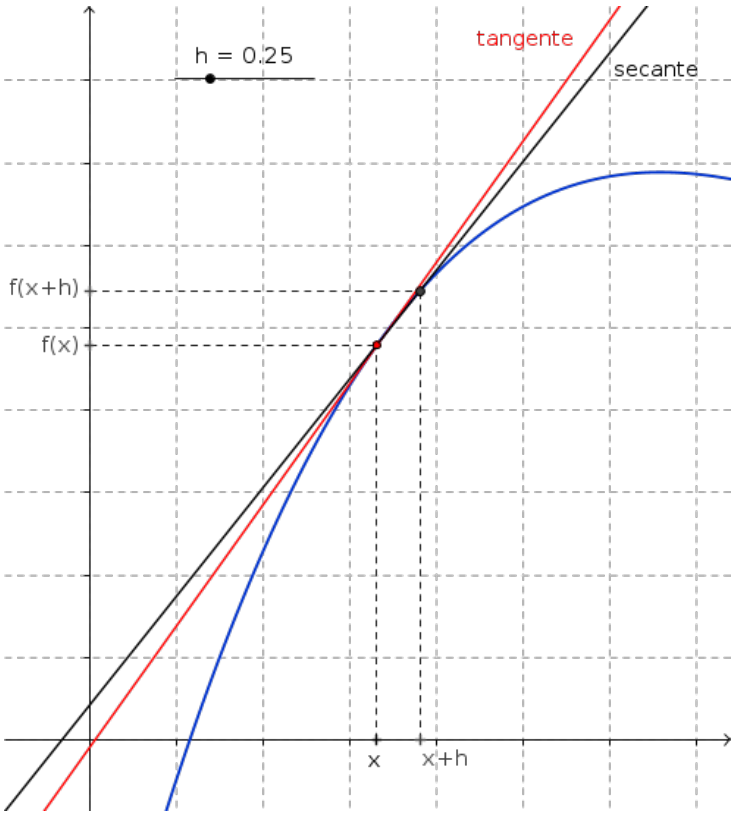


Figura 3.1: Interpretação geométrica da aproximação da derivada pela razão fundamental. Veja no Geogebra.

temos

$$f'\left(\frac{\pi}{3}\right) \approx D_h f(x) = \frac{f\left(\frac{\pi}{3}+h\right) - f\left(\frac{\pi}{3}\right)}{h} \tag{3.3}$$

$$= \frac{\operatorname{sen}\left(\frac{\pi}{3}\right) - \operatorname{sen}\left(\frac{\pi}{3}\right)}{h}. \tag{3.4}$$

Na Tabela 3.1 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 3.1: Valores aproximados da derivada de $f(x) = \operatorname{sen}(x)$ no ponto $x = \pi/6$ usado a expressão (3.2).

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 (3.2) 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.

### 3.1.1 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

$$f(x+h) = f(x) + hf'(x) + O(h^2). \tag{3.5}$$

Agora, isolando $f'(x)$, obtemos

$$f'(x) = \frac{f(x+h) - f(x)}{h} + O(h). \tag{3.6}$$

Isto nos fornece a chamada fórmula de diferenças finitas progressiva de ordem $h$

$$D_{+,h}f(x) := \frac{f(x+h) - f(x)}{h}. \tag{3.7}$$

Observemos que a ordem da fórmula se refere a ordem do **erro de truncamento** com respeito ao passo $h$.

**Exemplo 3.1.2.** 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

$$f'\left(\frac{\pi}{3}\right) \approx D_{+,h}f(x) = \frac{f\left(\frac{\pi}{3}+h\right) - f\left(\frac{\pi}{3}\right)}{h} \tag{3.8}$$

$$= \frac{\text{sen}\left(\frac{\pi}{3}+h\right) - \text{sen}\left(\frac{\pi}{3}\right)}{h}. \tag{3.9}$$

Na Tabela 3.2 temos os valores desta aproximação para diferentes escolhas de $h$, bem como, o erro absoluto da aproximação de $f'(\pi/3)$ por $D_{+,h}f(\pi/3)$.

Tabela 3.2: Resultados referente ao Exemplo 3.1.2.

| $h$ | $D_{+,h}f(\pi/3)$ | $\lvert f'(\pi/3) - D_{+,h}f(\pi/3)\rvert$ |
|---|---|---|
| $10^{-1}$ | 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 |

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)))
```

**Observação 3.1.1.** No exemplo acima (Exemplo 3.1.2), 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 3.2). 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 (3.5), obtemos

$$f(x-h) = f(x) - hf'(x) + O(h^2), \tag{3.10}$$

donde obtemos a fórmula de diferenças finitas regressiva de ordem $h$

$$D_{-,h}f(x) = \frac{f(x) - f(x-h)}{h}. \tag{3.11}$$

**Exemplo 3.1.3.** Consideremos o problema de aproximar a derivada da função $f(x) = \text{sen}(x)$ no ponto $\pi/3$. Usando a fórmula de diferenças finitas regressiva de ordem $h$ obtemos

$$f'\left(\frac{\pi}{3}\right) \approx D_{-,h}f(x) = \frac{f\left(\frac{\pi}{3}\right) - f\left(\frac{\pi}{3} - h\right)}{h} \tag{3.12}$$

$$= \frac{\text{sen}\left(\frac{\pi}{3}\right) - \text{sen}\left(\frac{\pi}{3} - h\right)}{h}. \tag{3.13}$$

Na Tabela 3.3 temos os valores desta aproximação para diferentes escolhas de $h$, bem como, o erro absoluto da aproximação de $f'(\pi/3)$ por $D_{-,h}f(\pi/3)$. No `GNU Octave`, podemos fazer estes cálculos com o seguinte código:

Tabela 3.3: Resultados referente ao Exemplo 3.1.3.

| $h$ | $D_{-,h}f(\pi/3)$ | $\lvert f'(\pi/3) - D_{-,h}f(\pi/3)\rvert$ |
|---|---|---|
| $10^{-1}$ | 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 |

```
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

$$f(x+h) = f(x) + hf'(x) + \frac{h}{2}f''(x) + O(h^3) \tag{3.14}$$

$$f(x-h) = f(x) - hf'(x) + \frac{h}{2}f''(x) + O(h^3). \tag{3.15}$$

Então, subtraindo esta segunda equação da primeira, temos

$$f(x+h) - f(x-h) = 2hf(x) + O(h^3). \tag{3.16}$$

Agora, isolando $f(x)$

$$f(x) = \frac{f(x+h) - f(x-h)}{2h} + O(h^2), \tag{3.17}$$

o que nos fornece a chamada fórmula de diferenças finitas central de ordem $h^2$

$$D_{0,h^2}f(x) := \frac{f(x+h) - f(x-h)}{2h}. \tag{3.18}$$

**Exemplo 3.1.4.** 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

$$f'\left(\frac{\pi}{3}\right) \approx D_{0,h^2}f(x) = \frac{f\left(\frac{\pi}{3}+h\right) - f\left(\frac{\pi}{3}-h\right)}{2h} \tag{3.19}$$

$$= \frac{\text{sen}\left(\frac{\pi}{3}+h\right) - \text{sen}\left(\frac{\pi}{3}-h\right)}{2h}. \tag{3.20}$$

Tabela 3.4: Resultados referente ao Exemplo 3.1.4.

| $h$ | $D_{0,h^2}f(\pi/3)$ | $\lvert f'(\pi/3) - D_{0,h^2}f(\pi/3)\rvert$ |
|---|---|---|
| $10^{-1}$ | 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 3.4 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

**E 3.1.1.** Calcule aproximações da derivada de

$$f(x) = \frac{\text{sen}(x+2) - e^{-x^2}}{x^2 + \ln(x+2)} + x \tag{3.21}$$

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 $h^2$.

**E 3.1.2.** 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.

## 3.2 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

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{3!}f'''(x) + O(h^4), \tag{3.22}$$

$$f(x-h) = f(x) - hf'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{3!}f'''(x) + O(h^4). \tag{3.23}$$

$$\tag{3.24}$$

Somando estas duas equações, obtemos

$$f(x+h) + f(x-h) = 2f(x) + h^2 f''(x) + O(h^4). \tag{3.25}$$

Então, isolando $f''(x)$ temos

$$f''(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} + O(h^2). \tag{3.26}$$

Isto nos leva a definição da **fórmula de diferenças finitas de ordem $h^2$ para a derivada segunda**

$$D^2_{0,h^2} f(x) := \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \tag{3.27}$$

**Exemplo 3.2.1.** 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.

$$f''(x) \approx D_{0,h^2} D_{0,h^2} f(x) = \frac{D_{0,h^2} f(x+h) - D_{0,h^2} f(x-h)}{2h} \tag{3.28}$$

b) Aplicação da fórmula de diferenças finitas central de ordem $h^2$ para a derivada segunda, i.e.

$$f''(x) \approx D^2_{0,h^2} f(x) = \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \tag{3.29}$$

Tabela 3.5: Resultados referente ao Exemplo 3.2.1. Notação: $\delta_{DD} := |f''(\pi/6) - D_{0,h^2} D_{0,h^2} f(\pi/6)|$ e $\delta_{D^2} := |f''(\pi/6) - D^2_{0,h^2} f(\pi/6)|$.

| $h$ | $D_{0,h^2} D_{0,h^2} f(\pi/6)$ | $\delta_{DD}$ | $D^2_{0,h^2} f(\pi/6)$ | $\delta_{D^2}$ |
|---|---|---|---|---|
| $10^{-1}$ | 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 3.5 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 (3.28) mostra que

$$D_{0,h^2} D_{0,h^2} f(x) = \underbrace{\frac{f(x+2h) - 2f(x) + f(x-2h)}{4h^2}}_{D^2_{0,(2h)^2} f(x)}. \tag{3.30}$$

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

**E 3.2.1.** Use a fórmula de diferenças finitas central de ordem $h^2$ para computar aproximações da segunda derivada de

$$f(x) = \frac{\text{sen}(x+2) - e^{-x^2}}{x^2 + \ln(x+2)} + x \tag{3.31}$$

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.

**E 3.2.2.** 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 $h^2$.

## 3.3 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

$$f(x) = p(x) + R_{n+1}(x), \tag{3.32}$$

onde $R(x)$ é o erro na aproximação de $f(x)$ por $p(x)$ e tem a forma

$$R_{n+1}(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{j=1}^{n+1}(x - x_j). \tag{3.33}$$

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 3.3.1.** *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*

$$R_{n+1}^{(k)} = \frac{f^{(n+1)}(\eta)}{(n+1-k)!}\prod_{j=1}^{n+1-k}(x - \xi_j), \tag{3.34}$$

*onde $\xi_j$ 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 $\xi_j$.*

*Demonstração.* Veja [3, Ch.6, Sec.5]. □

### 3.3.1 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.

$$f(x) = p(x) + R_2(x) \tag{3.35}$$

$$= f(x_1)\frac{x - x_2}{x_1 - x_2} + f(x_2)\frac{x - x_1}{x_2 - x_1} + R_2(x). \tag{3.36}$$

Denotando $h = x_2 - x_1$, temos

$$f(x) = f(x_1)\frac{x - x_2}{-h} + f(x_2)\frac{x - x_1}{h} + R_2(x). \tag{3.37}$$

e, derivando com respeito a $x$

$$f'(x) = \frac{f(x_2) - f(x_1)}{h} + R_2^{(1)}(x), \tag{3.38}$$

onde $R_2^{(1)}(x)$ é dado conforme o teorema 3.3.1.
Agora, escolhendo $x = x_1$, temos $x_2 = x_1 + h = x + h$ e, obtemos a **fórmula de diferenças finitas progressiva de ordem** $h$

$$f(x) = \underbrace{\frac{f(x+h) - f(x)}{h}}_{D_{+,h}f(x)} + O(h). \tag{3.39}$$

Se escolhermos $x = x_2$, temos $x_1 = x_2 - h = x - h$, obtemos a **fórmula de diferenças finitas regressiva de ordem** $h$

$$f(x) = \underbrace{\frac{f(x) - f(x-h)}{h}}_{D_{-,h}f(x)} + O(h). \tag{3.40}$$

### 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.

$$f(x) = f(x_1)\frac{(x - x_2)(x - x_3)}{(x_1 - x_2)(x_1 - x_3)} \tag{3.41}$$

$$+ f(x_2)\frac{(x - x_1)(x - x_3)}{(x_2 - x_1)(x_2 - x_3)} \tag{3.42}$$

$$+ f(x_3)\frac{(x - x_1)(x - x_2)}{(x_3 - x_1)(x_3 - x_2)} + R_3(x). \tag{3.43}$$

Derivando em relação a $x$, obtemos

$$f'(x) = f(x_1)\frac{(x_2 - x_3)(2x - x_2 - x_3)}{(x_1 - x_2)(x_1 - x_3)(x_2 - x_3)} \tag{3.44}$$

$$+ f(x_2)\frac{(x_1 - x_3)(-2x + x_1 + x_3)}{(x_1 - x_2)(x_1 - x_3)(x_2 - x_3)} \tag{3.45}$$

$$+ f(x_3)\frac{(x_1 - x_2)(2x - x_1 - x_2)}{(x_1 - x_2)(x_1 - x_3)(x_2 - x_3)} + R_3^{(1)}(x). \tag{3.46}$$

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

$$D_{+,h1,h2}f(x) = \frac{1}{h_1 h_2 (h_1 + h_2)}\left(-h_2(2h_1 + h_2)f(x)\right. \tag{3.47}$$

$$+ (h_1 + h_2)^2 f(x + h_1) \tag{3.48}$$

$$\left. - h_1^2 f(x + h_1 + h_2)\right). \tag{3.49}$$

Agora, assumindo um passo constante $h = h_1 = h_2$, obtemos a **fórmula de diferenças progressiva de ordem $h^2$**

$$D_{+,h^2}f(x) = \frac{1}{2h}\left[-3f(x) + 4f(x + h) - f(x + 2h)\right]. \tag{3.50}$$

Escolhendo $x = x_2$, $x_1 = x - h$ e $x_3 = x + h$ na equação (3.44), obtemos a **fórmula de diferenças finitas central de ordem $h^2$**

$$D_{0,h^2} = \frac{1}{2h}\left[f(x + h) - f(x - h)\right]. \tag{3.51}$$

Por fim, escolhendo $x = x_3$, $x_1 = x - 2h$ e $x_2 = x - h$ na equação (3.44), obtemos a **fórmula de diferenças finitas regressiva de ordem $h^2$**

$$D_{-,h^2} = \frac{1}{2h}\left[3f(x) - 4f(x - h) + f(x - 2h)\right]. \tag{3.52}$$

### 3.3.2 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

$$f(x) = \sum_{i=1}^{5} f(x_i) \left( \prod_{j=1, j\neq i}^{5} \frac{x - x_j}{x_i - x_j} \right) + R_5(x). \tag{3.53}$$

Calculando a derivada em relação a $x$, temos

$$f'(x) = \sum_{i=1}^{5} f(x_i) \left( \sum_{\substack{j=1 \\ j\neq i}}^{5} \prod_{k=1, k\neq i, k\neq j}^{5} \frac{x - x_k}{x_i - x_k} \right) + R_5^{(1)}(x). \tag{3.54}$$

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 $h^4$

$$\begin{aligned} D_{+,h^4} f(x) := \frac{1}{12h} [&f(x - 2h) - 8f(x - h) \\ &+ 8f(x + h) - f(x + 2h)]. \end{aligned} \tag{3.55}$$

## Exercícios

**E 3.3.1.** Use a fórmula de diferenças finitas central de ordem $h^4$ para computar a aproximação da derivada de

$$f(x) = \frac{\operatorname{sen}(x + 2) - e^{-x^2}}{x^2 + \ln(x + 2)} + x \tag{3.56}$$

no ponto $x = 2{,}5$ com passo $h = 0{,}1$.

**E 3.3.2.** 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.

**E 3.3.3.** 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 3.3.2). Para tanto, dê preferência para fórmulas centrais sempre que possível.

# Capítulo 4

# Técnicas de extrapolação

Neste capítulo, estudamos algumas técnicas de extrapolação, as quais serão usadas nos próximos capítulos.

## 4.1 Extrapolação de Richardson

Seja $F_1(h)$ uma aproximação de $I$ tal que

$$I = F_1(h) + \underbrace{k_1 h + k_2 h^2 + k_3 h^3 + O(h^4)}_{\text{erro de truncamento}}. \tag{4.1}$$

Então, dividindo $h$ por 2, obtemos

$$I = F_1\left(\frac{h}{2}\right) + k_1\frac{h}{2} + k_2\frac{h^2}{4} + k_3\frac{h^3}{8} + O(h^4). \tag{4.2}$$

Agora, de forma a eliminarmos o termo de ordem $h$ das expressões acima, subtraímos (4.1) de 2 vezes (4.2), o que nos leva a

$$I = \underbrace{\left[F_1\left(\frac{h}{2}\right) + \left(F_1\left(\frac{h}{2}\right) - F_1(h)\right)\right]}_{F_2(h)} - k_2\frac{h^2}{2} - k_3\frac{3h^3}{4} + O(h^4). \tag{4.3}$$

Ou seja, denotando

$$F_2(h) := F_1\left(\frac{h}{2}\right) + \left(F_1\left(\frac{h}{2}\right) - F_1(h)\right) \tag{4.4}$$

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 $N_2(h/2), i.e.$

$$I = F_2\left(\frac{h}{2}\right) - k_2\frac{h^2}{8} - k_2\frac{3h^3}{32} + O(h^4) \tag{4.5}$$

Então, subtraindo (4.3) de 4 vezes (4.5) de, obtemos

$$I = \underbrace{\left[3F_2\left(\frac{h}{2}\right) + \left(F_2\left(\frac{h}{2}\right) - F_2(h)\right)\right]}_{F_3(h)} + k_3\frac{3h^3}{8} + O(h^4). \tag{4.6}$$

Observemos, ainda, que $N_3(h)$ pode ser reescrita na forma

$$F_3(h) = F_2\left(\frac{h}{2}\right) + \frac{F_2\left(\frac{h}{2}\right) - F_2(h)}{3}, \tag{4.7}$$

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.

$$I = F_3\left(\frac{h}{2}\right) + k_3\frac{3h^3}{64} + O(h^4). \tag{4.8}$$

E, então, subtraindo (4.6) de 8 vezes (4.8), temos

$$I = \underbrace{\left[F_3\left(\frac{h}{2}\right) + \left(\frac{F_3\left(\frac{h}{2}\right) - F_3(h)}{7}\right)\right]}_{F_4(h)} + O(h^4). \tag{4.9}$$

Ou seja,

$$F_4(h) = \left[F_3\left(\frac{h}{2}\right) + \frac{F_3\left(\frac{h}{2}\right) - F_3(h)}{7}\right] \tag{4.10}$$

é uma aproximação de $I$ com erro de truncamento da ordem $h^4$. Estes cálculos nos motivam o seguinte teorema.

**Teorema 4.1.1.** *Seja $F_1(h)$ uma aproximação de $I$ com erro de truncamento da forma*

$$I - F_1(h) = \sum_{i=1}^{n} k_1 h^i + O(h^{n+1}). \tag{4.11}$$

*Então, para $j \geq 2$,*

$$F_j(h) := F_{j-1}\left(\frac{h}{2}\right) + \frac{F_{j-1}\left(\frac{h}{2}\right) - F_{j-1}(h)}{2^{j-1} - 1} \tag{4.12}$$

*é uma aproximação de $I$ com erro de truncamento da forma*

$$\begin{aligned} I - F_j(h) = &\sum_{i=j}^{n} (-1)^{j-1} \frac{(2^{i-1} - 1)\prod_{l=1}^{j-2}\left(2^{i-l-1} - 1\right)}{2^{(j-1)(i-j+1)} d_j} k_i h^i \\ &+ O(h^{n+1}), \end{aligned} \tag{4.13}$$

*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 (4.3). Assumimos, agora, que vale

$$\begin{aligned} I - F_j(h) = &(-1)^{j-1} \frac{(2^{j-1} - 1)\prod_{l=1}^{j-2}\left(2^{j-l-1} - 1\right)}{2^{(j-1)} d_j} k_j h^j \\ &+ \sum_{i=j+1}^{n} (-1)^{j-1} \frac{(2^{i-1} - 1)\prod_{l=1}^{j-2}\left(2^{i-l-1} - 1\right)}{2^{(j-1)(i-j+1)} d_j} k_i h^i \\ &+ O(h^{n+1}). \end{aligned} \tag{4.14}$$

para $j \geq 2$. Então, tomamos

$$\begin{aligned} I - F_j\left(\frac{h}{2}\right) = &(-1)^{j-1} \frac{(2^{j-1} - 1)\prod_{l=1}^{j-2}\left(2^{j-l-1} - 1\right)}{2^{(j-1)} d_j} k_j \frac{h^j}{2^j} \\ &+ \sum_{i=j+1}^{n} (-1)^{j-1} \frac{(2^{i-1} - 1)\prod_{l=1}^{j-2}\left(2^{i-l-1} - 1\right)}{2^{(j-1)(i-j+1)} d_j} k_i \frac{h^i}{2^i} \\ &+ O(h^{n+1}). \end{aligned} \tag{4.15}$$

Agora, subtraímos (4.14) de $2^j$ vezes (4.15), o que nos fornece

$$I = \left[F_j\left(\frac{h}{2}\right) + \frac{F_j\left(\frac{h}{2}\right) - F_j(h)}{2^j - 1}\right] + \sum_{i=j+1}^{n} (-1)^{(j+1)-1} \frac{(2^{i-1}-1)\prod_{l=1}^{(j+1)-2}\left(2^{i-l-1}-1\right)}{2^{((j+1)-1)(i-(j+1)+1)}2^{j-1}d_j} k_i h^i + O(h^{n+1}). \tag{4.16}$$

□

**Corolário 4.1.1.** *Seja $F_1(h)$ uma aproximação de $I$ com erro de truncamento da forma*

$$I - F_1(h) = \sum_{i=1}^{n} k_1 h^{2i} + O(h^{2n+2}). \tag{4.17}$$

*Então, para $j \geq 2$,*

$$F_j(h) := F_{j-1}\left(\frac{h}{2}\right) + \frac{F_{j-1}\left(\frac{h}{2}\right) - F_{j-1}(h)}{4^{j-1} - 1} \tag{4.18}$$

*é uma aproximação de $I$ com erro de truncamento da forma*

$$I - F_j(h) = \sum_{i=j}^{n} (-1)^{j-1} \frac{(4^{i-1}-1)\prod_{l=1}^{j-2}\left(4^{i-l-1}-1\right)}{4^{(j-1)(i-j+1)}d_j} k_i h^{2i} + O(h^{n+1}), \tag{4.19}$$

*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 4.1.1. □

**Exemplo 4.1.1.** Dada uma função $f(x)$, consideremos sua aproximação por diferenças finitas progressiva de ordem $h$, i.e.

$$\underbrace{f'(x)}\, I = \underbrace{\frac{f(x+h) - f(x)}{h}}_{F_1(h)} + \frac{f''(x)}{2}h + \frac{f'''(x)}{6}h^2 + O(h^3). \tag{4.20}$$

Estão, considerando a primeira extrapolação de Richardson, temos

$$F_2(h) = F_1\left(\frac{h}{2}\right) + \left(F_1\left(\frac{h}{2}\right) - F_1(h)\right) \tag{4.21}$$

$$= 4\frac{f(x+h/2) - f(x)}{h} - \frac{f(x+h) - f(x)}{h} \tag{4.22}$$

$$= \frac{-f(x+h) + 4f(x+h/2) - 3f(x)}{h}, \tag{4.23}$$

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 (3.50)).

**Exemplo 4.1.2.** Dada uma função $f(x)$, consideremos sua aproximação por diferenças finitas central de ordem $h^2$, i.e.

$$\underbrace{f'(x)}\, I = \underbrace{\frac{f(x+h) - f(x-h)}{2h}}_{F_1(h)} - \frac{f'''}{6}h^2 - \frac{f^{(5)}(x)}{120}h^4 + O(h^6). \tag{4.24}$$

Estão, considerando a primeira extrapolação de Richardson, temos

$$F_2(h) = F_1\left(\frac{h}{2}\right) + \frac{\left(F_1\left(\frac{h}{2}\right) - F_1(h)\right)}{3} \tag{4.25}$$

$$= \frac{1}{6h}\left[f(x-h) - 8f(x-h/2) + 8f(x+h/2) - f(x+h)\right] \tag{4.26}$$

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 (3.55)).

### 4.1.1 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

$$I - F_1(h) = k_1 h + k_2 h^2 + k_3 h^3 + \cdots + k_n h^n + O(h^{n+1}). \tag{4.27}$$

Então, as sucessivas extrapolações $F_2(h)$, $F_3(h)$, $\ldots$, $F_n(h)$ podem ser organizadas na seguinte forma tabular

$$T = \begin{bmatrix} F_1(h) & & & & \\ F_1(h/2) & F_2(h) & & & \\ F_1(h/2^2) & F_2(h/2) & F_3(h) & & \\ \vdots & \vdots & \vdots & & \\ F_1(h/2^n) & F_2(h/2^{n-1}) & F_3(h/2^{n-2}) & \cdots & F_n(h) \end{bmatrix} \tag{4.28}$$

Desta forma, temos que

$$F_j\left(\frac{h}{2^{i-1}}\right) = t_{i,j-1} + \frac{t_{i,j-1} - t_{i-1,j-1}}{2^{j-1} - 1} \tag{4.29}$$

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 4.1.3.** 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

$$f'\left(\frac{\pi}{3}\right) = \underbrace{\frac{f\left(\frac{\pi}{3} + h\right) - f\left(\frac{\pi}{3}\right)}{h}}_{F_1(h) := D_{+,h} f(\pi/3)} + \frac{f''(x)}{2}h + \frac{f'''(x)}{6}h^2 + \cdots \tag{4.30}$$

Na Tabela 4.1 temos os valores das aproximações de $f'(\pi/3)$ computadas via sucessivas extrapolações de Richardson a partir de (4.30) com $h = 0.1$.

Tabela 4.1: Resultados referente ao Exemplo 4.1.3.

| $O(h)$ | $O(h^2)$ | $O(h^3)$ | $O(h^4)$ |
|---|---|---|---|
| 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 4.1.4.** 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

$$f'\left(\frac{\pi}{3}\right) = \underbrace{\frac{f\left(\frac{\pi}{3}+h\right) - f\left(\frac{\pi}{3}-h\right)}{2h}}_{F_1(h) := D_{0,h^2} f(\pi/3)} - \frac{f'''(x)}{6}h^2 + \frac{f^{(5)}(x)}{120}h^4 - \cdots \tag{4.31}$$

Na Tabela 4.2 temos os valores das aproximações de $f'(\pi/3)$ computadas via sucessivas extrapolações de Richardson a partir de (4.31) 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 4.2: Resultados referente ao Exemplo 4.1.4.

| $O(h^2)$ | $O(h^4)$ | $O(h^6)$ | $O(h^8)$ |
|---|---|---|---|
| 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
```

### 4.1.2 Exercícios

**E 4.1.1.** Mostre que a primeira extrapolação de Richardson de

$$D_{-,h}f(x) = \frac{f(x) - f(x-h)}{h} \tag{4.32}$$

é igual a

$$D_{-,(h/2)^2}f(x) = \frac{3f(x) - 4f(x-h) + f(x-2h)}{h}. \tag{4.33}$$

**E 4.1.2.** Considere o problema de aproximar a derivada de

$$f(x) = \frac{\operatorname{sen}(x+2) - e^{-x^2}}{x^2 + \ln(x+2)} + x \tag{4.34}$$

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 5

# 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

$$\int_a^b f(x)\,dx \approx \sum_{i=1}^{n} f(x_i)w_i, \tag{5.1}$$

onde $x_i$ e $w_i$ são, respectivamente, o $i$-ésimo nodo e o $i$-ésimo peso da quadratura, $i = 1, 2, \ldots, n$.

## 5.1 Regras de Newton-Cotes

Dada uma função $f(x)$ e um intervalo $[a, b]$, denotamos por

$$I := \int_a^b f(x)\,dx. \tag{5.2}$$

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

$$f(x) = p(x) + R_{n+1}(x), \tag{5.3}$$

onde

$$p(x) = \sum_{i=1}^{n+1} f(x_i) \prod_{\substack{j=1 \\ j \neq i}}^{n+1} \frac{(x - x_j)}{x_i - x_j} \tag{5.4}$$

e

$$R_{n+1}(x) = \frac{f^{(n+1)}(\xi)}{(n+1)!} \prod_{j=1}^{n+1} (x - x_j), \tag{5.5}$$

onde $\xi = \xi(x)$ pertencente ao intervalo $[x_1, x_{n+1}]$. Deste modo, temos

$$\begin{aligned} I &:= \int_a^b f(x) && (5.6)\\ &= \int_a^b p(x)\,dx + \int_a^b R_{n+1}(x)\,dx && (5.7)\\ &= \underbrace{\sum_{i=1}^{n+1} f(x_i) \int_a^b \prod_{\substack{j=1 \\ j \neq i}}^{n+1} \frac{(x - x_j)}{x_i - x_j)}\,dx}_{\text{quadratura}} + \underbrace{\int_a^b R_{n+1}(x)\,dx}_{\text{erro de truncamento}} && (5.8) \end{aligned}$$

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].

### 5.1.1 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:

$$w_1 = \int_a^b \frac{x-b}{a-b}\,dx \tag{5.9}$$

$$= \frac{(b-a)}{2} = \frac{h}{2} \tag{5.10}$$

e

$$w_2 = \int_a^b \frac{x-a}{b-a}\,dx \tag{5.11}$$

$$= \frac{(b-a)}{2} = \frac{h}{2}. \tag{5.12}$$

Agora, estimamos o erro de truncamento com

$$E := \int_a^b R_2(x)\,dx \tag{5.13}$$

$$= \int_a^b \frac{f''(\xi(x))}{2}(x-a)(x-b)\,dx \tag{5.14}$$

$$\leq C\left|\int_a^b (x-a)(x-b)\,dx\right| \tag{5.15}$$

$$= C\frac{(b-a)^3}{6} = O(h^3). \tag{5.16}$$

Portanto, a **regra do trapézio** é dada por

$$\int_a^b f(x)\,dx = \frac{h}{2}(f(a)+f(b)) + O(h^3). \tag{5.17}$$

**Exemplo 5.1.1.** Consideremos o problema de computar a integral de $f(x) = xe^{-x^2}$ no intervalo $[0, 1/4]$. Analiticamente, temos

$$I = \int_0^{1/4} xe^{-x^2}\,dx = -\frac{e^{-x^2}}{2}\bigg|_0^{1/4} \tag{5.18}$$

$$= \frac{1-e^{-1/4}}{2} = 3{,}02935\text{E}{-2}. \tag{5.19}$$

[1]Neste capítulo, $h$ é escolhido como a distância entre os nodos.

Agora, usando a regra do trapézio, obtemos a seguinte aproximação para $I$

$$I \approx \frac{h}{2}(f(0) + f(1/2)) \tag{5.20}$$

$$= \frac{1/4}{2}\left(0 + \frac{1}{4}e^{-(1/4)^2}\right) = 2{,}93567\text{E}{-2}. \tag{5.21}$$

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:

$$w_1 = \int_a^b \frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)}\,dx \tag{5.22}$$

$$= \frac{(b-a)}{6} = \frac{h}{6}, \tag{5.23}$$

$$w_2 = \int_a^b \frac{(x-x_1)(x-x_3)}{(x_2-x_1)(x_2-x_3)}\,dx \tag{5.24}$$

$$= 4\frac{(b-a)}{6} = 4\frac{h}{6} \tag{5.25}$$

e

$$w_3 = \int_a^b \frac{(x-x_1)(x-x_2)}{(x_3-x_1)(x_3-x_2)}\,dx \tag{5.26}$$

$$= \frac{(b-a)}{6} = \frac{h}{6}. \tag{5.27}$$

Isto nos fornece a chamada **regra de Simpson**

$$I \approx \frac{h}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] \tag{5.28}$$

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.

$$\begin{aligned} f(x) = f(x_2) + f'(x_2)(x-x_2) + \frac{f''(x_2)}{2}(x-x_2)^2 \\ + \frac{f'''(x_2)}{6}(x-x_2)^3 \\ + \frac{f^{(4)}(\xi_1(x))}{24}(x-x_2)^4, \end{aligned} \tag{5.29}$$

donde

$$\begin{aligned} \int_a^b f(x)\,dx = 2hf(x_2) + \frac{h^3}{3}f''(x_2) \\ + \frac{1}{24}\int_a^b f^{(4)}(\xi_1(x))(x-x_2)^4\,dx. \end{aligned} \tag{5.30}$$

Daí, usando da fórmula de diferenças finitas central de ordem $h^2$, temos

$$f''(x_2) = \frac{f(x_1) - 2f(x_2) + f(x_3)}{h^2} + O(h^2). \tag{5.31}$$

Ainda, o último termo da equação (5.30) pode ser estimado por

$$\left|\frac{1}{24}\int_a^b f^{(4)}(\xi_1(x))(x-x_2)^4\,dx\right| \leq C\left|\int_a^b (x-x_2)^4\,dx\right| \tag{5.32}$$

$$= C(b-a)^5 = O(h^5). \tag{5.33}$$

Então, de (5.30), (5.31) e (5.1.1), temos

$$\int_a^b f(x)\,dx = \frac{h}{3}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] + O(h^5), \tag{5.34}$$

o que mostra que a **regra de Simpson tem erro de truncamento da ordem** $h^5$.

**Exemplo 5.1.2.** Aproximando a integral dada no Exemplo 5.1.1 pela a regra de Simpson, temos

$$\int_0^{1/4} f(x)\,dx \approx \frac{1/8}{3}\left[f(0) + 4f\left(\frac{1}{8}\right) + f\left(\frac{1}{4}\right)\right] \tag{5.35}$$

$$= \frac{1}{24}\left[\frac{1}{2}e^{-(1/8)^2} + \frac{1}{4}e^{-(1/4)^2}\right] \tag{5.36}$$

$$= 3{,}02959\text{E}{-}2. \tag{5.37}$$

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)
```

### 5.1.2 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

$$\int_a^b f(x)\,dx = \int_a^b f(x_1)\,dx + \int_a^b f'(\xi(x))(x - x_1)\,dx, \tag{5.38}$$

donde, denotando $h := (b - a)$, temos

$$\int_a^b f(x),dx = hf\left(\frac{a+b}{2}\right) + O(h^3). \tag{5.39}$$

Deixa-se para o leitor a verificação do erro de truncamento (veja, Exercício 5.1.3).

**Exemplo 5.1.3.** Aproximando a integral dada no Exemplo 5.1.1 pela a regra do ponto médio, temos

$$\int_0^{1/4} f(x)\,dx \approx \frac{1}{4} f\left(\frac{1}{8}\right) \tag{5.40}$$

$$= \frac{1}{32} e^{-(1/8)^2} \tag{5.41}$$

$$= 3{,}07655\text{E}{-2} \tag{5.42}$$

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

**E 5.1.1.** Aproxime

$$\int_{-1}^{0} \frac{\operatorname{sen}(x+2) - e^{-x^2}}{x^2 + \ln(x+2)}\,dx \tag{5.43}$$

usando a:

a) regra do ponto médio.

b) regra do trapézio.

c) regra de Simpson.

**E 5.1.2.** 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.

**E 5.1.3.** 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.

**E 5.1.4.** Obtenha a regra de Newton-Cotes aberta de 2 pontos e estime seu erro de truncamento.

## 5.2 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.

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} \int_{x_i}^{x_{i+1}} f(x)\,dx, \tag{5.46}$$

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.

### 5.2.1 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

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} \int_{\tilde{x}_i}^{\tilde{x}_{i+1}} f(x)\,dx \tag{5.47}$$

$$= \sum_{i=1}^{n} \left[ hf\left(\frac{\tilde{x}_i + \tilde{x}_{i+1}}{2}\right) + O(h^3) \right]. \tag{5.48}$$

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**

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} hf(x_i) + O(h^2). \tag{5.49}$$

**Exemplo 5.2.1.** 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

$$\underbrace{\int_a^b f(x)\,dx}_{I} \approx \underbrace{\sum_{i=1}^{n} hf(x_i)}_{S}, \tag{5.50}$$

onde $h = 1/(4n)$ e $x_i = (i - 1/2)h$, $i = 1, 2, \ldots, n$. Na Tabela 5.1, temos as aproximações computadas com diversos números de subintervalos, bem como, seus erros absolutos.

Tabela 5.1: Resultados referentes ao Exemplo 5.2.1.

| $n$ | $S$ | $\lvert I - S\rvert$ |
|---|---|---|
| 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))
```

### 5.2.2 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

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} \int_{x_i}^{x_{i+1}} f(x)\,dx \tag{5.51}$$

$$= \sum_{i=1}^{n} \left\{ \frac{h}{2}\left[f(x_i) + f(x_{i+1})\right] + O(h^3) \right\} \tag{5.52}$$

$$= f(x_1)\frac{h}{2} + \sum_{i=2}^{n} h f(x_i) + f(x_{n+1})\frac{h}{2} + O(h^2). \tag{5.53}$$

Desta forma, a regra composto do trapézio com $n$ subintervalos é

$$\int_a^b f(x)\,dx = \frac{h}{2}\left[f(x_1) + \sum_{i=2}^{n} 2f(x_i) + f(x_{n+1})\right] + O(h^2), \tag{5.54}$$

onde $h = (b-a)/n$ e $x_i = a + (i-1)h$, $i = 1, 2, \ldots, n$.

**Exemplo 5.2.2.** 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

$$\underbrace{\int_a^b f(x)\,dx}_{I} \approx \underbrace{\frac{h}{2}\left[f(x_1) + 2\sum_{i=2}^{n-1} f(x_i) + f(x_{n+1})\right]}_{S}, \tag{5.55}$$

onde $h = 1/(4n)$ e $x_i = (i-1)h$, $i = 1, 2, \ldots, n$. Na Tabela 5.2, temos as aproximações computadas com diversos números de subintervalos, bem como, seus erros absolutos.
Podemos fazer estas computações com o auxílio do seguinte código `GNU Octave`:

Tabela 5.2: Resultados referentes ao Exemplo 5.2.2.

| $n$ | $S$ | $\lvert I-S\rvert$ |
|---|---|---|
| 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 |

```
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))
```

### 5.2.3 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

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} \int_{\tilde{x}_i}^{\tilde{x}_{i+1}} f(x)\,dx \tag{5.56}$$

$$= \sum_{i=1}^{n} \left\{ \frac{h}{3} \left[ f(\tilde{x}_i) + 4f\left( \frac{\tilde{x}_i + \tilde{x}_{i+1}}{2} \right) + f(\tilde{x_{i+1}}) \right] + O(h^5) \right\}. \tag{5.57}$$

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

$$\int_a^b f(x)\,dx = \frac{h}{3}\left[f(x_1) + 2\sum_{i=2}^{n} f(x_{2i-1}) + 4\sum_{i=1}^{n} f(x_{2i}) + f(x_{n+1})\right] + O(h^4) \tag{5.58}$$

**Exemplo 5.2.3.** Consideremos o problema de computar a integral de $f(x) = xe^{-x^2}$ no intervalo $[0, 1]$. Usando a regra composta de Simpson com $n$ subintervalos, obtemos a aproximação

$$\underbrace{\int_a^b f(x)\,dx}_{I} \approx \underbrace{\frac{h}{3}\left[f(x_1) + 2\sum_{i=2}^{n} f(x_{2i-1}) + 4\sum_{i=1}^{n} f(x_{2i}) + f(x_{n+1})\right]}_{S}, \tag{5.59}$$

onde $h = 1/(8n)$ e $x_i = (i-1)h$, $i = 1, 2, \ldots, n$. Na Tabela 5.3, temos as aproximações computadas com diversos números de subintervalos, bem como, seus erros absolutos.

Tabela 5.3: Resultados referentes ao Exemplo 5.2.3.

| $n$ | $S$ | $\lvert I - S\rvert$ |
|---|---|---|
| 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

**E 5.2.1.** Aproxime

$$\int_{-1}^{0} \frac{\operatorname{sen}(x+2) - e^{-x^2}}{x^2 + \ln(x+2)}\, dx \tag{5.60}$$

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.

**E 5.2.2.** 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.

## 5.3 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

$$R_{k,1} := \frac{h_k}{2}\left[f(a) + 2\sum_{i=2}^{2k} f(x_i) + f(b)\right] \tag{5.61}$$

a regra do trapézio composta com $2k$ subintervalos de

$$I := \int_a^b f(x)\,dx. \tag{5.62}$$

Por sorte, o erro de truncamento de aproximar $I$ por $R_{k,1}$ tem a seguinte forma

$$I - R_{k,1} = \sum_{i=1}^{\infty} k_i h_k^{2i}, \tag{5.63}$$

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

$$R_{k,j} := R_{k,j-1} + \frac{R_{k,j-1} - R_{k-1,j-1}}{4^{j-1} - 1}, \tag{5.64}$$

$j = 2, 3, \ldots, n$, de forma a computarmos $R_{n,n}$, a qual fornece a aproximação desejada.

**Exemplo 5.3.1.** 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

$$R_{1,1} := \frac{1}{2}[f(0) + f(1)] = 1{,}83940\text{E}{-1} \tag{5.65}$$

$$R_{2,1} := \frac{1}{4}[f(0) + 2f(1/2) + f(1)] = 2{,}86670\text{E}{-1}. \tag{5.66}$$

Então, calculando

$$R_{2,2} = R_{2,1} + \frac{R_{2,1} - R_{1,1}}{3} = 3{,}20914\text{E}{-1}, \tag{5.67}$$

Tabela 5.4: Resultados referentes ao Exemplo 5.3.1.

| 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 |

a qual é a aproximação desejada.
Na Tabela 5.4, 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

**E 5.3.1.** Aproxime

$$\int_{-1}^{0} \frac{\operatorname{sen}(x+2) - e^{-x^2}}{x^2 + \ln(x+2)}\, dx \tag{5.68}$$

usando a quadratura de Romberg de ordem 4.

## 5.4 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

$$\int_a^b p(x)\, dx = \sum_{i=1}^{n} p(x_i) w_i \tag{5.69}$$

para todo polinômio $p(x)$ de grau menor $m$. Ou ainda, conforme descrito na definição a seguir.

**Definição 5.4.1.** Dizemos que uma dada quadratura numérica de pontos e nodos $\{x_i, w_i\}_{i=1}^{n}$ tem **grau de exatidão** $m$, quando

$$\int_a^b x^k\, dx = \sum_{i=1}^{n} x_i^k w_i,\ \forall k \leq m. \tag{5.70}$$

**Exemplo 5.4.1.** Determinemos o grau de exatidão da regra do ponto médio. Para tanto, verificamos para quais $k$ vale

$$\int_a^b x^k\, dx = (b-a)\left(\frac{a+b}{2}\right)^k. \tag{5.71}$$

Vejamos:

- $k = 0$:

$$\int_a^b x^0\,dx = x\big|_a^b = b - a, \tag{5.72}$$

$$(b-a)\left(\frac{a+b}{2}\right)^0 = b - a. \tag{5.73}$$

- $k = 1$:

$$\int_a^b x^1\,dx = \frac{x^2}{2}\bigg|_a^b = \frac{b^2}{2} - \frac{a^2}{2}, \tag{5.74}$$

$$(b-a)\left(\frac{a+b}{2}\right)^1 = (b-a)\frac{(a+b)}{2} = \frac{b^2}{2} - \frac{a^2}{2}. \tag{5.75}$$

- $k = 2$:

$$\int_a^b x^2\,dx = \frac{x^3}{3}\bigg|_a^b = \frac{b^3}{3} - \frac{a^3}{3}, \tag{5.76}$$

$$(b-a)\left(\frac{a+b}{2}\right)^2 \neq \frac{b^3}{3} - \frac{a^3}{3}. \tag{5.77}$$

Ou seja, a regra do ponto média tem grau de exatidão 1.

**Exemplo 5.4.2.** Determinemos o grau de exatidão da regra de Simpson. Para tanto, verificamos para quais $k$ vale

$$\int_a^b x^k\,dx = \frac{(b-a)}{6}\left(f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right)^k. \tag{5.78}$$

Vejamos:

- $k = 0$:

$$\int_a^b x^0\,dx = x\big|_a^b = b - a, \tag{5.79}$$

$$\frac{(b-a)}{6}\left(a^0 + 4\left(\frac{a+b}{2}\right)^0 + b^0\right) = b - a. \tag{5.80}$$

- $k = 1$:

$$\int_a^b x^1\,dx = \left.\frac{x^2}{2}\right|_a^b = \frac{b^2}{2} - \frac{a^2}{2}, \tag{5.81}$$

$$\frac{(b-a)}{6}\left(a^1 + 4\left(\frac{a+b}{2}\right)^1 + b^1\right) = \frac{(b-a)}{2}(a+b) \tag{5.82}$$

$$= \frac{b^2}{2} - \frac{a^2}{2}. \tag{5.83}$$

- $k = 2$:

$$\int_a^b x^2\,dx = \left.\frac{x^3}{3}\right|_a^b = \frac{b^3}{3} - \frac{a^3}{3}, \tag{5.84}$$

$$\frac{(b-a)}{6}\left(a^2 + 4\left(\frac{a+b}{2}\right)^2 + b^2\right) = \frac{(b-a)}{3}(a^2 + ab + b^2) \tag{5.85}$$

$$= \frac{b^3}{3} - \frac{a^3}{3}. \tag{5.86}$$

- $k = 3$:

$$\int_a^b x^3\,dx = \left.\frac{x^4}{4}\right|_a^b = \frac{b^4}{4} - \frac{a^4}{4}, \tag{5.87}$$

$$\frac{(b-a)}{6}\left(a^3 + 4\left(\frac{a+b}{2}\right)^3 + b^3\right) \tag{5.88}$$

$$= \frac{(b-a)}{6}\left[\frac{3a^3}{2} + \frac{3b}{2}a^2 + \frac{3a}{2}b^2 + \frac{3b^3}{2}\right] \tag{5.89}$$

$$= \frac{b^4}{4} - \frac{a^4}{4}. \tag{5.90}$$

- $k = 4$:

$$\int_a^b x^4\,dx = \left.\frac{x^5}{5}\right|_a^b = \frac{b^5}{5} - \frac{a^5}{5}, \tag{5.91}$$

$$\frac{(b-a)}{6}\left(a^4 + 4\left(\frac{a+b}{2}\right)^4 + b^4\right) \neq \frac{b^5}{5} - \frac{a^5}{5}. \tag{5.92}$$

Ou seja, a regra de Simpson tem grau de exatidão 3.

### Exercícios

**E 5.4.1.** Determine o grau de exatidão da regra do trapézio.

**E 5.4.2.** Determine o nodo e o peso da quadratura numérica de um único nodo e de grau de exatidão 1 para o intervalo de integração $[-1, 1]$.

## 5.5 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

$$\int_{-1}^{1} f(x)\,dx. \tag{5.93}$$

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

$$w_1 x_1^0 = \int_{-1}^{1} x^0\,dx \Rightarrow w_1 = x|_{-1}^{1} = 2. \tag{5.94}$$

Agora, exigindo o grau de exatidão 1, obtemos

$$w_1 x_1^1 = \int_{-1}^{1} x^1\,dx \Rightarrow 2x_1 = \left.\frac{x^2}{2}\right|_{-1}^{1} = 0 \tag{5.95}$$

$$\Rightarrow x_1 = 0. \tag{5.96}$$

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

$$\sum_{i=1}^{n} x_i^k w_i = \int_{-1}^{1} x^k\,dx,\ k = 0,1,2,\ldots,2n-1. \tag{5.97}$$

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 (5.97) 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

$$\int_{-1}^{1} p(x)P_n(x)\,dx = 0, \tag{5.98}$$

para todo polinômio $p(x)$ de grau menor que $n$. Com isso, estabelecemos o seguinte resultado.

**Teorema 5.5.1.** *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*

$$w_i = \int_{-1}^{1} \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j}\,dx. \tag{5.99}$$

*Demonstração.* Sejam $x_1, x_2, \ldots, x_n$ as raízes do polinômio de Legendre de grau $n$. Queremos mostrar que

$$\int_{-1}^{1} p(x)\,dx = \sum_{i=1}^{n} p(x_i)w_i, \tag{5.100}$$

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

$$\int_{-1}^{1} p(x)\,dx = \int_{-1}^{1} \sum_{i=1}^{n} p(x_i) \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j}\,dx \tag{5.101}$$

$$= \sum_{i=1}^{n} p(x_i) \int_{-1}^{1} \prod_{\substack{j=1 \\ j\neq i}}^{n} \frac{x-x_j}{x_i-x_j}\,dx \tag{5.102}$$

$$= \sum_{i=1}^{n} p(x_i)w_i. \tag{5.103}$$

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

$$p(x) = q(x)P_n(x) + r(x), \tag{5.104}$$

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 (5.98)), temos

$$\int_{-1}^{1} p(x)\,dx = \int_{-1}^{1} q(x)P_n(x) + r(x)\,dx \tag{5.105}$$

$$= \int_{-1}^{1} r(x)\,dx. \tag{5.106}$$

Agora, do resultado anterior aplicado a $r(x)$, temos

$$\int_{-1}^{1} p(x)\,dx = \sum_{i=1}^{n} r(x_i)w_i = \sum_{i=1}^{n} p(x_i)w_i. \tag{5.107}$$

Isto complete o resultado para polinômios de grau menor ou igual a $2n-1$. □

**Exemplo 5.5.1.** Consideremos a quadratura de Gauss-Legendre de 2 nodos. Do teorema anterior (Teorema 5.5.1, seus nodos são as raízes do polinômio de Legendre de grau 2

$$P_2(x) = \frac{3}{2}x^2 - \frac{1}{2}, \tag{5.108}$$

as quais são

$$x_1 = -\frac{\sqrt{3}}{3}, \quad x_2 = \frac{\sqrt{3}}{3}. \tag{5.109}$$

Os pesos são, então

$$w_1 = \int_{-1}^{1} \frac{x - x_1}{x_2 - x_1}\,dx \tag{5.110}$$

$$= \frac{\sqrt{3}}{2}\left[\frac{x^2}{2} + \frac{\sqrt{3}}{3}x\right]_{-1}^{1} \tag{5.111}$$

$$= 1 \tag{5.112}$$

e

$$w_2 = \int_{-1}^{1} \frac{x - x_2}{x_1 - x_2}\, dx \tag{5.113}$$

$$= -\frac{\sqrt{3}}{2}\left[\frac{x^2}{2} - \frac{\sqrt{3}}{3}x\right]_{-1}^{1} \tag{5.114}$$

$$= 1 \tag{5.115}$$

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:

- $k = 0$:

$$\int_{-1}^{1} x^0\, dx = 2 \tag{5.116}$$

$$x_1^0 w_1 + x_2^0 w_2 = \left(-\frac{\sqrt{3}}{3}\right)^0 + \left(\frac{\sqrt{3}}{3}\right)^0 = 2. \tag{5.117}$$

- $k = 1$:

$$\int_{-1}^{1} x^1\, dx = 0 \tag{5.118}$$

$$x_1^1 w_1 + x_2^1 w_2 = \left(-\frac{\sqrt{3}}{3}\right)^1 + \left(\frac{\sqrt{3}}{3}\right)^1 = 0. \tag{5.119}$$

- $k = 2$:

$$\int_{-1}^{1} x^2\, dx = \frac{2}{3} \tag{5.120}$$

$$x_1^2 w_1 + x_2^2 w_2 = \left(-\frac{\sqrt{3}}{3}\right)^2 + \left(\frac{\sqrt{3}}{3}\right)^2 = \frac{2}{3}. \tag{5.121}$$

- $k = 3$:

$$\int_{-1}^{1} x^3\, dx = 0 \tag{5.122}$$

$$x_1^3 w_1 + x_2^3 w_2 = \left(-\frac{\sqrt{3}}{3}\right)^3 + \left(\frac{\sqrt{3}}{3}\right)^3 = 0. \tag{5.123}$$

- $k = 4$:

$$\int_{-1}^{1} x^4 \, dx = \frac{2}{5} \tag{5.124}$$

$$x_1^4 w_1 + x_2^4 w_2 = \left(-\frac{\sqrt{3}}{3}\right)^4 + \left(\frac{\sqrt{3}}{3}\right)^4 = \frac{2}{9}. \tag{5.125}$$

Tabela 5.5: 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 5.5.1.** O conjunto de nodos e pesos da quadratura de Gauss-Legendre para $n = 1, 2, 3, 4, 5$ são apresentados na Tabela 5.5[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 (5.99) ou [5, Cap.4, Sec. 4.6]:

$$w_i = \frac{2}{(1-x_i^2)\left[P_n'(x_i)\right]^2}, \quad i = 1, 2, \ldots, n. \tag{5.126}$$

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 5.5.2.** 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:

- $n = 1$:
$$\int_{-1}^{1} \cos(x)\,dx \approx 2\cos 0 = 2. \tag{5.127}$$
- $n = 2$:
$$\int_{-1}^{1} xe^{-x^2}\,dx \approx \cos(-\sqrt{3}/3) + \cos(-\sqrt{3}/3) = 1{,}67582. \tag{5.128}$$
- $n = 3$:
$$\begin{aligned}\int_{-1}^{1} xe^{-x^2}\,dx \approx \frac{8}{9}\cos 0 + \frac{5}{9}\cos(-\sqrt{3/5}) \\ + \frac{5}{9}\cos(\sqrt{3/5}) = 1{,}68300.\end{aligned} \tag{5.129}$$

Na Tabela 5.6, 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:

| $n$ | $\tilde{I}$ | $\lvert I-\tilde{I}\rvert$ |
|---|---|---|
| 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 5.6: Resultados referentes ao Exemplo 5.5.2.

```
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))
```

### 5.5.1 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

$$x = \frac{b-a}{2}(u+1) + a \tag{5.130}$$

temos

$$dx = \frac{b-a}{2}du \tag{5.131}$$

e, portanto,

$$\int_a^b f(x)\,dx = \int_{-1}^1 f\left(\frac{b-a}{2}(u+1)+a\right)\cdot\frac{b-a}{2}du. \tag{5.132}$$

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 5.5.3.** Usemos a quadratura de Gauss-Legendre com 2 pontos para aproximar a integral

$$\int_0^1 xe^{-x^2}\,dx. \tag{5.133}$$

Fazendo a mudança de variável $x = u/2 + 1/2$, temos

$$\int_0^1 xe^{-x^2}\,dx = \int_{-1}^1 \left(\frac{u}{2}+\frac{1}{2}\right)e^{-\left(\frac{u}{2}+\frac{1}{2}\right)^2}\,du. \tag{5.134}$$

Então, aplicando a quadratura temos

$$\int_0^1 xe^{-x^2}\,dx = \left(-\frac{\sqrt{3}}{6}+\frac{1}{2}\right)e^{-\left(-\frac{\sqrt{3}}{6}+\frac{1}{2}\right)^2} + \left(\frac{\sqrt{3}}{6}+\frac{1}{2}\right)e^{-\left(\frac{\sqrt{3}}{6}+\frac{1}{2}\right)^2} \tag{5.135}$$

$$= 3{,}12754\text{E}{-}1. \tag{5.136}$$

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

**E 5.5.1.** Aproxime

$$\int_{-1}^1 \frac{\operatorname{sen}(x+2)-e^{-x^2}}{x^2+\ln(x+2)}\,dx \tag{5.137}$$

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.

**E 5.5.2.** Aproxime

$$\int_0^1 \frac{\operatorname{sen}(x+2)-e^{-x^2}}{x^2+\ln(x+2)}\,dx \tag{5.138}$$

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.

**E 5.5.3.** Aproxime

$$\int_{-1}^{1} \frac{\operatorname{sen}(x+2) - e^{-x^2}}{x^2 + \ln(x+2)}\, dx \tag{5.139}$$

usando a quadratura de Gauss-Legendre com:

a) $n = 5$ ponto.

b) $n = 10$ pontos.

c) $n = 20$ pontos.

## 5.6 Quadraturas gaussianas com pesos

A quadratura gaussiana estudada na seção anterior (Seção 5.5) é um caso particular de quadraturas de máximo grau de exatidão para integrais da forma

$$\int_a^b f(x)w(x)\, dx, \tag{5.140}$$

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

$$\int_a^b q(x)p_n(x)w(x)\, dx = 0, \tag{5.141}$$

para todo polinômio $q(x)$ de grau menor que $n$.

### 5.6.1 Quadratura de Gauss-Chebyshev

Quadraturas de Gauss-Chebyshev são quadraturas gaussianas para integrais da forma

$$\int_{-1}^{1} f(x)(1-x^2)^{-1/2}\,dx. \tag{5.142}$$

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

$$x_i = \cos\left(\frac{2i-1}{2n}\pi\right), \tag{5.143}$$

$$w_i = \frac{\pi}{n}. \tag{5.144}$$

**Exemplo 5.6.1.** Considere o problema de aproximar a integral

$$\int_{-1}^{1} \frac{e^{-x^2}}{\sqrt{1-x^2}}\,dx. \tag{5.145}$$

Usando a quadratura de Gauss-Chebyshev de $n$ pontos temos:

- $n = 1$:

$$\int_{-1}^{1} \frac{e^{-x^2}}{\sqrt{1-x^2}}\,dx \approx \pi e^{-\cos(\pi/2)^2} = \pi. \tag{5.146}$$

- $n = 2$:

$$\int_{-1}^{1} \frac{e^{-x^2}}{\sqrt{1-x^2}}\,dx \approx \frac{\pi}{2}e^{-\cos(\pi/4)^2} + \frac{\pi}{2}e^{-\cos(3\pi/4)^2} \tag{5.147}$$

$$= 1{,}90547. \tag{5.148}$$

- $n = 3$:

$$\int_{-1}^{1} \frac{e^{-x^2}}{\sqrt{1-x^2}}\,dx \approx \frac{\pi}{3}e^{-\cos(\pi/6)^2} + \frac{\pi}{3}e^{-\cos(\pi/2)^2} + \frac{\pi}{3}e^{-\cos(5\pi/6)^2} \tag{5.149}$$

$$= 2{,}03652. \tag{5.150}$$

Na Tabela 5.7, 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:

| $n$ | $\tilde{I}$ |
|---|---|
| 1 | 3,14159 |
| 2 | 1,90547 |
| 3 | 2,03652 |
| 4 | 2,02581 |
| 5 | 2,02647 |
| 6 | 2,02644 |
| 10 | 2,02644 |

Tabela 5.7: Resultados referentes ao Exemplo 5.6.1.

```
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)
```

### 5.6.2 Quadratura de Gauss-Laguerre

Quadraturas de Gauss-Laguerre são quadraturas gaussianas para integrais da forma

$$\int_0^\infty f(x)e^{-x}\,dx. \tag{5.151}$$

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

$$w_i = -\frac{1}{n[L_n'(x_i)]^2},\ i = 1, 2, \ldots, n. \tag{5.152}$$

Na Tabela 5.8, temos os pontos da quadratura de Gauss-Laguerre para diversos valores de $n$.
No `GNU Octave`, os pontos da quadratura de Gauss-Laguerre podem ser obtido com o seguinte código:

```
pkg load miscellaneous
```

Tabela 5.8: 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 |

```
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 5.6.2.** Na Tabela 5.9, 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 $n$.
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];
```

| $n$ | $\tilde{I}$ |
|---|---|
| 1 | 8,41471E−01 |
| 2 | 4,32459E−01 |
| 3 | 4,96030E−01 |
| 4 | 5,04879E−01 |
| 5 | 4,98903E−01 |

Tabela 5.9: Resultados referentes ao Exemplo 5.6.1.

```
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)
```

### 5.6.3 Quadratura de Gauss-Hermite

Quadraturas de Gauss-Hermite são quadraturas gaussianas para integrais da forma

$$\int_{-\infty}^{\infty} f(x)e^{-x^2}\,dx. \tag{5.153}$$

Seus nodos $x_i$, $i = 1, 2, \ldots, n$ são as raízes do $n$-ésimo polinômio de Hermite e os pesos são dados por

$$w_i = \frac{2^{n+1}n!\sqrt{\pi}}{[H_n'(x_i)]^2}. \tag{5.154}$$

Na Tabela 5.10, temos os pontos da quadratura de Gauss-Hermite para diversos valores de $n$.
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")
```

Tabela 5.10: 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 |

```
for i=1:n
  printf("%1.7E %1.7E\n",x(i),w(i))
endfor
```

**Exemplo 5.6.3.** Na Tabela 5.11, 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 5.11: Resultados referentes ao Exemplo 5.6.3.

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

**E 5.6.1.** Aproxime

$$\int_{-1}^{1} \frac{\operatorname{sen}(x+2) - e^{-x^2}}{\sqrt{1-x^2}}\, dx \tag{5.155}$$

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.

**E 5.6.2.** Aproxime

$$\int_{0}^{\infty} \left(\operatorname{sen}(x+2) - e^{-x^2}\right) e^{-x}\, dx \tag{5.156}$$

usando a quadratura de Gauss-Laguerre com:

a) $n = 3$ pontos.

b) $n = 4$ pontos.

c) $n = 5$ pontos.

**E 5.6.3.** Aproxime

$$\int_{-\infty}^{\infty} \text{sen}(x+2)e^{-x^2} - e^{-2x^2}\, dx \tag{5.157}$$

usando a quadratura de Gauss-Hermite com:

a) $n = 3$ pontos.

b) $n = 4$ pontos.

c) $n = 5$ pontos.

## 5.7 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

$$\int_a^b f(x)\, dx \approx \frac{(b-a)}{n} \sum_{i=1}^{n} f(x_i), \tag{5.158}$$

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 5.7.1.** Na tabela 5.12 temos aproximações $\tilde{I}$ computadas para

$$I = \int_0^1 xe^{-x^2}\, dx \tag{5.159}$$

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`.
Os resultados presentes na Tabela 5.12 podem ser computados no `GNU Octave` com o seguinte código:

```
#inic. gerador randômico
rand("seed",0)
#fun. obj.
```

| $n$ | $\tilde{I}$ | $\lvert I-\tilde{I}\rvert$ |
|---|---|---|
| 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 5.12: Resultados referentes ao Exemplo 5.7.1.

```
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))
```

### 5.7.1 Exercícios

**E 5.7.1.** Use o método de Monte Carlo para obter uma aproximação de

$$\int_{-1}^{1} \frac{\operatorname{sen}(x+2) - e^{-x^2}}{x^2 + \ln(x+2)}\, dx \tag{5.160}$$

com precisão de $10^{-2}$.

# Capítulo 6

# 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

$$y'(t) = f(t,y(t)), \quad t > t_0, \tag{6.1}$$
$$y(t_0) = y_0. \tag{6.2}$$

## 6.1 Método de Euler

Dado um problema de valor inicial

$$y'(t) = f(t,y(t)), \quad t > t_0, \tag{6.3}$$
$$y(t_0) = y_0, \tag{6.4}$$

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

$$\frac{y(t+h) - y(t)}{h} \approx f(t,y) \tag{6.5}$$
$$\Rightarrow y(t+h) \approx y(t) + hf(t,y(t)). \tag{6.6}$$

Ou seja, se conhecermos a solução $y$ no tempo $t$, então (6.6) 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 $y(t_0 + h)$, (6.6) 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

$$y^{(1)} = y_0, \tag{6.7}$$

$$y^{(i+1)} = y^{(i)} + hf(t^{(i)}, y^{(i)}), \tag{6.8}$$

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 6.1.1.** Consideremos o seguinte problema de valor inicial

$$y' - y = \text{sen}(t), t > 0 \tag{6.9}$$

$$y(0) = \frac{1}{2}. \tag{6.10}$$

A solução analítica deste é

$$y(t) = e^t - \frac{1}{2}\,\text{sen}(t) - \frac{1}{2}\cos(t). \tag{6.11}$$

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 (6.35) na forma

$$y' = y + \text{sen}(t) =: f(t, y). \tag{6.12}$$

Então, escolhendo $h = 0{,}1$, a iteração do método de Euler (6.7)-(6.8) nos fornece o método de Euler com passo $h = 0{,}1$

$$y^{(1)} = 0{,}5 \tag{6.13}$$

$$\begin{aligned} y^{(2)} &= y^{(1)} + hf(t^{(1)}, y^{(1)}) \\ &= 0{,}5 + 0{,}1[0{,}5 + \text{sen}(0)] \\ &= 0{,}55 \end{aligned} \tag{6.14}$$

$$\begin{aligned} y^{(3)} &= y^{(2)} + hf(t^{(2)}, y^{(2)}) \\ &= 0{,}55 + 0{,}1[0{,}55 + \text{sen}(0{,}1)] \\ &= 6{,}14983\text{E}{-}01 \end{aligned} \tag{6.15}$$

$$\vdots$$

$$y^{(11)} = 1{,}85259 \tag{6.16}$$

Na Figura 6.1, temos os esboços das soluções analítica e numérica.
As aproximações obtidas neste exemplo podem ser computadas no `GNU Octave` com o seguinte código:

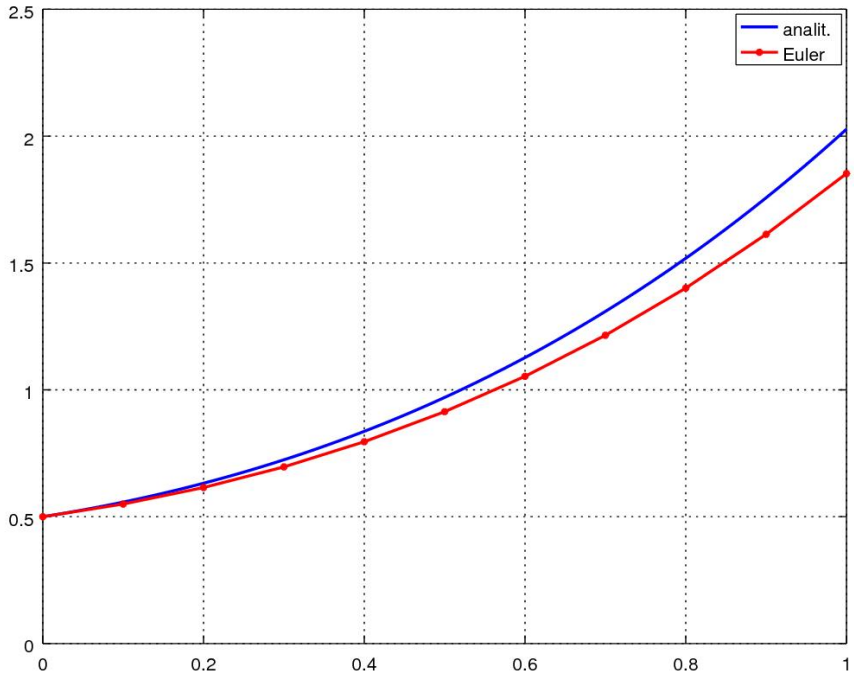


Figura 6.1: Esboço das soluções referente ao Exemplo 6.1.2.

```
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")
```

### 6.1.1 Análise de consistência e convergência

O método de Euler com passo $h$ aplicado ao problema de valor inicial (6.3)-(6.4), pode ser escrito da seguinte forma

$$\tilde{y}(t^{(1)};h) = y_0, \tag{6.17}$$

$$\tilde{y}(t^{(i+1)};h) = \tilde{y}(t^{(i)};h) + h\Phi(t^{(i)},\tilde{y}(t^{(i)});h), \tag{6.18}$$

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

$$\Phi(t,y;h) := f(t,y(t)). \tag{6.19}$$

Agora, considerando a solução exata $y(t)$ de (6.3)-(6.4), introduzimos

$$\Delta(t,y;h) := \begin{cases} \frac{y(t+h)-y(t)}{h} & , h \neq 0, \\ f(t,y(t)) & , h = 0, \end{cases} \tag{6.20}$$

Com isso, vamos analisar o chamado **erro de discretização local**

$$\tau(t,y;h) := \Delta(t,y;h) - \Phi(t,y;h), \tag{6.21}$$

a qual estabelece uma medida quantitativa com que a solução exata $y(t)$ no tempo $t+h$ satisfaz a iteração de Euler.

**Definição 6.1.1.** (Consistência) Um método de passo simples (6.17)-(6.18) é dito consistente quando

$$\lim_{h\to 0} \tau(t,y;h) = 0, \tag{6.22}$$

ou, equivalentemente, quando

$$\lim_{h\to 0} \Phi(t,y;h) = f(t,y). \tag{6.23}$$

**Observação 6.1.1.** Da Definição 6.1.1, temos que o método de Euler é consistente.

A **ordem do erro de discretização local** de um método de passo simples (6.17)-(6.18) é dita ser $p$, quando

$$\tau(t,y;h) = O(h^p). \tag{6.24}$$

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.

$$y(t+h) = y(t) + hy'(t) + \frac{h^2}{2}y''(t) + \frac{h^3}{6}y'''(t+\theta h),\ 0<\theta<1. \tag{6.25}$$

Como $y(t) = f(t,y(t))$ e assumindo a devida suavidade de $f$, temos

$$\begin{aligned} y''(t) &= \frac{d}{dt}f(t,y(t)) && (6.26)\\ &= f_t(t,y) + f_y(t,y)y' && (6.27)\\ &= f_t(t,y) + f_y(t,y)f(t,y). && (6.28)\end{aligned}$$

Então,

$$\Delta(t,y;h) = f(t,y(t)) + \frac{h}{2}[f_t(t,y) + f_y(t,y)f(t,y)] + O(h^2). \tag{6.29}$$

Portanto, para o método de Euler temos

$$\begin{aligned} \tau(t,y;h) &:= \Delta(t,y;h) - \Phi(t,y;h) && (6.30)\\ &= \frac{h}{2}[f_t(t,y) + f_y(t,y)f(t,y)] + O(h^2) && (6.31)\\ &= O(h). && (6.32)\end{aligned}$$

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**

$$e(t;h_n) := \tilde{y}(t;h_n) - y(t), \quad h_n := \frac{t-t_0}{n}. \tag{6.33}$$

E, com isso, dizemos que o método é **convergente** quando

$$\lim_{n\to\infty} e(t,h_n) = 0, \tag{6.34}$$

bem como, dizemos que o método tem erro de discretização global de ordem $h^p$ quando $e(t,h_n) = O(h^p)$.

**Observação 6.1.2.** 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 6.1.2.** Consideremos o seguinte problema de valor inicial

$$y' - y = \text{sen}(t), t > 0 \tag{6.35}$$

$$y(0) = \frac{1}{2}. \tag{6.36}$$

Na Tabela 6.1, 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 6.1: Resultados referentes ao Exemplo **??**

Os resultados mostrados na Tabela 6.1 podem ser computados no `GNU Octave` com o auxílio do seguinte código:

```
f = @(t,y) y+sin(t);

h=1e-2;
n=fix(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

**E 6.1.1.** Considere o seguinte problema de valor inicial

$$y' + e^{-y^2+1} = 2, \quad t > 1, \tag{6.37}$$
$$y(1) = -1. \tag{6.38}$$

Use o método de Euler com passo $h = 0{,}1$ para computar o valor aproximado de $y(2)$.

## 6.2 Métodos de Runge-Kutta

Os métodos de Runge-Kutta de $s$-estágios são métodos de passo simples da seguinte forma

$$y^{(i+1)} = y^{(i)} + h(c_1 k_1 + \cdots + c_s k_s) \tag{6.39}$$

onde

$$k_1 := f(t^{(i)}, y^{(i)}), \tag{6.40}$$
$$k_2 := f(t^{(i)} + \alpha_2 h, y^{(i)} + h\beta_{21} k_1), \tag{6.41}$$
$$k_3 := f(t^{(i)} + \alpha_3 h, y^{(i)} + h(\beta_{31} k_1 + \beta_{32} k_2)), \tag{6.42}$$
$$\vdots \tag{6.43}$$
$$k_s := f(t^{(i)} + \alpha_s h, y^{(i)} + h(\beta_{s1} k_1 + \cdots + \beta_{s,s-1} k_{s-1})), \tag{6.44}$$

$t^{(i)} = t_0 + (i-1)h$ 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].

### 6.2.1 Métodos de Runge-Kutta de ordem 2

Precisamos apenas de 2 estágios para obtermos métodos de Runge-Kutta de ordem 2. Portanto, assumimos

$$y^{(i+1)} = y^{(i)} + h\left[c_1 f(t^{(i)},y^{(i)}) + c_2 f(t^{(i)} + \alpha_2 h, y^{(i)} + h\beta_{21} f(t^{(i)},y^{(i)}))\right]. \tag{6.45}$$

Neste caso, o erro de discretização local é dado por

$$\tau(t,y;h) = \Delta(t,y;h) - \Phi(t,y;h), \tag{6.46}$$

onde, da equação (6.29) temos

$$\Delta(t,y;h) = f(t,y(t)) + \frac{h}{2}[f_t(t,y) + f_y(t,y)f(t,y)] + O(h^2) \tag{6.47}$$

e de (6.45)

$$\Phi(t,y;h) = c_1 f(t,y) + c_2 f(t + \alpha_2 h, y + h\beta_{21} f(t,y)) \tag{6.48}$$

Agora, tomando a expansão de série de Taylor em torno de $t$ de $\Phi(t,y;h)$, temos

$$\Phi(t,y;h) = (c_1 + c_2)f(t,y) + c_2 h[\alpha_2 f_t(t,y) + \beta_{21} f_y(t,y)f(t,y)) + O(h^2). \tag{6.49}$$

Então, por comparação de (6.47) e (6.49), temos

$$c_1 + c_2 = 1 \tag{6.50}$$

$$c_2\alpha_2 = \frac{1}{2} \tag{6.51}$$

$$c_2\beta_{21} = \frac{1}{2}. \tag{6.52}$$

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

$$c_1 = 0, \quad c_2 = 1, \quad \alpha_2 = \frac{1}{2}, \quad \beta_{21} = \frac{1}{2}. \tag{6.53}$$

Logo, a iteração do método do ponto médio é

$$y^{(1)} = y_0 \tag{6.54}$$

$$y^{(i+1)} = y^{(i)} + hf\left(t^{(i)} + \frac{h}{2}, y^{(i)} + \frac{h}{2} f(t^{(i)}, y^{(i)}\right). \tag{6.55}$$

**Exemplo 6.2.1.** Consideremos o seguinte problema de valor inicial

$$y' - y = \text{sen}(t), t > 0 \tag{6.56}$$

$$y(0) = \frac{1}{2}. \tag{6.57}$$

Na Tabela 6.2, 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 6.2: Resultados referentes ao Exemplo 6.2.1.

Os resultados mostrados na Tabela 6.2 podem ser computados no `GNU Octave` com o auxílio do seguinte código:

```
f = @(t,y) y+sin(t);

h=1e-1;
n=fix(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

$$c_1 = \frac{1}{2}, \quad c_2 = \frac{1}{2}, \quad \alpha_2 = 1, \quad \beta_{21} = 1. \tag{6.58}$$

Logo, a iteração do método de Euler modificado é

$$y^{(1)} = y_0 \tag{6.59}$$

$$y^{(i+1)} = y^{(i)} + \frac{h}{2}\left[f(t^{(i)},y^{(i)}) + f(t^{(i)} + h,y^{(i)} + hf(t^{(i)},y^{(i)})\right]. \tag{6.60}$$

**Exemplo 6.2.2.** Consideremos o seguinte problema de valor inicial

$$y' - y = \text{sen}(t), t > 0 \tag{6.61}$$

$$y(0) = \frac{1}{2}. \tag{6.62}$$

Na Tabela 6.3, 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 6.3: Resultados referentes ao Exemplo 6.2.2

Os resultados mostrados na Tabela 6.3 podem ser computados no `GNU Octave` com o auxílio do seguinte código:

```
f = @(t,y) y+sin(t);

h=1e-1;
n=fix(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)))
```

### 6.2.2 Método de Runge-Kutta de ordem 4

Um dos métodos de Runge-Kutta mais empregados é o seguinte método de ordem 4:

$$y^{(i+1)} = y^{(i)} + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4), \tag{6.63}$$

onde

$$k_1 := f(t^{(i)}, y^{(i)}), \tag{6.64}$$
$$k_2 := f(t^{(i)} + h/2, y^{(i)} + hk_1/2), \tag{6.65}$$
$$k_3 := f(t^{(i)} + h/2, y^{(i)} + hk_2/2), \tag{6.66}$$
$$k_4 := f(t^{(i)} + h, y^{(i)} + hk_3), \tag{6.67}$$

$t^{(i)} = t_0 + (i-1)h$ e $y^{(1)} = y_0$.

**Exemplo 6.2.3.** Consideremos o seguinte problema de valor inicial

$$y' - y = \text{sen}(t), t > 0 \tag{6.68}$$
$$y(0) = \frac{1}{2}. \tag{6.69}$$

Na Tabela 6.4, 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 6.4: Resultados referentes ao Exemplo 6.2.3

Os resultados mostrados na Tabela 6.4 podem ser computados no `GNU Octave` com o auxílio do seguinte código:

```
f = @(t,y) y+sin(t);

h=1e-4;
n=fix(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)))
```

### 6.2.3 Exercícios

**E 6.2.1.** Considere o seguinte problema de valor inicial

$$y' + e^{-y^2+1} = 2, \quad t > 1, \tag{6.70}$$
$$y(1) = -1. \tag{6.71}$$

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.

## 6.3 Método adaptativo com controle de erro

Consideremos um problema de valor inicia

$$y'(t) = f(t,y(t)), \quad t > t_0, \tag{6.72}$$
$$y(t_0) = y_0. \tag{6.73}$$

e um método de passo simples

$$y^{(1)} = y_0, \tag{6.74}$$
$$y^{(i+1)}(h^{(i+1)}) = y^{(i)} + h^{(i+1)}\Phi(t^{(i)},y^{(i)};h^{(i+1)}), \tag{6.75}$$

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.

$$|e(t^{(i+1)};h^{(i+1)})| := |y^{(i+1)}(h^{(i+1)}) - y(t^{(i+1)})| \approx TOL. \tag{6.76}$$

Para um método de ordem $h^p$, pode-se mostrar que (veja, [3, Cap. 7, Seç. 7.2])

$$y^{(i+1)}(h^{(i+1)}) = y(t^{(i+1)}) + e_p(t^{(i+1)})(h^{(i+1)})^p, \tag{6.77}$$

onde $e(t^{(i+1)})$ é uma função apropriada. Então, assumindo que $e(t^{(i)};h^{(i)}) = 0$, temos

$$e_p(t^{(i+1)}) = h^{(i+1)}e_p'(t^{(i)}) \tag{6.78}$$

e, portanto, para termos (6.76) impomos que

$$|(h^{(i+1)})^{p+1} e'_p(t^{(i)})| = TOL. \tag{6.79}$$

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 (6.77) temos

$$y^{(i+1)}\left(\frac{h^{(i+1)}}{2}\right) = y(t^{(i+1)}) + e_p(t^{(i+1)})\frac{(h^{(i+1)})^p}{2^p} \tag{6.80}$$

e, então, subtraindo esta de (6.77) temos

$$y^{(i+1)}(h^{(i+1)}) - y^{(i+1)}\left(\frac{h^{(i+1)}}{2}\right) = e_p(t^{(i+1)})\left(\frac{h^{(i+1)}}{2}\right)^p (2^p - 1), \tag{6.81}$$

donde

$$e_p(t^{(i+1)})\left(\frac{h^{(i+1)}}{2}\right)^p = \frac{y^{(i+1)}(h^{(i+1)}) - y^{(i+1)}\left(\frac{h^{(i+1)}}{2}\right)}{2^p - 1}. \tag{6.82}$$

Daí, de (6.78), obtemos

$$e'_p(t^{(i)})h^{(i+1)}\left(\frac{h^{(i+1)}}{2}\right)^p = \frac{y^{(i+1)}(h^{(i+1)}) - y^{(i+1)}\left(\frac{h^{(i+1)}}{2}\right)}{2^p - 1}, \tag{6.83}$$

o que nos fornece a seguinte aproximação de $e'_p(t^{(i)})$

$$e'_p(t^{(i)}) = \frac{1}{(h^{(i+1)})^{p+1}}\frac{2^p}{2^p - 1}\left[y^{(i+1)}(h^{(i+1)}) - y^{(i+1)}\left(\frac{h^{(i+1)}}{2}\right)\right]. \tag{6.84}$$

Assim sendo, de (6.79) temos que o passo $h^{(i+1)}$ apropriado é tal que

$$\frac{2^p}{2^p - 1}\left|y^{(i+1)}(h^{(i+1)}) - y^{(i+1)}\left(\frac{h^{(i+1)}}{2}\right)\right| \approx TOL. \tag{6.85}$$

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

$$\frac{2^p}{2^p - 1}\left|y^{(i+1)}(h) - y^{(i+1)}\left(\frac{h}{2}\right)\right| > TOL, \tag{6.86}$$

tomamos sucessivas divisões de $h$ por 2, até satisfazermos (6.85). Obtido o $h$ que satisfaz (6.85), temos computado $y^{(i+1)}$ com $h^{(i+1)} = h$.

**Exemplo 6.3.1.** Consideremos o seguinte problema de valor inicial

$$y' - y = \text{sen}(t), t > 0 \tag{6.87}$$

$$y(0) = \frac{1}{2}. \tag{6.88}$$

A Figura 6.2 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 (6.1) fica evidente o controle do erro.

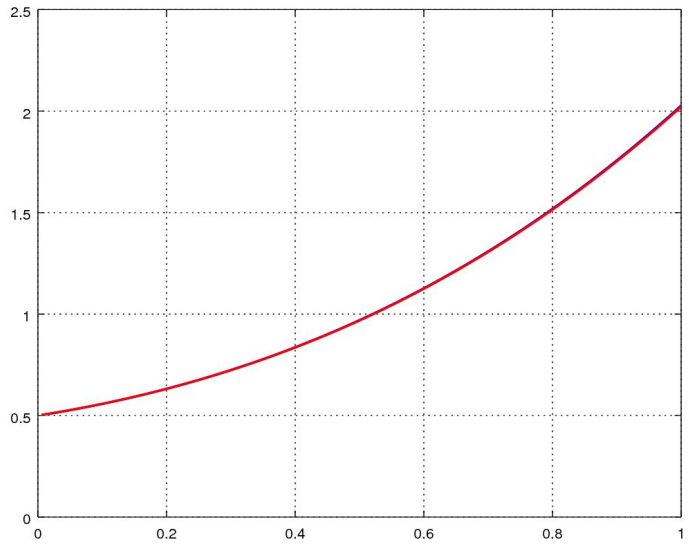


Figura 6.2: Resultados referentes ao Exemplo 6.3.1.

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;
```

```
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

**E 6.3.1.** Considere o seguinte problema de valor inicial

$$y' + e^{-y^2+1} = 2, \quad t > 1, \tag{6.89}$$
$$y(1) = -1. \tag{6.90}$$

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 $TOL = 10^{-4}$.

## 6.4 Métodos de passo múltiplo

Dado um problema de valor inicial

$$y'(t) = f(t,y(t)), \quad t > t_0, \tag{6.91}$$
$$y(t_0) = y_0. \tag{6.92}$$

temos

$$y(t) = y(t_0) + \int_{t_0}^{t} f(s,y(s))\, ds. \tag{6.93}$$

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

$$y\left(t^{(i+k)}\right) = y\left(t^{(i-j)}\right) + \int_{t^{(i-j)}}^{t^{(i+k)}} f(s,y(s))\, ds. \tag{6.94}$$

A ideia é, então, aproximar a integral acima por uma quadratura numérica. Seguindo as regras de Newton-Cotes (veja, Cap. 5 Seç. 5.1), escolhemos os nodos da quadratura como $x_l = t^{(i-l+1)}$, $l = 1, 2, \ldots, m$, e, então

$$\int_{t^{(i-j)}}^{t^{i+k}} f(x,y(x))\, dx \approx \sum_{l=1}^{m} f\left(x_l, y(x_l)\right) w_l, \tag{6.95}$$

e

$$w_l = \int_{t^{(i-j)}}^{t^{(i+k)}} \prod_{\substack{p=1 \\ p \neq l}}^{m} \frac{x - x_p}{x_l - x_p}\, dx. \tag{6.96}$$

Agora, fazendo a mudança de variável $u = (x - t^{(i)})/h$, obtemos

$$w_l = h \int_{-j}^{k} \prod_{\substack{p=1 \\ p \neq l}}^{m} \frac{u + p - 1}{-l + p}\, du \tag{6.97}$$

Assim sendo, temos o seguinte esquema numérico

$$y^{(i+k)} = y^{(i-j)} + h \sum_{l=1}^{m} c_l f(t^{(i-l+1)}, y^{(i-l+1)}), \tag{6.98}$$

onde

$$c_l = \int_{-j}^{k} \prod_{\substack{p=1 \\ p \neq l}}^{m} \frac{s + p - 1}{-l + p}\, ds. \tag{6.99}$$

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]).

### 6.4.1 Métodos de Adams-Bashforth

Métodos de Adams-Bashforth são métodos de passo múltiplo obtidos ao escolhermos $j = 0$ e $k = 1$ no esquema numérico (6.98). 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 (6.99), temos

$$c_1 = \int_0^1 s + 1\, ds = \frac{3}{2} \tag{6.100}$$

e

$$c_2 = \int_0^1 -s\, ds = -\frac{1}{2}. \tag{6.101}$$

Então, de (6.98) temos a iteração do **método de Adams-Bashforth de** 2 **passos**:

$$y^{(1)} = y_0, \tag{6.102}$$

$$y^{(i+1)} = y^{(i)} + \frac{h}{2}\left[3f(t^{(i)},y^{(i)}) - f(t^{(i-1)},y^{(i-1)})\right], \tag{6.103}$$

com $t^{(i)} = t_0 + (i-1)h$.

**Exemplo 6.4.1.** Consideremos o seguinte problema de valor inicial

$$y' - y = \text{sen}(t), t > 0 \tag{6.104}$$

$$y(0) = \frac{1}{2}. \tag{6.105}$$

Na Tabela 6.5, 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 6.5: Resultados referentes ao Exemplo 6.4.1

Os resultados mostrados na Tabela 6.5 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 (6.99) obtemos, de (6.98), a iteração do **método de Adams-Bashforth de** 3 **passos**:

$$y^{(1)} = y_0, \tag{6.106}$$

$$y^{(i+1)} = y^{(i)} + \frac{h}{12}\left[23f(t^{(i)},y^{(i)}) - 16f(t^{(i-1)},y^{(i-1)}) + 5f(t^{(i-2)},y^{(i-2)})\right], \tag{6.107}$$

com $t^{(i)} = t_0 + (i-1)h$.

**Exemplo 6.4.2.** Consideremos o seguinte problema de valor inicial

$$y' - y = \text{sen}(t), t > 0 \tag{6.108}$$

$$y(0) = \frac{1}{2}. \tag{6.109}$$

Na Tabela 6.6, 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 6.6: Resultados referentes ao Exemplo 6.4.2

Os resultados mostrados na Tabela 6.6 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 (6.99) obtemos, de (6.98), a iteração do **método de Adams-Bashforth de** 4 **passos**:

$$y^{(1)} = y_0, \tag{6.110}$$

$$\begin{aligned} y^{(i+1)} = y^{(i)} + \frac{h}{24}\Big[&55f(t^{(i)},y^{(i)}) \\ &-59f(t^{(i-1)},y^{(i-1)}) + 37f(t^{(i-2)},y^{(i-2)}) \\ &-9f(t^{(i-3)},y^{(i-3)})\Big], \end{aligned} \tag{6.111}$$

com $t^{(i)} = t_0 + (i-1)h$.

**Exemplo 6.4.3.** Consideremos o seguinte problema de valor inicial

$$y' - y = \text{sen}(t), t > 0 \tag{6.112}$$

$$y(0) = \frac{1}{2}. \tag{6.113}$$

Na Tabela 6.7, 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 6.7: Resultados referentes ao Exemplo 6.4.3

Os resultados mostrados na Tabela 6.7 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

**E 6.4.1.** Considere o seguinte problema de valor inicial

$$y' + e^{-y^2+1} = 2, \quad t > 1, \tag{6.114}$$

$$y(1) = -1. \tag{6.115}$$

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 7

# 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

$$\alpha(x)u'' + \beta(x)u' + \gamma(x)u = f(x), \quad c_1 < x < c_2, \tag{7.1}$$
$$\eta_1 u'(c_1) + \theta_1 u(c_1) = g_1 \tag{7.2}$$
$$\eta_2 u'(c_2) + \theta_2 u(c_2) = g_2 \tag{7.3}$$

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 $\eta_1$ e $\theta_1$ não simultaneamente nulos, bem como, os coeficientes $\eta_2$ e $\theta_2$, também, não simultaneamente nulos.

## 7.1 Método de diferenças finitas

Consideramos o seguinte problema linear de valor de contorno

$$\alpha(x)u'' + \beta(x)u' + \gamma(x)u = f(x), \quad c_1 < x < c_2, \tag{7.4}$$
$$\eta_1 u'(c_1) + \theta_1 u(c_1) = g_1 \tag{7.5}$$
$$\eta_2 u'(c_2) + \theta_2 u(c_2) = g_2 \tag{7.6}$$

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 $\eta_1$ e $\theta_1$ não simultaneamente nulos, bem como, os coeficientes $\eta_2$ e $\theta_2$, também, não simultaneamente nulos.

A aproximação pelo método de diferenças finitas de (7.4)-(7.6) 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

$$h = \frac{c_2 - c_1}{n-1}, \tag{7.7}$$

e os pontos da partição podem ser indexados da seguinte forma

$$x_i = c_1 + (i-1)h. \tag{7.8}$$

### 2. Discretização das equações

Começando pela equação (7.4), no ponto $x = x_i$ temos

$$\alpha(x_i)u''(x_i) + \beta(x_i)u'(x_i) + \gamma(x_i)u(x_i) = f(x_i) \tag{7.9}$$

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.

$$u''(x_i) = \underbrace{\frac{u(x_i - h) - 2u(x_i) + u(x_i + h)}{h^2}}_{D^2_{0,h^2}u(x_i)} + O(h^2). \tag{7.10}$$

A primeira derivada de $u$ também pode ser substituída pela fórmula de diferenças finitas central de ordem $h^2$, i.e.

$$u'(x_i) = \underbrace{\frac{u(x_i + h) - u(x_i - h)}{2h}}_{D_{0,h^2}u(x_i)} + O(h^2). \tag{7.11}$$

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 (7.9), obtemos

$$\alpha(x_i)\left(\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2}\right) + \beta(x_i)\left(\frac{u_{i+1} - u_{i-1}}{2h}\right) + \gamma(x_i)u_i + O(h^2) = f(x_i), \tag{7.12}$$

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

$$\left(\frac{\alpha(x_i)}{h^2} - \frac{\beta(x_i)}{2h}\right)u_{i-1} + \left(\gamma(x_i) - \frac{2\alpha(x_i)}{h^2}\right)u_i + \left(\frac{\alpha(x_i)}{h^2} + \frac{\beta(x_i)}{2h}\right)u_{i+1} = f(x_i), \tag{7.13}$$

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

$$u'(c_1) = \frac{-3u(c_1) + 4u(c_1 + h) - u(c_1 + 2h}{2h} + O(h^2). \tag{7.14}$$

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 (7.5) temos

$$\eta_1\left(\frac{-3u_1 + 4u_2 - u_3}{2h}\right) + \theta_1 u_1 + O(h^2) = g_1. \tag{7.15}$$

Então, desconsiderando o termo do erro de truncamento, obtemos a seguinte equação discreta

$$\left(\theta_1 - \frac{3\eta_1}{2h}\right)u_1 + \frac{2\eta_1}{h}u_2 - \frac{\eta_1}{2h}u_3 = g_1. \tag{7.16}$$

Procedendo de forma análoga para a condição de contorno (7.6), usamos a fórmula de diferenças finitas regressiva de ordem $h^2$ para a derivada $u'(c_2)$, i.e.

$$u'(c_2) = \frac{3u(c_2) - 4u(c_2 - h) + u(c_2 - 2h)}{2h} + O(h^2). \tag{7.17}$$

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 (7.6) obtemos

$$\eta_2 \left( \frac{3u_n - 4u_{n-1} + u_{n-2}}{2h} \right) + \theta_2 u_n + O(h^2) = g_2. \tag{7.18}$$

Então, desconsiderando o termo do erro de truncamento, obtemos

$$\frac{\eta_2}{2h} u_{n-2} - \frac{2\eta_2}{h} u_{n-1} + \left( \theta_2 + \frac{3\eta_2}{2h} \right) u_n = g_2. \tag{7.19}$$

Por fim, as equações (7.16)-(7.19) formam o seguinte problema discretizado pelo método de diferenças finitas

$$\left( \theta_1 - \frac{3\eta_1}{2h} \right) u_1 + \frac{2\eta_1}{h} u_2 - \frac{\eta_1}{2h} u_3 = g_1. \tag{7.20}$$

$$\begin{aligned} &\left( \frac{\alpha(x_i)}{h^2} - \frac{\beta(x_i)}{2h} \right) u_{i-1} + \left( \gamma(x_i) - \frac{2\alpha(x_i)}{h^2} \right) u_i \\ &+ \left( \frac{\alpha(x_i)}{h^2} + \frac{\beta(x_i)}{2h} \right) u_{i+1} = f(x_i),\ i = 2, \ldots, n-1, \end{aligned} \tag{7.21}$$

$$\frac{\eta_2}{2h} u_{n-2} - \frac{2\eta_2}{h} u_{n-1} + \left( \theta_2 + \frac{3\eta_2}{2h} \right) u_n = g_2. \tag{7.22}$$

### 3. Resolução do problema discreto

O problema discreto (7.20)-(7.22) consiste em um sistema linear de $n$ equações com $n$ incógnitas. Na forma matricial temos

$$A\tilde{u} = b \tag{7.23}$$

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:

$$a_{11} = \left(\theta_1 - \frac{3\eta_1}{2h}\right), \tag{7.24}$$

$$a_{12} = \frac{2\eta_1}{h}, \tag{7.25}$$

$$a_{13} = -\frac{\eta_1}{2h}, \tag{7.26}$$

$$a_{i,i-1} = \left(\frac{\alpha(x_i)}{h^2} - \frac{\beta(x_i)}{2h}\right), \ i = 2, \ldots, n-1, \tag{7.27}$$

$$a_{i,i} = \left(\gamma(x_i) - \frac{2\alpha(x_i)}{h^2}\right), \ i = 2, \ldots, n-1, \tag{7.28}$$

$$a_{i,i+1} = \left(\frac{\alpha(x_i)}{h^2} + \frac{\beta(x_i)}{2h}\right), \ i = 2, \ldots, n-1, \tag{7.29}$$

$$a_{n,n-2} = \frac{\eta_2}{2h}, \tag{7.30}$$

$$a_{n,n-1} = -\frac{2\eta_2}{h}, \tag{7.31}$$

$$a_{n,n} = \left(\theta_2 + \frac{3\eta_2}{2h}\right). \tag{7.32}$$

Com isso em mente, a matriz $A$ tem a seguinte estrutura

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & & & & \\ a_{21} & a_{22} & a_{23} & & & & \\ & \ddots & \ddots & \ddots & & & \\ & & a_{i,i-1} & a_{i,i} & a_{i,i+1} & & \\ & & \ddots & \ddots & \ddots & & \\ & & & a_{n-1,n-2} & a_{n-1,n-1} & a_{n-1,n} & \\ & & & a_{n,n-2} & a_{n,n-1} & a_{n,n} \end{bmatrix}. \tag{7.33}$$

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 7.1.1.** Consideremos o seguinte problema de valor de contorno

$$-u'' = \text{sen}(x), \quad 0 \leq x \leq 2, \tag{7.34}$$
$$u(0) = 0, \tag{7.35}$$
$$u(2) = \text{sen}(2). \tag{7.36}$$

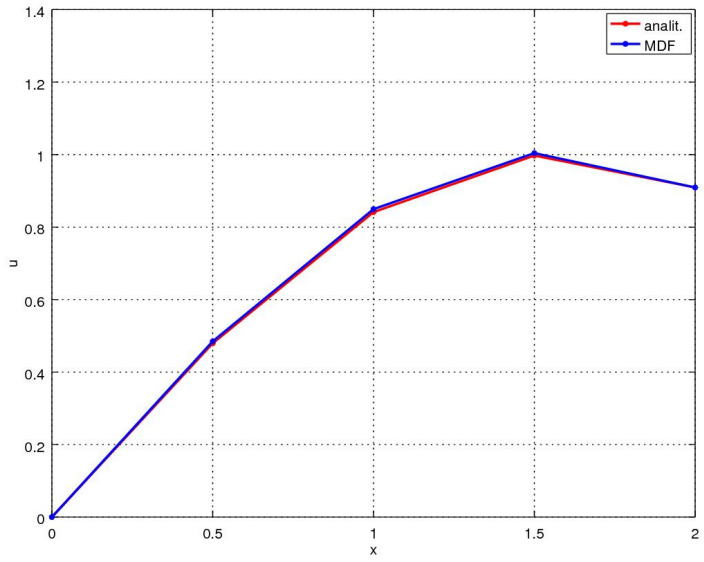


Figura 7.1: Resultado referente ao Exemplo 7.1.1.

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

$$u_1 = 0, \tag{7.37}$$
$$-\frac{1}{h^2}u_{i-1} + \frac{2}{h^2}u_i - \frac{1}{h^2}u_{i+1} = \text{sen}(x_i),\ i = 2, \ldots, n-1, \tag{7.38}$$
$$u_n = \text{sen}(2), \tag{7.39}$$

onde $h = \pi/(n-1)$ e $x_i = (i-1)h$.
Resolvendo este sistema com $h = 0{,}5$ obtemos a solução numérica apresentada na Figura 7.1. Ainda, na Tabela 7.1 temos a comparação na norma $L^2$ da

| $h$ | $n$ | $\|\tilde{u} - u\|_{L^2}$ |
|---|---|---|
| $10^{-1}$ | 21 | 1,0E−03 |
| $10^{-2}$ | 201 | 3,3E−05 |
| $10^{-3}$ | 2001 | 1,0E−06 |

Tabela 7.1: Resultados referentes ao Exemplo 7.1.1.

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

**E 7.1.1.** Considere o seguinte problema de valor inicial

$$-u'' + u' = f(x), \ -1 < x < 1, \tag{7.40}$$
$$u(-1) = 0, \tag{7.41}$$
$$u'(1) = 0, \tag{7.42}$$

onde

$$f(x) = \begin{cases} 1 & , x \leq 0 \\ 0 & , x > 0 \end{cases} \tag{7.43}$$

Use uma aproximação adequada pelo método de diferenças finitas para obter o valor aproximado de $u(0)$ com precisão de 2 dígitos significativos.

# Capítulo 8

# 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.

## 8.1 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

$$u_{xx} + u_{yy} = f(x, y),\ (x, y) \in D, \tag{8.1}$$
$$u(x_{\text{ini}}, y) = 0,\ y_{\text{ini}} \leq y \leq y_{\text{fin}}, \tag{8.2}$$
$$u(x_{\text{fin}}, y) = 0,\ y_{\text{ini}} \leq y \leq y_{\text{fin}}, \tag{8.3}$$
$$u(x, y_{\text{ini}}) = 0,\ x_{\text{ini}} \leq x \leq x_{\text{fin}}, \tag{8.4}$$
$$u(x, y_{\text{fin}}) = 0,,\ x_{\text{ini}} \leq x \leq x_{\text{fin}}, \tag{8.5}$$

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 7), 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

$$x_i := x_{\text{ini}} + (i-1)h_x, \quad h_x = \frac{x_{\text{fin}} - x_{\text{ini}}}{n_x - 1}, \tag{8.6}$$

$$y_j := y_{\text{ini}} + (j-1)h_y, \quad h_y = \frac{y_{\text{fin}} - y_{\text{ini}}}{n_y - 1}, \tag{8.7}$$

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

$$P(\overline{D}) = \{(x_1, y_1), (x_1, y_2), \ldots, (x_i, y_j), \ldots, (x_{n_x}, y_{n_y})\}, \tag{8.8}$$

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

$$u_{xx}(x,y) = \frac{u(x+h,y) - 2u(x,y) + u(x-h,y)}{h^2} + O(h^2), \tag{8.9}$$

$$u_{yy}(x,y) = \frac{u(x,y+h) - 2u(x,y) + u(x,y-h)}{h^2} + O(h^2). \tag{8.10}$$

Daí, denotando $u_{ij} \approx u(x_i, y_j)$ temos

$$u_{xx}(x_i,y_j) = \frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + O(h^2), \tag{8.11}$$

$$u_{yy}(x_i,y_j) = \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} + O(h^2). \tag{8.12}$$

Então, da equação 8.1 temos

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{h^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{h^2} + O(h^2) = f(x_i,y_j). \tag{8.13}$$

Adora, denotando $u_k := u_{j+(i-1)n}$, desprezando o termo do erro de truncamento e rearranjando os termos nesta última equação temos

$$\frac{1}{h^2}u_{k-n} + \frac{1}{h^2}u_{k-1} - \frac{4}{h^2}u_k + \frac{1}{h^2}u_{k+1} + \frac{1}{h^2}u_{k+n} = f(x_i,y_j), \tag{8.14}$$

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 (8.2)-(8.5):

$$u_{1,j} = 0, \quad u_{n,j} = 0, \tag{8.15}$$
$$u_{i,1} = 0, \quad u_{i,n} = 0, \tag{8.16}$$

$i,j = 1,2,\ldots,n$.

Com isso, o problema discreto obtido da aplicação do método de diferenças finitas consiste no sistema linear de $n^2$ equações (8.14)-e(8.16) para as $n^2$ incógnitas $u_k$, $k = 1,2,\ldots,n^2$.

### 3. Resolução do problema discreto

O problema discreto (8.14)-(8.16) pode ser escrito na forma matricial

$$A\tilde{u} = b, \tag{8.17}$$

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

$$i = 1,n,\ j = 1,2,\ldots,n: \ \ b_k = 0, \tag{8.18}$$
$$i = 1,2,\ldots,n,\ j = 1,n: \ \ b_k = 0, \tag{8.19}$$
$$i,j = 2,3,\ldots,n-1: \ \ b_k = f(x_i,y_j). \tag{8.20}$$

Além disso, a matriz dos coeficientes $A$ é tal que

$$i = 1,n,\ j = 1,2,\ldots,n: \ a_{k,k} = 1, \tag{8.21}$$
$$i = 1,2,\ldots,n,\ j = 1,n: \ a_{k,k} = 1, \tag{8.22}$$
$$i,j = 2,3,\ldots,n-1: \ a(k,k-n) = \frac{1}{h^2}, \tag{8.23}$$
$$a(k,k-1) = \frac{1}{h^2}, \tag{8.24}$$
$$a(k,k) = -\frac{4}{h^2}, \tag{8.25}$$
$$a(k,k+1) = \frac{1}{h^2}, \tag{8.26}$$
$$a(k,k+n) = \frac{1}{h^2}. \tag{8.27}$$

Assim sendo, basta empregarmos um método apropriado para resolver o sistema linear (8.17) para obter a solução aproximada de $u$ nos nodos $(x_i, y_j)$.

**Exemplo 8.1.1.** Consideremos o seguinte problema

$$u_{xx} + u_{yy} = -\operatorname{sen}(x)\operatorname{sen}(y),\ (x,y) \in (0,\pi) \times (0,\pi), \tag{8.28}$$
$$u(0,y) = 0,\ y \in [0,\pi], \tag{8.29}$$
$$u(\pi,y) = 0,\ y \in [0,\pi], \tag{8.30}$$
$$u(x,0) = 0,\ x \in [0,\pi], \tag{8.31}$$
$$u(x,\pi) = 0,\ x \in [0,\pi]. \tag{8.32}$$

A Figura 8.1 mostra um esboço do gráfico da solução aproximada obtida pelo método de diferenças finitas apresentado acima (equações (8.14)-(8.16)) 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 8.1: Resultados referentes ao Exemplo 8.1.1.

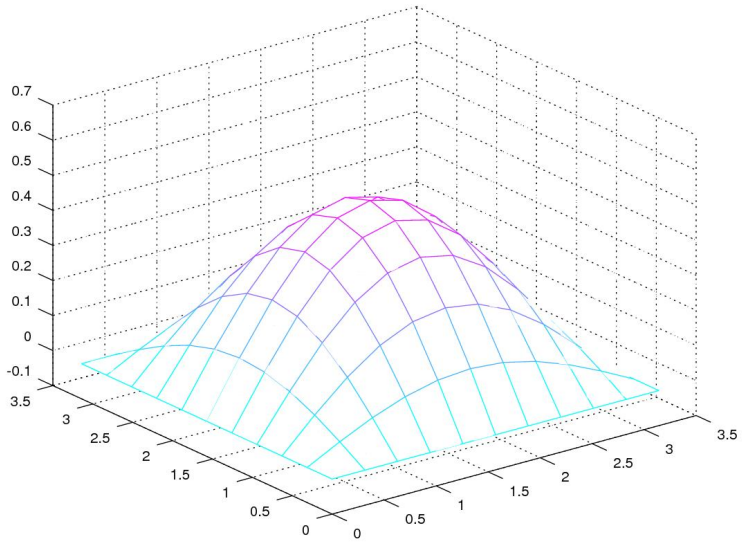


Figura 8.1: Resultado referente ao Exemplo 8.1.1.

Na Tabela 8.1 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

**E 8.1.1.**

$$-(u_{xx} + u_{yy}) = f(x),\ (x,y) \in (0,1)^2, \tag{8.33}$$
$$u(0,y) = 0,\ y \in [0,1], \tag{8.34}$$
$$u(1,y) = 0,\ y \in [0,1], \tag{8.35}$$
$$\left.\frac{\partial u}{\partial y}\right|_{y=0} = 0,\ x \in [0,1], \tag{8.36}$$
$$u(x,1) = 0,\ x \in [0,1]. \tag{8.37}$$

onde

$$f(x) = \begin{cases} 1 & ,x \leq 0{,}5 \\ 0 & ,x > 0{,}5 \end{cases} \tag{8.38}$$

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.

## 8.2 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

$$u_t - \alpha u_{xx} = f(t,x),\ t > t_0,\ x \in D, \tag{8.39}$$
$$u(t_0,x) = u_0(x),\ x \in D, \tag{8.40}$$
$$u(t, x_{\text{ini}}) = 0,\ t > t_0, \tag{8.41}$$
$$u(t, x_{\text{fin}}) = 0,\ t > t_0 \tag{8.42}$$

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 (8.39), temos

$$\frac{\mathrm{d}}{\mathrm{d}t}u_i - \alpha\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2} = f(t,x_i), \tag{8.43}$$

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

$$\frac{\mathrm{d}}{\mathrm{d}t}u_2 = -\frac{2\alpha}{h^2}u_2 + \frac{\alpha}{h^2}u_3 + f(t,x_2), \tag{8.44}$$
$$\frac{\mathrm{d}}{\mathrm{d}t}u_i = \frac{\alpha}{h^2}u_{i-1} - \frac{2\alpha}{h^2}u_i + \frac{\alpha}{h^2}u_{i+1} + f(t,x_i), \tag{8.45}$$
$$\frac{\mathrm{d}}{\mathrm{d}t}u_{n-1} = \frac{\alpha}{h^2}u_{n-2} - \frac{2\alpha}{h^2}u_{n-1} + f(t,x_{n-1}), \tag{8.46}$$
$$\tag{8.47}$$

onde $i = 3, 4, \ldots, n-2$ e com condições iniciais dadas por (8.40), i.e.

$$u_j(t_0) = u_0(x),\ j = 2, 3, \ldots, n-1. \tag{8.48}$$

Ainda, observamos que o sistema (8.44) pode ser escrito de forma mais compacta como

$$\frac{\mathrm{d}\tilde{u}}{\mathrm{d}t} = A\tilde{u} + \tilde{f}, \tag{8.49}$$

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

$$A = \begin{bmatrix} -\frac{2\alpha}{h^2} & \frac{\alpha}{h^2} & 0 & 0 & 0 & \cdots & 0 & 0 \\ \frac{\alpha}{h^2} & -\frac{2\alpha}{h^2} & \frac{\alpha}{h^2} & 0 & 0 & \cdots & 0 & 0 \\ 0 & \frac{\alpha}{h^2} & -\frac{2\alpha}{h^2} & \frac{\alpha}{h^2} & 0 & \cdots & 0 & 0 \\ 0 & 0 & \ddots & \ddots & \ddots & \cdots & 0 & 0 \\ 0 & 0 & & 0 & 0 & \cdots & \frac{\alpha}{h^2} & -\frac{2\alpha}{h^2} \end{bmatrix}. \tag{8.50}$$

**Discretização temporal**

Aqui, vamos usar o método de Euler (veja, 6.1) para aproximar a solução de (8.50)-(8.48). 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

$$\tilde{u}^{(1)} = \tilde{u}_0 \tag{8.51}$$

$$\tilde{u}^{(k+1)} = \tilde{u}^{(k)} + h_t \left( A\tilde{u}^{(k)} + \tilde{f}^{(k)} \right), \tag{8.52}$$

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

$$\tilde{u}^{(1)} = \tilde{u}_0 \tag{8.53}$$

$$\tilde{u}^{(k+1)} = (I - h_t A)\, \tilde{u}^{(k)} + h_t \tilde{f}^{(k)}. \tag{8.54}$$

**Observação 8.2.1.** O esquema numérico acima é **condicionalmente estável**. Pode-se mostrar a seguinte condição de estabilidade [2, Cap. 12, Seç. 2]:

$$\alpha \frac{h_t}{h^2} \leq \frac{1}{2}. \tag{8.55}$$

**Exemplo 8.2.1.** Consideremos o seguinte problema

$$u_t - u_{xx} = \text{sen}(x),\ t > 0,\ 0 \leq x \leq \pi, \tag{8.56}$$
$$u(0,x) = 0,\ 0 < x < \pi, \tag{8.57}$$
$$u(t,0) = 0,\ t > 0 \tag{8.58}$$
$$u(t,\pi) = 0,\ t > 0. \tag{8.59}$$

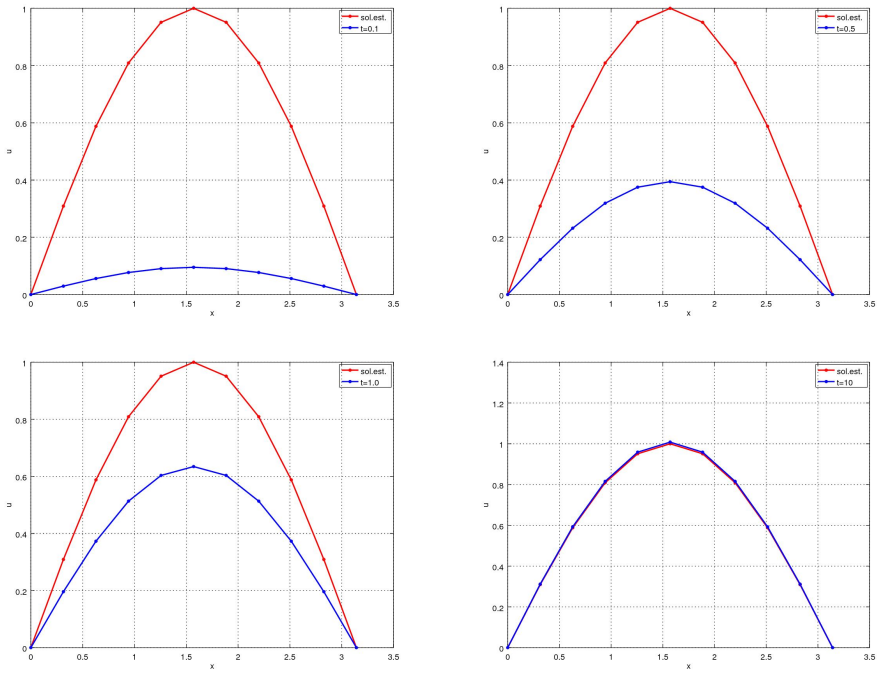


Figura 8.2: Resultados referentes ao Exemplo 8.2.1.

Este problema tem solução estacionário $u(x) = \text{sen}(x)$. Na Figura 8.2, 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

**E 8.2.1.** Considere o seguinte problema

$$u_t - u_{xx} = f(x),\ t > 0,\ 0 \leq x \leq 1, \tag{8.60}$$
$$u(0,x) = 0,\ 0 < x < 1, \tag{8.61}$$
$$u(t,0) = 1,\ t > 0 \tag{8.62}$$
$$u(t,1) = 0,\ t > 0. \tag{8.63}$$

com

$$f(x) = \begin{cases} 1 & ,x \leq 0{,}5, \\ 0 & ,x > 0{,}5 \end{cases} \tag{8.64}$$

Use o método de diferenças finitas para obter uma aproximação de $u(1,0.5)$ com dois dígitos significativos de precisão.

## 8.3 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

$$u_{tt} - \alpha u_{xx} = 0,\ t > t_0,\ x \in D, \tag{8.65}$$
$$u(x,t_0) = f(x),\ x \in D, \tag{8.66}$$
$$\frac{\partial u}{\partial t}(x,t_0) = g(x),\ x \in D, \tag{8.67}$$
$$u(x_{\text{ini}},t) = 0,\ t > t_0, \tag{8.68}$$
$$u(x_{\text{fin}},t) = 0,\ t > t_0 \tag{8.69}$$

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 (8.65). Para tanto, denotamos $u_{i,j} \approx u(x_i,t_j)$ e de (8.65) temos

$$\frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{h_t^2} - \alpha \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{h_x^2} = 0, \tag{8.70}$$

para $j = 2, 3, \ldots, n_t - 1$ e $i = 2, 3, \ldots, n_x - 1$. Rearranjando os termos, temos

$$u_{i,j+1} = \alpha \frac{h_t^2}{h_x^2} u_{i-1,j} + 2\left(1 - \alpha \frac{h_t^2}{h_x^2}\right) u_{i,j} + \alpha \frac{h_t^2}{h_x^2} u_{i+1,j} - u_{i,j-1}, \tag{8.71}$$

para $j = 2, 3, \ldots, n_t - 1$ e $i = 2, 3, \ldots, n_x - 1$.
Agora, das condições de contorno (8.68) e (8.69), temos $u_{1,j} = u_{n_x,j} = 0$, $j = 2, 3, \ldots, n_t$. Com isso, o sistema (8.71) torna-se

$$u_{2,j+1} = 2\left(1 - \alpha \frac{h_t^2}{h_x^2}\right) u_{2,j} + \alpha \frac{h_t^2}{h_x^2} u_{3,j} - u_{2,j-1}, \tag{8.72}$$

$$u_{i,j+1} = \alpha \frac{h_t^2}{h_x^2} u_{i-1,j} + 2\left(1 - \alpha \frac{h_t^2}{h_x^2}\right) u_{i,j} + \alpha \frac{h_t^2}{h_x^2} u_{i+1,j} - u_{i,j-1}, \tag{8.73}$$

$$u_{n_x-1,j+1} = \alpha \frac{h_t^2}{h_x^2} u_{n_x-2,j} + 2\left(1 - \alpha \frac{h_t^2}{h_x^2}\right) u_{n_x-1,j} - u_{i,j-1}, \tag{8.74}$$

$$\tag{8.75}$$

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

$$\begin{bmatrix} u_{2,j+1} \\ u_{3,j+1} \\ \vdots \\ u_{n_x-1,j+1} \end{bmatrix} = \begin{bmatrix} 2(1-\lambda) & \lambda & 0 & \cdots & 0 \\ \lambda & 2(1-\lambda) & \lambda & \cdots & 0 \\ 0 & \ddots & \ddots & \ddots & \cdots \\ 0 & 0 & \cdots & \lambda & 2(1-\lambda) \end{bmatrix} \begin{bmatrix} u_{2,j} \\ u_{3,j} \\ \vdots \\ u_{n_x-1,j} \end{bmatrix} - \begin{bmatrix} u_{2,j-1} \\ u_{3,j-1} \\ \vdots \\ u_{n_x-1,j-1} \end{bmatrix}, \tag{8.76}$$

para $j = 2, 3, \cdots, n_t - 1$, onde $\lambda := \alpha h_t^2 / h_x^2$.
Esta última equação (8.76) 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 (8.66), da qual temos

$$u_{i,1} = f(x_i),\ i = 2, 3, \ldots, n_t. \tag{8.77}$$

Agora, usando a fórmula de diferenças finitas progressiva de ordem 1 na condições inicial (8.67), obtemos

$$u_{i,2} = u_{i,1} + h_t g(x_i),\ i = 2, 3, \ldots, n_t. \tag{8.78}$$

Com tudo isso, observamos que as equações (8.77), (8.78) e (8.76), nesta ordem, nos fornece um algoritmo iterativo no tempo para computar as aproximações da solução $u$.

**Observação 8.3.1.** Pode-se mostrar a seguinte condição de estabilidade

$$\alpha \frac{h_t^2}{h_x^2} \leq 1. \tag{8.79}$$

**Exemplo 8.3.1.** Consideremos o seguinte problema

$$u_{tt} - u_{xx} = 0,\ t > 0,\ 0 < x < 1, \tag{8.80}$$
$$u(0,x) = x(1-x),\ 0 < x < 1, \tag{8.81}$$
$$u_t(0,x) = 0,\ 0 < x < 1, \tag{8.82}$$
$$u(t,0) = 0,\ t > 0 \tag{8.83}$$
$$u(t,\pi) = 0,\ t > 0. \tag{8.84}$$

Na Figura 8.3, 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;
```

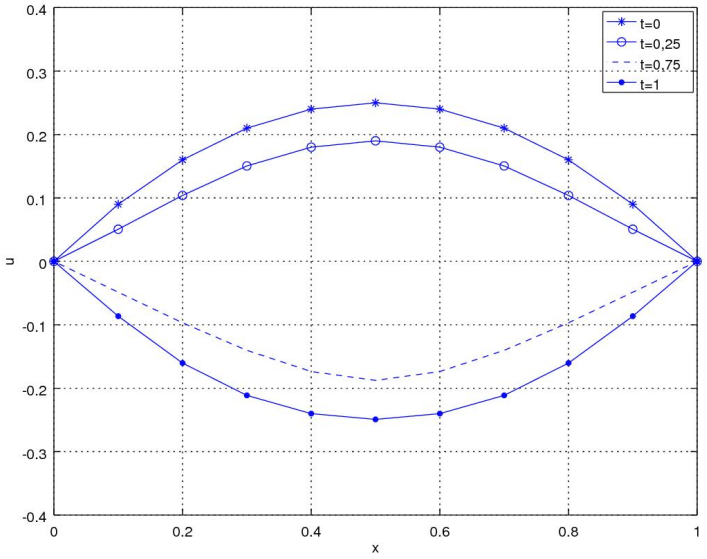


Figura 8.3: Resultados referentes ao Exemplo 8.3.1.

```
lambda = ht^2/hx^2;

#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

**E 8.3.1.** Considere o seguinte problema

$$u_{tt} - u_{xx} = 0,\ t > 0,\ 0 < x < 1, \tag{8.85}$$
$$u(0,x) = x(1-x),\ 0 < x < 1, \tag{8.86}$$
$$u_t(0,x) = 1,\ 0 < x < 1, \tag{8.87}$$
$$u(t,0) = 0,\ t > 0 \tag{8.88}$$
$$u(t,\pi) = 0,\ t > 0. \tag{8.89}$$

Use o método de diferenças finitas para obter uma aproximação de $u(0{,}75,\, 1)$ com dois dígitos significativos de precisão.

# Resposta dos Exercícios

**E 1.1.1.** Código. a) 89,15625; b) 11,25; c) 55981; d) $0,\overline{3}$; e) $0,\overline{3}$

**E 1.1.2.** Código. a) $(101101,1)_2$; b) $(0,1\overline{03})_4$

**E 1.1.3.** Código. a) $(1121,022)_4$; b) $(1011,01)_2$

**E 2.1.1.** Código. $c_1 = -1{,}3259$, $c_2 = 8{,}66071\text{E}{-2}$, $\|r(c)\|_2 = 1{,}01390$.

**E 2.1.2.** Código. $c_1 = -4{,}50361\text{E}{-1}$, $c_2 = -2{,}78350\text{E}{-1}$, $c_3 = 1{,}46291$, $c_4 = 2{,}09648$, $\|r(c)\|_2 = 5{,}71346$

**E 2.1.3.** Código. $c_1 = -2{,}76842$, $c_2 = -7{,}17935\text{E}{-1}$, $c_3 = 1{,}37014\text{E}{-1}$, $\|r(c)\|_2 = 2{,}48880\text{E}{+1}$

**E 2.1.4.** Código. $c_1 = 2{,}10131\text{E}{+0}$, $c_2 = -9{,}73859\text{E}{-1}$, $c_3 = 1.25521\text{E}{+0}$

**E 2.2.1.** 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.

**E 2.2.2.** Código. a) $c_1 = 2{,}69971\text{E}{+0}$, $c_2 = -1{,}44723\text{E}{+0}$, $c_3 = 1.24333\text{E}{+0}$; b) $c_1 = 2{,}69971\text{E}{+0}$, $c_2 = -1{,}44723\text{E}{+0}$, $c_3 = 1.24333\text{E}{+0}$

**E 3.1.1.** Código. a) $D_{+,h}f(2{,}5) = 1{,}05949$; b) $D_{-,h}f(2{,}5) = 1{,}05877$; c) $D_{0,h^2}f(2{,}5) = 1{,}05913$;

**E 3.1.2.** 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 |

**E 3.2.1.** Código. a) 7,25162E–2; b) 7.24701E–2; c) 7,24696E–2; d) 7,24696E–2; $h = 10^{-2}$;

**E 3.2.2.** Código. 4,0;

**E 3.3.1.** Código. 1,05913

**E 3.3.2.** 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) $\frac{1}{12h}\left[-25f(x) + 48f(x+h) - 36f(x+2h) + 16f(x+3h) - 3f(x+4h)\right]$

**E 3.3.3.** 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 |

**E 4.1.2.** Código. a) 1,05919; b) 1,05916; c) 1,05913

**E 5.1.1.** Código. a) 3,33647E−1; b) 1,71368E−1; c) 2,79554E−1

**E 5.1.2.** Código. a) 4,02000E−1; b) $1{,}04250E+0$; c) 8,08667E−1

**E 5.1.3.** Use um procedimento semelhante aquele usado para determinar a ordem do erro de truncamento da regra de Simpson.

**E 5.1.4.**

$$\int_a^b f(x)\,dx = \frac{3h}{2}\left[f\left(a + \frac{1}{3}(b-a)\right)\right. \tag{5.44}$$

$$\left. + f\left(a + \frac{2}{3}(b-a)\right)\right] + O(h^3),\ h = \frac{(b-a)}{3} \tag{5.45}$$

**E 5.2.1.** Código. a) 2,69264E−1; b) 2,68282E−1; c) 2,68937E−1

**E 5.2.2.** Código. a) 8,12000E−1; b) 1,03850; c) 8,11667E−1

**E 5.3.1.** Código. 2,68953E−1

**E 5.4.1.** 1

**E 5.4.2.** $x_1 = 0$, $w_1 = 2$

**E 5.5.1.** Código. a) −2,61712E − 1; b) 2,55351E − 1; c) 8,97510E − 2; d) 1,27411E−1; e) 1.21016E−1.

**E 5.5.2.** Código. a) −1,54617E − 1; b) −1,50216E − 1; c) −1,47026E − 1; d) −1,47190E−1; e) −1,47193E−1.

**E 5.5.3.** Código. a) 1,21016E−1; b) 1,21744E−1; c) 1,21744E−1

**E 5.6.1.** Código. a) $-2{,}84951E - 01$; b) 2,66274E−01; c) 1,49496E−01; d) 1,60085E−01; e) 1,59427E−01.

**E 5.6.2.** Código. a) −1,03618E−1; b) −5,56446E−2; c) −4,19168E−2

**E 5.6.3.** Código. a) −1,31347; b) −1,23313; c) −1,26007

**E 5.7.1.** Código. 1,2E−1

**E 6.1.1.** Código. −5,87722E−1

**E 6.2.1.** Código. a) −6,00654E−1; b) −6,00703E−1; c) −5,99608E−1

**E 6.3.1.** Código. −5.99240E−1

**E 6.4.1.** Código. a) −6,00696E−1; b) −5,96694E−1; c) −5,96161E−1

**E 7.1.1.** Código. 7,2E−1

**E 8.1.1.** Código. 2,9E−2

**E 8.2.1.** Código. 5,6E−1

**E 8.3.1.** Código. 6,3E−2

# Referências Bibliográficas


[1] A. Björk. *Numerical methods for least squares problems*. SIAM, 1996.

[2] R.L. Burden, J.D. Faires, and A.M. Burden. *Análise Numérica*. CENGAGE Learning, 10. ed. edition, 2015.

[3] E. Isaacson and H.B. Keller. *Analysis of numerical methods*. Dover, 1994.

[4] J. Nocedal and S.J. Wright. *Numerical optimization*. Springer, 2006.

[5] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. 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 Remissivo