Algoritmos Numéricos
Uma abordagem direta

Bernardo Sunderhus

29 de maio de 2016

Sumário

Prefácio

Structure of book

Add short description about each chapter in this book.

Acknowledgements

- Um agradecimento especial ao professor Thomas W. Rauber que disponibilizou suas anotações pessoais.

1

Definições em computação numérica

1.1 Representação numérica

É possível representar um número sendo um sistema contendo um polinomio e uma base

$$
\begin{aligned}
&número = sinal \times (a_j, a_{j-1}, ..., \bullet, a_{-1}, a_{-2}, ..., a_{-k})_B \\
&sinal \in \{-1, 1\} \\
&B \in \mathbb{N}, B \geq 2
\end{aligned}
$$

Do que foi estabelecido a cima podemo gerar

$$
\begin{aligned}
número &= sinal \times a_j B^j + a_{j-1} B^{j-1} + ... + a_0 B^0 + a_{-1} B^{-1} + ... + a_{-k} B^{-k} \\
&= sinal \times \sum_{i=-k}^{j} a_i B^i
\end{aligned}
$$

1.1.1 Números inteiros e positivos

$$k = 0, \quad sinal = +1$$

Exemplo com base $B = 10$

$$
\begin{aligned}
(1\ 2\ 3) &= 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 \\
(4\ 5\ 6) &= 4 \times 10^2 + 5 \times 10^1 + 6 \times 10^0 \\
(1\ 0\ 0\ 1) &= 1 \times 10^3 + 0 \times 10^2 + 0 \times 10^1 + 1 \times 10^0
\end{aligned}
$$

1.1.2 Números em geral

$$-(1\ 2\ 3\ .\ 5)_{10} = -(1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0 + 4 \times 10^{-1} + 5 \times 10^{-2})$$
$$(A\ .\ B\ C)_{16}{}^1 = 10 \times 16^0 + 11 \times 16^{-1} + 12 \times 16^{-2}$$

1.2 Bases importantes para computação

Base	Name	Digits
2	binary	0,1
8	octal	0,...,7
10	decimal	0,...,9
16	hexadecimal	0,...9,A,...,F

1.3 Algumas trocas de bases

$$(1\ 0\ 1)_2 = (5)_{10}$$
$$(A\ B)_{16} = (1\ 7\ 1)_{10}$$

1.3.1 Números inteiros

número $\in \mathbb{Z}$

Decimal $\longrightarrow$ binário

Dado um número decimal N e sendo b o equivalente binário que queremos
while $N \neq 0$ **do**
  $b \leftarrow b + N\%2$ [2]
  $N \leftarrow N \backslash 2$ [3]
end while

[1]Em hexadecimal os números vão de 0,...9,A,...F

[2]resto da divisão

[3]a divisão exata ou divisão inteira é aquela que assume valores inteiros como resposta. Caso a divisão não gere um número inteiro, esse valor extra é descartado da resposta e é chamada de resto da divisão. ex: $3/2 = 2,\ resto = 1$

exemplo:

$N = 23_{10}$
$N = N \backslash 2 = 11$ $(N\%2 = 1 \longrightarrow b_0 = 1)$
$N = N \backslash 2 = 5$ $(N\%2 = 1 \longrightarrow b_1 = 1)$
$N = N \backslash 2 = 2$ $(N\%2 = 1 \longrightarrow b_2 = 1)$
$N = N \backslash 2 = 1$ $(N\%2 = 0 \longrightarrow b_3 = 0)$
$N = N \backslash 2 = 0$ $(N\%2 = 1 \longrightarrow b_4 = 1)$
return $b = (10111)_2$

Binário $\longrightarrow$ decimal

Dado um texto/vetor binário β e j o comprimento de β
$N \leftarrow \beta(j)$
for $i = j - 1, ..., 0$ **do**
  $N \leftarrow 2 \times N + \beta(i)$
end for

exemplo:

$\beta = (10111)_2 \rightarrow j = 4$
$N_4 \leftarrow 1$
$N_3 \leftarrow 0 + 2 \times 1 \rightarrow 2$
$N_2 \leftarrow 1 + 2 \times 2 \rightarrow 5$
$N_1 \leftarrow 1 + 2 \times 5 \rightarrow 11$
$N_0 \leftarrow 1 + 2 \times 11 \rightarrow 23$
return $N_0 = N = 23_{10}$

1.3.2 Números fracionários

Os métodos a baixo servem para computar somente a parte fracionária, eles assumem que o entrada do algoritmo só tem valor fracionário.

Decimal $\longrightarrow$ binário

Dado um número natural finito N e um número maximo de iterações i
$b \leftarrow$ "0."
$aux \leftarrow 0$
while $N \neq 0$ *and* $aux < i$ **do**
  $N2 \leftarrow N \times 2$
  $digit \leftarrow N2 \backslash 1$
  $b \leftarrow b + digit$
  $N \leftarrow N2 - digit$
  $aux \leftarrow aux + 1$

end while
return b

Binária $\longrightarrow$ decimal

Dado uma string/vetor binário b e j o tamanho de b.
$N \leftarrow b[1]$
for $i = 2, ..., j$ **do**
  $N = 2 * N + b[i]$
end for
return N

1.4 Aritmética de ponto flutuante

$$x = sinal \times (.\ d_1, ..., d_t) \times B^e$$

Representação do subconjunto de $\mathbb{R}$.

Tal representação é impossível usando um número t finito de digitos fracionários, o que gera erro na representaçao e nos cálculos.

$$Sinal \in \{-1, 1\}$$
$$d_1, ..., d_t : \text{mantissa de t dígitos}$$
$$d_j \in \{0, ..., B-1\} \ \ j = 1, ..., t$$
$$d_1 \neq 0$$

Exemplo:

$$B = 10, t = 3, e \in \{-5, 5\}$$
$$\longrightarrow x = \pm\ 0 \bullet d_1\ d_2\ d_3 \ \times\ 10^e$$

Maior e menor número representavel:

$$menor = 0.1000 \times 10^5 \rightarrow 0.000001$$
$$Maior = 0.999 \times 10^5 \rightarrow 99900$$

Possíveis casos para x

Seja $G = \{x \in \mathbb{R} | m \leq |x| \leq M$ e $x \in \mathbb{R}$

1. $x \in G$

   a) Representação exata (sem erros)
   Ex: $1.23_{10} \longrightarrow 0.123 \times 10^1$

   b) Representação aproximada
   Ex: $456.789_{10} \longrightarrow 0.456789 \times 10^3 \xrightarrow{t=3} 0.457 \times 10^3$ arredondado.

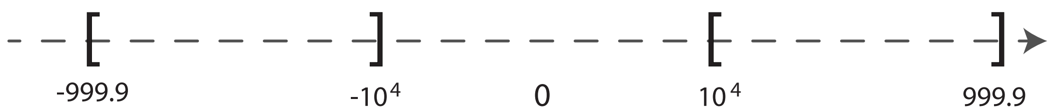
2. $|x| < m$
   Ex: $0.0000000345 = 0.345 \times 10^7 \xrightarrow{e \in \{-5,5\}}$ Não representável, "under flow".

3. $|x| > M$
   Ex: $875000000 = 0.875 \times 10^9 \xrightarrow{e \in \{-5,5\}}$ Não representável "over flow".

1.4.1 Um sistema de ponto flutuante pode ser representado por:

$F(B, t, l, u) = F$(Base, número de digitos, expoente menor, expoente maior)

Ex: $F(10, 4, -3, 3)$
$menor = 0.1 \times 10^3 \longrightarrow 10^{-4}$
$maior = 0.9999 \times 10^3 \longrightarrow 999.9$
região de under flow:$(-10^4, 0) \cup (0, 10^{-4})$
região de over flow:$(-\infty, -999.9) \cup (999.9, \infty)$



*IEEE 754 : F(2,23,-127,128)

1.4.2 Erros

Erros absolutos e relativos

$$
\begin{aligned}
&x : \text{Valor exato}\\
&\overline{x} : \text{Valor aproximado de x}\\
&Def : \text{Erro absoluto } EA_x = x - \overline{x}\\
&|EA_x| \geq 0\\
&Def : \text{Erro relativo } ER_x = \frac{EA_x}{\overline{x}}
\end{aligned}
$$

Exemplo:

$$
\begin{aligned}
&x \in (2112.8, 2113)\ \overline{x} = 2112.9\\
&y \in (5.2, 5.3)\ \overline{y} = 5.3\\
\\
&|EA_x| < 0.1\\
&|ER_x| < 4.7 \times 10^{-5}\\
&|EA_y| < 0.1\\
&|ER_y| < 0.02
\end{aligned}
$$

Representação de x por $\overline{x}$ é relativamente mais precisa do que y por $\overline{y}$

Erro de arredondamento e truncamento

$$
\begin{aligned}
B &= 10, t \text{ digitos}\\
x &= 0.d_1d_2...d_t \times 10^e\\
&= 0.d_1d_2...d_{t-1} \times 10^e\\
&+ 0.000...0d_t \times 10^e\\
&= 0.d_1d_2...d_{t-1} \times 10^e + 0.d_t \times 10^{e-t}\\
&:= f_x \times 10^e + g_x \times 10^{e-t}
\end{aligned}
$$

Exemplo:

$$
\begin{aligned}
t = 4, x &= 234.57\\
&= 0.2345 \times 10^3 + 0.7 \times 10^{-1}\\
&= f_x \times 10^3 + g_x \times 10^{-1}
\end{aligned}
$$

1. Truncamento

$$
\begin{aligned}
&g_x \leftarrow 0, \overline{x} \leftarrow f_x \times 10^e \\
&|EA_x| < 10^{e-t} \\
&|ER_x| < 10^{-t+1}
\end{aligned}
$$

2. Arredondamento

a) $|g_x| < \frac{1}{2}$

$$
\begin{aligned}
g_x \leftarrow 0, \overline{x} = f_x \times 10^e, \\
|EA_x| < \frac{1}{2} \times 10^{e-t} \\
|ER_x| < \frac{1}{2} \times 10^{-t+1}
\end{aligned}
$$

b) $|g_x| \geq \frac{1}{2}$

$$
\begin{aligned}
g_x \leftarrow 10^{e-t}, \\
\overline{x} = f_x \times 10^e + 10^{e-t}, \\
|EA_x| \leq \frac{1}{2} \times 10^{e-t} \\
|ER_x| < \frac{1}{2} \times 10^{-t+1}
\end{aligned}
$$

$\longrightarrow |EA_x| \leq \frac{1}{2} \times 10^{e-t},\ |ER_x| < \frac{1}{2} \times 10^{-t+1}$

Propagação de erros em sequências de operações aritméticas

1. Propagação dos erros dos operandos

a) Adição

Ex: $F(10, 4, -3, 3)$

$X = 1.3749213$ Valor exato

$\overline{X} = 0.1374 \times 10^1 \qquad EA_x = X - \overline{X} = 0.0009213...$

$Y = 573.27482$

$\overline{Y} = 0.5732 \times 10^3 \qquad EA_y = Y = \overline{Y} = 0.0007482$

$X + Y = \overline{X} + \overline{Y} + EA_X + EA_Y$

$\bigoplus$: Operação de adição no sistema de ponto flutuante

$\overline{X} \bigoplus \overline{Y} (EA_x + EA_y$ desconhecidos)

$= \overline{\overline{X} + \overline{Y}}$

$\dfrac{0.0013 \times 10^3}{0.5732 \times 10^3} = 0.5745 \times 10^3$(somente quatro casas na mantissa)

$\overline{X} + \overline{Y} = 1.374 + 573.2 = 574.574$

$\overline{\overline{X} + \overline{Y}} = 574.5$

$EA_{\overline{x}+\overline{Y}} = \overline{X} + \overline{Y} - \overline{\overline{X} + \overline{Y}} = 0.074$

$X + Y = 574.6497413$ Vs

b) Operações:

Operação	Erro abs.	Erro rel.
Adição	$EA_{x+y} = EA_x + EA_y$	$ER_{x+y} = ER_x \times \frac{\overline{x}}{\overline{x}+\overline{y}} + ER_y \times \frac{\overline{y}}{\overline{x}+\overline{y}}$
Subtração	$EA_{x-y} = EA_x - EA_y$	$ER_{x-y} = ER_x \times \frac{\overline{x}}{\overline{x}-\overline{y}} - ER_y \times \frac{\overline{y}}{\overline{x}-\overline{y}}$
Multiplicação	$EA_{xy} \approx \overline{x} \times EA_y + \overline{y} \times EA_x$	$ER_{xy} \approx ER_x + ER_y$
Divisão	$EA_{\frac{x}{y}} \approx \frac{\overline{y}EA_x - \overline{x}EA_y}{\overline{y}^2}$	$ER_{\frac{x}{y}} \approx ER_x - ER_y$

2

Resolução de Sistemas Lineares

Sistema Linear: m equações lineares em n variáveis
Variavel:x_j $j = 1, ..., n$
Equação: $i \rightarrow a_{i1}x_1 + a_{i2}x_2 + ... + a_{in}x_n = b_1$

Em cálculo numérico levaremos em conta na maioria o caso m = n
Em forma matricial temos: $Ax = b$
com

$$A = \begin{bmatrix} a_{11} & a12 & \cdots & a_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \quad x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_n \end{bmatrix}$$

Problema em questào: Dado A e b determinar x.

2.1 Métodos Diretos

Regra de Cramer

$$x_i = \frac{det(A_i)}{det(A)}$$

det: Determinante
A_i: Substituição da coluna i por b em A
Esse método se torna extremamente complexo para n » 2, pois se houver n+1 determinantes de ordem n, teremos: $(n+1) \times n! \times (n-1)$ multiplicações
Com n = 20 um computador que faça 100 milhões de multiplicações por segundo levaria 300.000 anos para computar.

Inversa

$Ax = b \longrightarrow A^{-1}Ax = A^{-1}b \longrightarrow x = A^{-1}b$ caso A^{-1} exista.
Problemas: ineficiente e instável (graças ao erro de arredondamento existente nas inversões, que se acumulam).

Resolução de SL Triangulares

Dado a matrix A triangular superior n x n com elementos diagonais a_{ii} não nulos.

$$\text{A} = \begin{bmatrix} a_11 & a_12 & \cdots & a_{1n} \\ 0 & a_22 & \cdots & a_{2n} \\ 0 & 0 & \ddots & \ddots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix}$$

```
x_n = b_n/a_nn
for k = n − 1, ..., 1 do
  soma = b_k
  for j = k + 1, ..., n do
    soma = soma - a_kj x_j
  end for
  x_k = soma / a_kk
end for
```

Esforço computacional:
n divisões
$\sum_{j=1}^{n_1} j = n(n-1)/2$ adições
$\sum_{j=1}^{n-1} j = n(n-1)/2$ multiplicações
Gerando uma complexidade $O(n^2)$ (a mesma do bubblesort)