MAKRON
Books

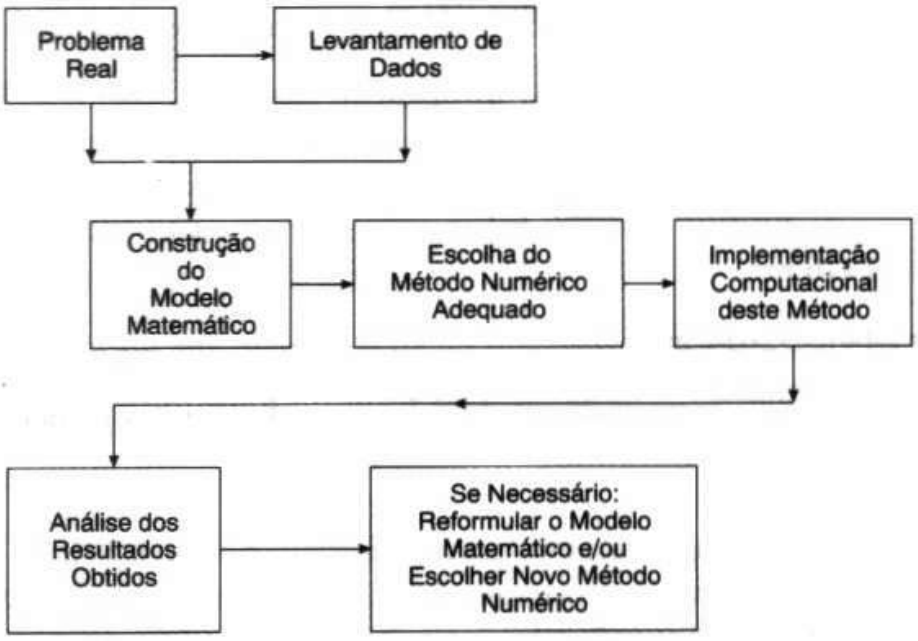
CAPÍTULO 1

NOÇÕES BÁSICAS SOBRE ERROS

1.1 INTRODUÇÃO

Nos capítulos seguintes, estudaremos métodos numéricos para a resolução de problemas que surgem nas mais diversas áreas.

A resolução de tais problemas envolve várias fases que podem ser assim estruturadas:



Não é raro acontecer que os resultados finais estejam distantes do que se esperaria obter, ainda que todas as fases de resolução tenham sido realizadas corretamente.

Os resultados obtidos dependem também:

- da precisão dos dados de entrada;
- da forma como estes dados são representados no computador;
- das operações numéricas efetuadas.

Os dados de entrada contêm uma imprecisão inerente, isto é, não há como evitar que ocorram, uma vez que representam medidas obtidas usando equipamentos específicos, como, por exemplo, no caso de medidas de corrente e tensão num circuito elétrico, ou então podem ser dados resultantes de pesquisas ou levantamentos, como no caso de dados populacionais obtidos num recenseamento.

Neste capítulo, estudaremos os erros que surgem da representação de números num computador e os erros resultantes das operações numéricas efetuadas, [23], [26] e [31].

1.2 REPRESENTAÇÃO DE NÚMEROS

Exemplo 1

Calcular a área de uma circunferência de raio 100 m.

RESULTADOS OBTIDOS

a) A = 31400 m^2

b) A = 31416 m^2

c) A = 31415.92654 m^2

Como justificar as diferenças entre os resultados? É possível obter "exatamente" esta área?

Exemplo 2

Efetuar os somatórios seguintes em uma calculadora e em um computador:

$$S = \sum_{i=1}^{30000} x_i \qquad \text{para } x_i = 0.5 \text{ e para } x_i = 0.11$$

RESULTADOS OBTIDOS

i)	para $x_i = 0.5$:	na calculadora:	S = 15000
		no computador:	S = 15000
ii)	para $x_i = 0.11$:	na calculadora:	S = 3300
		no computador:	S = 3299.99691

Como justificar a diferença entre os resultados obtidos pela calculadora e pelo computador para $x_i = 0.11$?

Os erros ocorridos nos dois problemas dependem da representação dos números na máquina utilizada.

A representação de um número depende da base escolhida ou disponível na máquina em uso e do número máximo de dígitos usados na sua representação.

O número π, por exemplo, não pode ser representado através de um número finito de dígitos decimais. No Exemplo 1, o número π foi escrito como 3.14, 3.1416 e 3.141592654 respectivamente nos casos (*a*), (*b*) e, (*c*). Em cada um deles foi obtido um resultado diferente, e o erro neste caso depende exclusivamente da aproximação escolhida para π. Qualquer que seja a circunferência, a sua área nunca será obtida exatamente, uma vez que π é um número irracional.

Como neste exemplo, qualquer cálculo que envolva números que não podem ser representados através de um número finito de dígitos não fornecerá como resultado um valor exato. Quanto maior o número de dígitos utilizados, maior será a precisão obtida. Por isso, a melhor aproximação para o valor da área da circunferência é aquela obtida no caso (*c*).

Além disto, um número pode ter representação finita em uma base e não-finita em outras bases. A base decimal é a que mais empregamos atualmente. Na Antiguidade, foram utilizadas outras bases, como a base 12, a base 60. Um computador opera normalmente no sistema binário.

Observe o que acontece na interação entre o usuário e o computador: os dados de entrada são enviados ao computador pelo usuário no sistema decimal; toda esta informação é convertida para o sistema binário, e as operações todas serão efetuadas neste sistema.

Os resultados finais serão convertidos para o sistema decimal e, finalmente, serão transmitidos ao usuário. Todo este processo de conversão é uma fonte de erros que afetam o resultado final dos cálculos.

Na próxima seção, estudaremos os processos para conversão de números do sistema binário para o sistema decimal e vice-versa.

1.2.1 CONVERSÃO DE NÚMEROS NOS SISTEMAS DECIMAL E BINÁRIO

Veremos inicialmente a conversão de números inteiros.

Considere os números $(347)_{10}$ e $(10111)_2$. Estes números podem ser assim escritos:

$$(347)_{10} = 3 \times 10^2 + 4 \times 10^1 + 7 \times 10^0$$
$$(10111)_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

De um modo geral, um número na base β, $(a_j a_{j-1} \ldots a_2 a_1 a_0)_\beta$, $0 \leq a_k \leq (\beta - 1)$, $k = 1, \ldots, j$, pode ser escrito na forma polinomial:

$$a_j\beta^j + a_{j-1}\beta^{j-1} + \ldots + a_2\beta^2 + a_1\beta^1 + a_0\beta^0.$$

Com esta representação, podemos facilmente converter um número representado no sistema binário para o sistema decimal.

Por exemplo:

$$(10111)_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$$

Colocando agora o número 2 em evidência teremos:

$$(10111)_2 = 2 \times (1 + 2 \times (1 + 2 \times (0 + 2 \times 1))) + 1 = (23)_{10}$$

Deste exemplo, podemos obter um processo para converter um número representado no sistema binário para o sistema decimal:

A representação do número $(a_j a_{j-1} \ldots a_2 a_1 a_0)_2$ na base 10, denotada por b_0, é obtida através do processo:

$$b_j = a_j$$
$$b_{j-1} = a_{j-1} + 2b_j$$

$$b_{j-2} = a_{j-2} + 2b_{j-1}$$
$$\vdots \qquad \vdots$$
$$b_1 = a_1 + 2b_2$$
$$b_0 = a_0 + 2b_1$$

Para $(10111)_2$, a seqüência obtida será:

$$b_4 = a_4 = 1$$
$$b_3 = a_3 + 2b_4 = 0 + 2 \times 1 = 2$$
$$b_2 = a_2 + 2b_3 = 1 + 2 \times 2 = 5$$
$$b_1 = a_1 + 2b_2 = 1 + 2 \times 5 = 11$$
$$b_0 = a_0 + 2b_1 = 1 + 2 \times 11 = 23$$

Veremos agora um processo para converter um número inteiro representado no sistema decimal para o sistema binário. Considere o número $N_0 = (347)_{10}$ e $(a_j a_{j-1} \ldots a_1 a_0)_2$ a sua representação na base 2.

Temos então que:

$$347 = 2 \times (a_j \times 2^{j-1} + a_{j-1} \times 2^{j-2} + \ldots + a_2 \times 2 + a_1) + a_0 = 2 \times 173 + 1$$

e, portanto, o dígito $a_0 = 1$ representa o resto da divisão de 347 por 2. Repetindo agora este processo para o número $N_1 = 173$:

$$173 = a_j \times 2^{j-1} + a_{j-1} \times 2^{j-2} + \ldots + a_2 \times 2 + a_1$$

obteremos o dígito a_1, que será o resto da divisão de N_1 por 2. Seguindo este raciocínio obtemos a seqüência de números N_j e a_j.

$$N_0 = 347 = 2 \times 173 + 1 \Rightarrow a_0 = 1$$
$$N_1 = 173 = 2 \times 86 + 1 \Rightarrow a_1 = 1$$
$$N_2 = 86 = 2 \times 43 + 0 \Rightarrow a_2 = 0$$
$$N_3 = 43 = 2 \times 21 + 1 \Rightarrow a_3 = 1$$
$$N_4 = 21 = 2 \times 10 + 1 \Rightarrow a_4 = 1$$
$$N_5 = 10 = 2 \times 5 + 0 \Rightarrow a_5 = 0$$
$$N_6 = 5 = 2 \times 2 + 1 \Rightarrow a_6 = 1$$

$$N_7 = 2 = 2 \times 1 + 0 \Rightarrow a_7 = 0$$
$$N_8 = 1 = 2 \times 0 + 1 \Rightarrow a_8 = 1$$

Portanto, a representação de $(347)_{10}$ na base 2 será 101011011.

No caso geral, considere um número inteiro N na base 10 e a sua representação binária denotada por: $(a_j a_{j-1} \ldots a_2 a_1 a_0)_2$. O algoritmo a seguir obtém a cada k o dígito binário a_k.

Passo 0: $k = 0$
$N_k = N$

Passo 1: Obtenha q_k e r_k tais que:
$N_k = 2 \times q_k + r_k$
Faça $a_k = r_k$

Passo 2: Se $q_k = 0$, pare.
Caso contrário, faça $N_{k+1} = q_k$.
Faça $k = k + 1$ e volte para o passo 1.

Consideremos agora a conversão de um número fracionário da base 10 para a base 2.

Sejam, por exemplo:

$r = 0.125$, $s = 0.66666...$, $t = 0.414213562...$.

Dizemos que r tem representação finita e que s e t têm representação infinita.

Dado um número entre 0 e 1 no sistema decimal, como obter sua representação binária?

Considerando o número $r = 0.125$, existem dígitos binários: $d_1, d_2, \ldots, d_j \ldots$, tais que $(0.\ d_1 d_2 \ldots d_j \ldots)_2$ será sua representação na base 2.

Assim,

$$(0.125)_{10} = d_1 \times 2^{-1} + d_2 \times 2^{-2} + \ldots + d_j \times 2^{-j} + \ldots$$

Multiplicando cada termo da expressão acima por 2 teremos:

$$2 \times 0.125 = 0.250 = 0 + 0.25 = d_1 + d_2 \times 2^{-1} + d_3 \times 2^{-2} + \ldots + d_j \times 2^{-j+1} + \ldots$$

e, portanto, d_1 representa a parte inteira de 2×0.125 que é igual a zero e $d_2 \times 2^{-1} + d_3 \times 2^{-2} + \ldots + d_j \times 2^{-j+1} + \ldots$ representa a parte fracionária de 2×0.125 que é 0.250.

Aplicando agora o mesmo procedimento para 0.250, teremos:

$$\begin{aligned} 0.250 &= d_2 \times 2^{-1} + d_3 \times 2^{-2} + \ldots + d_j \times 2^{-j+1} + \ldots \Rightarrow 2 \times 0.250 = 0.5 = \\ &= d_2 + d_3 \times 2^{-1} + d_4 \times 2^{-2} + \ldots + d_j \times 2^{-j+2} + \ldots \Rightarrow d_2 = 0 \end{aligned}$$

e repetindo o processo para 0.5:

$$\begin{aligned} 0.5 &= d_3 \times 2^{-1} + d_4 \times 2^{-2} + \ldots + d_j \times 2^{-j+2} + \ldots \Rightarrow \\ & 2 \times 0.5 = 1.0 = d_3 + d_4 \times 2^{-1} + \ldots + d_j \times 2^{-j+3} + \ldots \Rightarrow d_3 = 1 \end{aligned}$$

Como a parte fracionária de 2×0.5 é zero, o processo de conversão termina, e teremos: $d_1 = 0$, $d_2 = 0$ e $d_3 = 1$ e, portanto, o número $(0.125)_{10}$ tem representação finita na base 2: $(0.001)_2$

Um número real entre 0 e 1 pode ter representação finita no sistema decimal, mas representação *infinita* no sistema binário.

No caso geral, seja r um número entre 0 e 1 no sistema decimal e $(0.\, d_1 d_2 \ldots d_j \ldots)_2$ sua representação no sistema binário.

Os dígitos binários $d_1, d_2, \ldots, d_j, \ldots$ são obtidos através do seguinte algoritmo:

Passo 0: $r_1 = r$; $k = 1$

Passo 1: Calcule $2r_k$.
Se $2r_k \geqslant 1$, faça: $d_k = 1$,
caso contrário, faça: $d_k = 0$

Passo 2: Faça $r_{k+1} = 2r_k - d_k$.
Se $r_{k+1} = 0$, pare.
Caso contrário:

Passo 3: $k = k + 1$.
Volte ao passo 1.

Observar que o algoritmo pode ou não parar após um número finito de passos. Para $r = (0.125)_{10}$ teremos $r_4 = 0$. Já para $r = (0.1)_{10}$, teremos: $r_1 = 0.1$

$$\begin{aligned}
k = 1 \quad 2r_1 = 0.2 \Rightarrow \quad & d_1 = 0 \\
& r_2 = 0.2 \\
k = 2 \quad 2r_2 = 0.4 \Rightarrow \quad & d_2 = 0 \\
& r_3 = 0.4 \\
k = 3 \quad 2r_3 = 0.8 \Rightarrow \quad & d_3 = 0 \\
& r_4 = 0.8 \\
k = 4 \quad 2r_4 = 1.6 \Rightarrow \quad & d_4 = 1 \\
& r_5 = 0.6 \\
k = 5 \quad 2r_5 = 1.2 \Rightarrow \quad & d_5 = 1 \\
& r_6 = 0.2 = r_2
\end{aligned}$$

Como $r_6 = r_2$, teremos que os resultados para k de 2 a 5 se repetirão e então: $r_{10} = r_6 = r_2 = 0.2$ e assim indefinidamente.

Concluímos que:

$$(0.1)_{10} = (0.0001100110011\overline{0011}...)_2$$

e, portanto, o número $(0.1)_{10}$ não tem representação binária finita.

O fato de um número não ter representação finita no sistema binário pode acarretar a ocorrência de erros aparentemente inexplicáveis em cálculos efetuados em sistemas computacionais binários.

Analisando o Exemplo 2 da Seção 1.2 e usando o processo de conversão descrito anteriormente, temos que o número $(0.5)_{10}$ tem representação finita no sistema binário: $(0.1)_2$; já o número $(0.11)_{10}$ terá representação infinita:

$$(0.000111000010100011110\overline{1011100001010001111 01}...)_2$$

Um computador que opera no sistema binário irá armazenar uma aproximação para $(0.11)_{10}$, uma vez que possui uma quantidade fixa de posições para guardar os dígitos da mantissa de um número, e esta aproximação será usada para realizar os cálculos. Não se pode, portanto, esperar um resultado exato.

Considere agora um número entre 0 e 1 representado no sistema binário:

$(r)_2 = (0.\ d_1d_2...d_j...)_2$

Como obter sua representação no sistema decimal?

Um processo para conversão é equivalente ao que descrevemos anteriormente. Definindo $r_1 = r$, a cada iteração k, o processo de conversão multiplica o número r_k por $(10)_{10} = (1010)_2$ e obtém o dígito b_k como sendo a parte inteira deste produto convertida para a base decimal. É importante observar que as operações devem ser efetuadas no sistema binário, [26]. O algoritmo a seguir formaliza este processo.

Passo 0: $r_1 = r$; $k = 1$

Passo 1: Calcule $w_k = (1010)_2 \times r_k$.
Seja z_k a parte inteira de w_k
b_k é a conversão de z_k para a base 10.

Passo 2: Faça $r_{k+1} = w_k - z_k$
Se $r_{k+1} = 0$, pare.

Passo 3: $k = k + 1$.
Volte ao passo 1.

Por exemplo, considere o número:

$(r)_2 = (0.000111)_2 = (0.b_1b_2 \ldots b_j)_{10}$

Seguindo o algoritmo, teremos:

$r_1 = (0.000111)_2$;
$w_1 = (1010)_2 \times r_1 = 1.00011 \Rightarrow b_1 = 1$ e $r_2 = 0.00011$;
$w_2 = (1010)_2 \times r_2 = 0.1111 \Rightarrow b_2 = 0$ e $r_3 = 0.1111$;
$w_3 = (1010)_2 \times r_3 = 1001.011 \Rightarrow b_3 = 9$ e $r_4 = 0.011$;
$w_4 = (1010)_2 \times r_4 = 11.11 \Rightarrow b_4 = 3$ e $r_5 = 0.11$;
$w_5 = (1010)_2 \times r_5 = 111.1 \Rightarrow b_5 = 7$ e $r_6 = 0.1$;
$w_6 = (1010)_2 \times r_6 = 101 \Rightarrow b_6 = 5$ e $r_7 = 0$;

Portanto $(0.000111)_2 = (0.109375)_{10}$

Podemos agora entender melhor por que o resultado da operação

$$S = \sum_{i=1}^{30000} 0.11$$

não é obtido exatamente num computador. Já vimos que $(0.11)_{10}$ não tem representação finita no sistema binário. Supondo um computador que trabalhe com apenas 6 dígitos na mantissa, o número $(0.11)_{10}$ seria armazenado como $(0.000111)_2$ e este número representa exatamente $(0.109375)_{10}$. Portanto, todas as operações que envolvem o número $(0.11)_{10}$ seriam realizadas com esta aproximação. Veremos na próxima seção a representação de números em aritmética de ponto flutuante com o objetivo de se entender melhor a causa de resultados imprecisos em operações numéricas.

1.2.2 ARITMÉTICA DE PONTO FLUTUANTE

Um computador ou calculadora representa um número real no sistema denominado *aritmética de ponto flutuante*. Neste sistema, o número r será representado na forma:

$$\pm (. d_1 d_2 ... d_t) \times \beta^e$$

onde:

β é a base em que a máquina opera;
t é o número de dígitos na mantissa; $0 \leq d_j \leq (\beta - 1)$, $j = 1, ..., t$, $d_1 \neq 0$;
e é o expoente no intervalo [l, u].

Em qualquer máquina, apenas um subconjunto dos números reais é representado exatamente, e, portanto, a representação de um número real será realizada através de truncamento ou de arredondamento.

Considere, por exemplo, uma máquina que opera no sistema:

$\beta = 10$; $t = 3$; $e \in [-5, 5]$.

Os números serão representados na seguinte forma nesse sistema:

$0. d_1 d_2 d_3 \times 10^e$, $0 \leq d_j \leq 9$, $d_1 \neq 0$, $e \in [-5, 5]$

O menor número, em valor absoluto, representado nesta máquina é:

$$m = 0.100 \times 10^{-5} = 10^{-6}$$

e o maior número, em valor absoluto, é:

$$M = 0.999 \times 10^{5} = 99900$$

Considere o conjunto dos números reais $\mathbb{R}$ e o seguinte conjunto:

$$G = \{x \in \mathbb{R} \mid m \leqslant |x| \leqslant M\}$$

Dado um número real x, várias situações poderão ocorrer:

Caso 1) $x \in G$:

por exemplo: $x = 235.89 = 0.23589 \times 10^{3}$. Observe que este número possui 5 dígitos na mantissa. Estão representados exatamente nesta máquina os números: 0.235×10^{3} e 0.236×10^{3}. Se for usado o truncamento, x será representado por 0.235×10^{3} e, se for usado o arredondamento, x será representado por 0.236×10^{3} (o truncamento e o arredondamento serão estudados na Seção 1.3.2.);

Caso 2) $|x| < m$:

por exemplo: $x = 0.345 \times 10^{-7}$. Este número não pode ser representado nesta máquina porque o expoente *e* é menor que –5. Esta é uma situação em que a máquina acusa a ocorrência de *underflow*;

Caso 3) $|x| > M$:

por exemplo: $x = 0.875 \times 10^{9}$. Neste caso, o expoente *e* é maior que 5 e a máquina acusa a ocorrência de *overflow*.

Algumas linguagens de programação permitem que as variáveis sejam declaradas em *precisão dupla*. Neste caso, esta variável será representada no sistema de aritmética de ponto flutuante da máquina, mas com aproximadamente o dobro de dígitos disponíveis na mantissa. É importante observar que, neste caso, o tempo de execução e requerimentos de memória aumentam de forma significativa.

O *zero* em ponto flutuante é, em geral, representado com o menor expoente possível na máquina. Isto porque a representação do zero por uma mantissa nula e um expoente qualquer para a base β pode acarretar perda de dígitos significativos no resultado da adição deste *zero* a um outro número. Por exemplo, em uma máquina que opera na base 10 com 4 dígitos na mantissa, para $x = 0.0000 \times 10^4$ e $y = 0.3134 \times 10^{-2}$ o resultado de $x + y$ seria 0.3100×10^{-2}, isto é, são perdidos dois dígitos do valor exato y. Este resultado se deve à forma como é efetuada a adição em ponto flutuante, que estudaremos na Seção 1.3.3.

Exemplo 3

Dar a representação dos números a seguir num sistema de aritmética de ponto flutuante de três dígitos para $\beta = 10$, $m = -4$ e $M = 4$.

x	Representação obtida por arredondamento	Representação obtida por truncamento
1.25	0.125×10	0.125×10
10.053	0.101×10^2	0.100×10^2
–238.15	-0.238×10^3	-0.238×10^3
2.71828...	0.272×10	0.271×10
0.000007	(expoente menor que –4)	=
718235.82	(expoente maior que 4)	=

1.3 ERROS

1.3.1 ERROS ABSOLUTOS E RELATIVOS

No Exemplo 1 da Seção 1.2, diferentes resultados para a área da circunferência foram obtidos, dependendo da aproximação adotada para o valor de π.

Definimos como *erro absoluto* a diferença entre o valor exato de um número x e de seu valor aproximado $\bar{x}$: $EA_x = x - \bar{x}$.

Em geral, apenas o valor $\bar{x}$ é conhecido, e, neste caso, é impossível obter o valor exato do erro absoluto. O que se faz é obter um limitante superior ou uma estimativa para o módulo do erro absoluto.

Por exemplo, sabendo-se que $\pi \in (3.14, 3.15)$ tomaremos para π um valor dentro deste intervalo e teremos, então, $|EA_{\pi}| = |\pi - \bar{\pi}| < 0.01$.

Seja agora o número x representado por $\bar{x} = 2112.9$ de tal forma que $|EA_x| < 0.1$, ou seja, $x \in (2112.8, 2113)$ e seja o número y representado por $\bar{y} = 5.3$ de tal forma que $|EA_y| < 0.1$, ou seja, $y \in (5.2, 5.4)$. Os limitantes superiores para os erros absolutos são os mesmos. Podemos dizer que ambos os números estão representados com a mesma precisão?

É preciso comparar a ordem de grandeza de x e y. Feito isto, é fácil concluir que o primeiro resultado é mais preciso que o segundo, pois a ordem de grandeza de x é maior que a ordem de grandeza de y. Então, dependendo da ordem de grandeza dos números envolvidos, o erro absoluto não é suficiente para descrever a precisão de um cálculo. Por esta razão, o erro relativo é amplamente empregado.

O *erro relativo* é definido como o erro absoluto dividido pelo valor aproximado:

$$ER_x = \frac{EA_x}{\bar{x}} = \frac{x - \bar{x}}{\bar{x}}$$

No exemplo anterior, temos

$$|ER_x| = \frac{|EA_x|}{|\bar{x}|} < \frac{0.1}{2112.9} \approx 4.7 \times 10^{-5}$$

e

$$|ER_y| = \frac{|EA_y|}{|\bar{y}|} < \frac{0.1}{5.3} \approx 0.02,$$

confirmando, portanto, que o número x é representado com maior precisão que o número y.

1.3.2 ERROS DE ARREDONDAMENTO E TRUNCAMENTO EM UM SISTEMA DE ARITMÉTICA DE PONTO FLUTUANTE

Vimos na Seção 1.2 que a representação de um número depende fundamentalmente da máquina utilizada, pois seu sistema estabelecerá a base numérica adotada, o total de dígitos na mantissa etc.

Seja um sistema que opera em aritmética de ponto flutuante de t dígitos na base 10, e seja x, escrito na forma

$$x = f_x \times 10^e + g_x \times 10^{e-t} \text{ onde } 0.1 \leq f_x < 1 \text{ e } 0 \leq g_x < 1.$$

Por exemplo, se t = 4 e x = 234.57, então

$$x = 0.2345 \times 10^3 + 0.7 \times 10^{-1}, \text{ donde } f_x = 0.2345 \text{ e } g_x = 0.7.$$

É claro que na representação de x neste sistema $g_x \times 10^{e-t}$ não pode ser incorporado totalmente à mantissa. Então, surge a questão de como considerar esta parcela na mantissa e definir o erro absoluto (ou relativo) máximo cometido.

Vimos na Seção 1.2.2 que podem ser adotados dois critérios: o do arredondamento e o do truncamento (ou cancelamento).

No *truncamento*, $g_x \times 10^{e-t}$ é desprezado e $\bar{x} = f_x \times 10^e$. Neste caso, temos

$$|EA_x| = |x - \bar{x}| = |g_x| \times 10^{e-t} < 10^{e-t}, \text{ visto que } |g_x| < 1$$

$$|ER_x| = \frac{|EA_x|}{|\bar{x}|} = \frac{|g_x| \times 10^{e-t}}{|f_x| \times 10^e} < \frac{10^{e-t}}{0.1 \times 10^e} = 10^{-t+1}$$

visto que 0.1 é o menor valor possível para f_x.

No *arredondamento*, f_x é modificado para levar em consideração g_x. A forma de arredondamento mais utilizada é o arredondamento simétrico:

$$\bar{x} = \begin{cases} f_x \times 10^e & \text{se } |g_x| < \frac{1}{2} \\ f_x \times 10^e + 10^{e-t} & \text{se } |g_x| \geq \frac{1}{2} \end{cases}$$

Portanto se $|g_x| < \frac{1}{2}$, g_x é desprezado, caso contrário, somamos o número 1 ao último dígito de f_x.

Então, se $|g_x| < \frac{1}{2}$ teremos

$$|EA_x| = |x - \bar{x}| = |g_x| \times 10^{e-t} < \frac{1}{2} \times 10^{e-t}$$

e

$$|ER_x| = \frac{|EA_x|}{|\bar{x}|} = \frac{|g_x| \times 10^{e-t}}{|f_x| \times 10^e} < \frac{0.5 \times 10^{e-t}}{0.1 \times 10^e} = \frac{1}{2} \times 10^{-t+1}$$

E se $|g_x| \geqslant 1/2$, teremos

$$\begin{aligned} |EA_x| &= |x - \bar{x}| = |(f_x \times 10^e + g_x \times 10^{e-t}) - (f_x \times 10^e + 10^{e-t})| \\ &= |g_x \times 10^{e-t} - 10^{e-t}| = |(g_x - 1)| \times 10^{e-t} \leqslant \frac{1}{2} \times 10^{e-t} \end{aligned}$$

e

$$|ER_x| = \frac{|EA_x|}{|\bar{x}|} \leqslant \frac{1/2 \times 10^{e-t}}{|f_x \times 10^e + 10^{e-t}|} < \frac{1/2 \times 10^{e-t}}{|f_x| \times 10^e} <$$

$$< \frac{1/2 \times 10^{e-t}}{0.1 \times 10^e} = \frac{1}{2} \times 10^{-t+1}$$

Portanto, em qualquer caso teremos

$$|EA_x| \leqslant \frac{1}{2} \times 10^{e-t} \quad \text{e} \quad |ER_x| < \frac{1}{2} \times 10^{-t+1}$$

Apesar de incorrer em erros menores, o uso do arredondamento acarreta um tempo maior de execução e por esta razão o truncamento é mais utilizado.

1.3.3 ANÁLISE DE ERROS NAS OPERAÇÕES ARITMÉTICAS DE PONTO FLUTUANTE

Dada uma seqüência de operações, como, por exemplo, $u = [(x + y) - z - t] + w$, é importante a noção de como o erro em uma operação propaga-se ao longo das operações subseqüentes.

O erro total em uma operação é composto pelo erro das parcelas ou fatores e pelo erro no resultado da operação.

Nos exemplos a seguir, vamos supor que as operações são efetuadas num sistema de aritmética de ponto flutuante de quatro dígitos, na base 10, e com acumulador de precisão dupla.

Exemplo 4

Dados $x = 0.937 \times 10^4$ e $y = 0.1272 \times 10^2$, obter $x + y$.

A adição em aritmética de ponto flutuante requer o alinhamento dos pontos decimais dos dois números. Para isto, a mantissa do número de menor expoente deve ser deslocada para a direita. Este deslocamento deve ser de um número de casas decimais igual à diferença entre os dois expoentes.

Alinhando os pontos decimais dos valores acima, temos

$$x = 0.937 \times 10^4 \text{ e } y = 0.001272 \times 10^4.$$

Então,

$$x + y = (0.937 + 0.001272) \times 10^4 = 0.938272 \times 10^4.$$

Este é o resultado exato desta operação. Dado que em nosso sistema t é igual a 4, este resultado dever ser arredondado ou truncado.

Então, $\overline{x + y} = 0.9383 \times 10^4$ no arredondamento e $\overline{x + y} = 0.9382 \times 10^4$ no truncamento.

Exemplo 5

Sejam x e y do Exemplo 4. Obter xy:

$$\begin{aligned} xy &= (0.937 \times 10^4) \times (0.1272 \times 10^2) \\ &= (0.937 \times 0.1272) \times 10^6 = \\ &= 0.1191864 \times 10^6. \end{aligned}$$

Então, $\overline{xy} = 0.1192 \times 10^6$, se for efetuado o arredondamento, e $\overline{xy} = 0.1191 \times 10^6$, se for efetuado o truncamento.

Os Exemplos 4 e 5 mostram que ainda que as parcelas ou fatores de uma operação estejam representados exatamente no sistema, não se pode esperar que o resultado armazenado seja exato.

Na maioria dos sistemas, o resultado exato da operação que denotaremos por OP é normalizado e em seguida arredondado ou truncado para t dígitos, obtendo assim o resultado aproximado $\overline{OP}$ que é armazenado na memória da máquina.

Então, conforme vimos na Seção 1.3.2, o erro relativo no resultado de uma operação (supondo que as parcelas ou fatores estão representados exatamente) será:

$|ER_{OP}| < 10^{-t+1}$ no truncamento

$|ER_{OP}| < \frac{1}{2} \times 10^{-t+1}$ no arredondamento.

Veremos a seguir as fórmulas para os erros absoluto e relativo nas operações aritméticas com erros nas parcelas ou fatores.

Vamos supor que o erro final é arredondado.

Sejam x e y, tais que $x = \bar{x} + EA_x$ e $y = \bar{y} + EA_y$.

Adição: $x + y$

$$x + y = (\bar{x} + EA_x) + (\bar{y} + EA_y) = (\bar{x} + \bar{y}) + (EA_x + EA_y).$$

Então, o erro absoluto na soma, denotado por EA_{x+y} é a soma dos erros absolutos das parcelas:

$$EA_{x+y} = EA_x + EA_y.$$

O erro relativo será:

$$ER_{x+y} = \frac{EA_{x+y}}{\bar{x} + \bar{y}} = \frac{EA_x}{\bar{x}} \left(\frac{\bar{x}}{\bar{x} + \bar{y}} \right) + \frac{EA_y}{\bar{y}} \left(\frac{\bar{y}}{\bar{x} + \bar{y}} \right) =$$

$$= ER_x \left(\frac{\bar{x}}{\bar{x} + \bar{y}} \right) + ER_y \left(\frac{\bar{y}}{\bar{x} + \bar{y}} \right).$$

Subtração: $x - y$

Analogamente temos:

$$EA_{x-y} = EA_x - EA_y \text{ e}$$

$$ER_{x-y} = \frac{EA_x - EA_y}{\bar{x} - \bar{y}} = ER_x \left(\frac{\bar{x}}{\bar{x} - \bar{y}} \right) - ER_y \left(\frac{\bar{y}}{\bar{x} - \bar{y}} \right).$$

Multiplicação: xy

$$xy = (\bar{x} + EA_x)(\bar{y} + EA_y) =$$
$$= \bar{x}\,\bar{y} + \bar{x}\,EA_y + \bar{y}\,EA_x + (EA_x)(EA_y)$$

Considerando que $(EA_x)(EA_y)$ é um número pequeno, podemos desprezar este termo na expressão acima.

Teremos então $EA_{xy} \approx \bar{x}\,EA_y + \bar{y}\,EA_x$

$$ER_{xy} \approx \frac{\bar{x}\,EA_y + \bar{y}\,EA_x}{\bar{x}\,\bar{y}} = \frac{EA_x}{\bar{x}} + \frac{EA_y}{\bar{y}} = ER_x + ER_y.$$

Divisão: x/y

$$\frac{x}{y} = \frac{\bar{x} + EA_x}{\bar{y} + EA_y} = \frac{\bar{x} + EA_x}{\bar{y}} \left(\frac{1}{1 + \frac{EA_y}{\bar{y}}} \right).$$

Representando o fator $\dfrac{1}{1 + \dfrac{EA_y}{\bar{y}}}$ sob a forma de uma série infinita, teremos

$$\frac{1}{1 + \dfrac{EA_y}{\bar{y}}} = 1 - \frac{EA_y}{\bar{y}} + \left(\frac{EA_y}{\bar{y}}\right)^2 - \left(\frac{EA_y}{\bar{y}}\right)^3 + \ldots$$

e desprezando os termos com potências maiores que 1, teremos

$$\frac{x}{y} \approx \frac{\bar{x} + EA_x}{\bar{y}} \left(1 - \frac{EA_y}{\bar{y}}\right) = \frac{\bar{x}}{\bar{y}} + \frac{EA_x}{\bar{y}} - \frac{\bar{x}\, EA_y}{\bar{y}^2} - \frac{EA_x\, EA_y}{\bar{y}^2}.$$

Então

$$\frac{x}{y} \approx \frac{\bar{x}}{\bar{y}} + \frac{EA_x}{\bar{y}} - \frac{\bar{x}\, EA_y}{\bar{y}^2}.$$

$$\text{Assim, } EA_{x/y} \approx \frac{EA_x}{\bar{y}} - \frac{\bar{x}\, EA_y}{\bar{y}^2} = \frac{\bar{y}\, EA_x - \bar{x}\, EA_y}{\bar{y}^2}$$

e

$$ER_{x/y} \approx \left(\frac{\bar{y}\, EA_x - \bar{x}\, EA_y}{\bar{y}^2}\right) \frac{\bar{y}}{\bar{x}} = \frac{EA_x}{\bar{x}} - \frac{EA_y}{\bar{y}} = ER_x - ER_y$$

Escrevemos todas essas fórmulas sem considerar o erro de arredondamento ou truncamento no resultado final.

A análise completa da propagação de erros se faz considerando os erros nas parcelas ou fatores e no resultado de cada operação efetuada.

Exemplo 6

Supondo que x, y, z e t estejam representados exatamente, qual o erro total no cálculo de u = (x + y)z – t? (Calcularemos o erro relativo e denotaremos por RA o erro relativo de arredondamento no resultado da operação.)

Seja s = x + y. O erro relativo nesta operação será:

$$ER_s = ER_x \left(\frac{\bar{x}}{\bar{x} + \bar{y}}\right) + ER_y \left(\frac{\bar{y}}{\bar{x} + \bar{y}}\right) + RA = 0 + RA .$$

Assim, $|ER_s| = |RA| < \frac{1}{2} \times 10^{-t+1}$.

Calculando agora s × z, teremos m = s × z, e o erro relativo desta operação será:

$$ER_m = ER_s + ER_z + RA = ER_s + \frac{EA_z}{\bar{z}} + RA = RA_s + 0 + RA.$$

Então,

$$|ER_m| \leq |RA_s| + |RA| < \frac{1}{2} \times 10^{-t+1} + \frac{1}{2} \times 10^{-t+1} = 10^{-t+1}.$$

Calcularemos u = m – t e o erro relativo desta operação será:

$$ER_u = \frac{EA_m - EA_t}{\bar{m} - \bar{t}} + RA = \frac{EA_m}{\bar{m} - \bar{t}} + RA = \frac{EA_m}{\bar{m}} \left(\frac{\bar{m}}{\bar{m} - \bar{t}}\right) + RA$$

$$= ER_m \left(\frac{\bar{m}}{\bar{m} - \bar{t}}\right) + RA.$$

Então,

$$|ER_u| \leq |ER_m| \left|\frac{\bar{m}}{\bar{m} - \bar{t}}\right| + RA < 10^{-t+1} \left|\frac{\bar{m}}{\bar{m} - \bar{t}}\right| + \frac{1}{2} \times 10^{-t+1}$$

Finalmente,

$$|ER_u| < \left(\frac{|\bar{m}|}{|\bar{m} - \bar{t}|} + \frac{1}{2}\right) \times 10^{-t+1} .$$

Exemplo 7

Vimos que dados x, y e z = x – y, então:

$$ER_z = \frac{EA_x - EA_y}{\bar{x} - \bar{y}} = \frac{EA_x}{\bar{x}}\left(\frac{\bar{x}}{\bar{x} - \bar{y}}\right) - \frac{EA_y}{\bar{y}}\left(\frac{\bar{y}}{\bar{x} - \bar{y}}\right).$$

Se x e y são números positivos arredondados, então:

$$\left|\frac{EA_x}{\bar{x}}\right| < \frac{1}{2} \times 10^{-t+1} \quad \text{e} \quad \left|\frac{EA_y}{\bar{y}}\right| < \frac{1}{2} \times 10^{-t+1}.$$

É claro que se $x \approx y$, então o erro relativo em z pode ser grande, como por exemplo, se $\bar{x} = 0.2357 \times 10^3$ e $\bar{y} = 0.2353 \times 10^3$, então $\bar{z} = \bar{x} - \bar{y} = 0.0004 \times 10^3$ e isto é denominado *cancelamento subtrativo.*

O erro relativo em z é limitado superiormente por:

$$|ER_z| < \left(\frac{0.2357 \times 10^3 + 0.2353 \times 10^3}{0.0004 \times 10^3}\right) \times \frac{1}{2} \times 10^{-3} \ (t = 4) \Rightarrow$$

$$\Rightarrow |ER_z| < 0.5888 \approx 59\%.$$

Este erro relativo é propagado nas próximas operações (pois cada expressão para o erro relativo depende do erro relativo em cada parcela ou fator).

Por exemplo, para w = zt, se $\bar{t} = 0.4537 \times 10^3$ teremos $\bar{w} = 0.1815 \times 10^3$ e

$$|ER_w| \leq |ER_z| + |ER_t| < 0.59 + \frac{1}{2} \times 10^{-3} = 0.5905 \approx 59\%.$$

Desta forma, o cancelamento subtrativo pode dar origem a grandes erros nas operações seguintes.