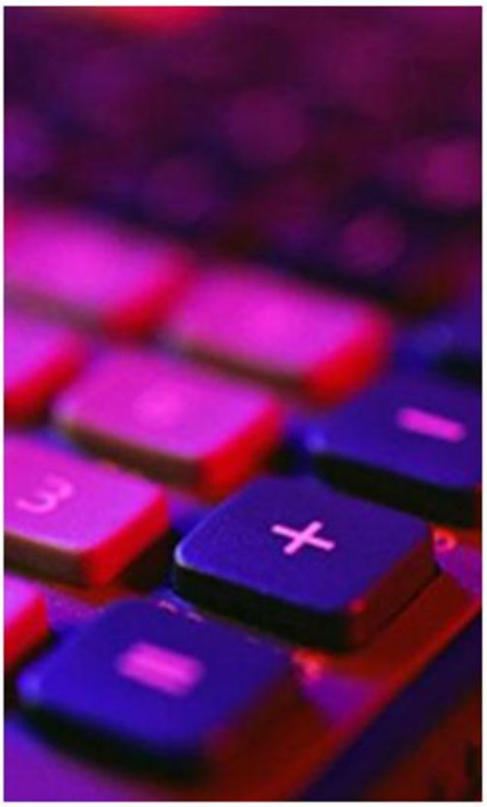

Capítulo 1

Sistema de Numeração e Erros

Objetivos do capítulo

- Entender como se compõe um sistema de numeração e como fazer mudanças de base
- Conhecer o sistema de normatização de precisão simples, dupla e a idéia generalização
- Entender os erros de arredondamento cometidos no processo de armazenamento de dados

O número é um dos elementos principais no estudo da matemática. Apesar de usarmos o número com muita freqüência parecemos não conhecê-lo muito bem. Responder a pergunta, “o que é número?” e entender em detalhes a formação deles não parece uma preocupação da grande maioria de quem utiliza esse ente.

Um número é algo bem intuitivo, não é uma coisa física, palpável. Quando escrevemos 257, o número não é o que está escrito e sim a idéia formada pela visualização dos símbolos, isto significa que, mesmo que utilizássemos outros símbolos para representar 257, o número não mudaria. O número é uma idéia de grandeza e não a simbologia que o representa. Considere o *algarismo romano* X, ele tem o mesmo significado de 10 no sistema de numeração usual, ambos é a representação do mesmo número “dez”, só que expresso com simbologias diferentes.

Estamos acostumados a usar sempre símbolos, denominados algarismos, do conjunto S = {0,1,2,3,4,5,6,7,8,9}, para representar qualquer número. Outros símbolos também são usados para representações de números como fracionários, irracionais e outros.

EXEMPLO 1

4,325; 0,00001 e $\frac{2}{3}$ são exemplos de representações de números racionais.

π, ln2 e $\sqrt[5]{3}$ são exemplos de números irracionais.

Nosso objetivo é estudar os números na forma em que serão tratados pelos métodos numéricos, ou seja, números racionais e com representação finita, com uma vírgula separando a parte inteira da parte fracionária.

Para representar um número no sistema de numeração padrão, usamos um ou mais algarismos do conjunto S, descrito anteriormente. Este sistema é denominado *sistema de numeração decimal* ou *sistema de base 10*, por conter exatamente dez algarismos. Se excluíssemos o algarismo 9, do conjunto S, teríamos o conjunto {0,1,2,...,8} que contêm exatamente 9 algarismos, e mesmo assim seria possível representar qualquer número com apenas algarismos desse conjunto, e teríamos um sistema de numeração denominado *sistema de numeração de base 9*. Da mesma forma, podemos construir os

sistemas de numeração de base 8, 7, ..., 2. Parece estranho, mas realmente podemos representar qualquer número utilizando apenas os algarismos 0 (zero) e 1 (um). Além desses sistemas de numeração é possível criar sistema com base maior que 10. Neste caso é necessário definir símbolos para representar os algarismos maiores que nove.

A seguir temos uma tabela mostrando a representação dos números de 0 a 20, escritos em base de 2 a 10.

	Base de numeração								
Número	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**
Zero	0	0	0	0	0	0	0	0	0
Um	1	1	1	1	1	1	1	1	1
Dois	2	2	2	2	2	2	2	2	10
Três	3	3	3	3	3	3	3	10	11
Quatro	4	4	4	4	4	4	10	11	100
Cinco	5	5	5	5	5	10	11	12	101
Seis	6	6	6	6	10	11	12	20	110
Sete	7	7	7	10	11	12	13	21	111
Oito	8	8	10	11	12	13	20	22	1000
Nove	9	10	11	12	13	14	21	100	1001
Dez	10	11	12	13	14	20	22	101	1010
Onze	11	12	13	14	15	21	23	102	1011
Doze	12	13	14	15	20	22	30	110	1100
Treze	13	14	15	16	21	23	31	111	1101
Quatorze	14	15	16	20	22	24	32	112	1110
Quinze	15	16	17	21	23	30	33	120	1111
Dezesseis	16	17	20	22	24	31	100	121	10000
Dezessete	17	18	21	23	25	32	101	122	10001
Dezoito	18	20	22	24	30	33	102	200	10010
Dezenove	19	21	23	25	31	34	103	201	10011
Vinte	20	22	24	26	32	40	110	202	10100

Tabela 1.1. Representação dos números de zero a vinte, nas bases de 2 a 10.

A menor base de numeração é 2, conhecida como base binária e utiliza, como vimos anteriormente, apenas os algarismos 0 e 1. Porém, não temos um limite para a maior base, isso significa que podemos ter bases 11, 12, 13, e assim por diante. O problema de se ter uma base relativamente grande está na quantidade de símbolos necessários para representar os algarismos, por exemplo, na base 12 os números *dez* e *onze* passam a ser algarismos e precisamos representar cada um deles com apenas um símbolo, é inviável usar "10" e "11", respectivamente, para representá-los, pois isso geraria uma confusão, não saberíamos se estamos tratando de um ou dois algarismos.

É comum para bases maiores que dez, utilizarmos as letras maiúsculas do nosso alfabeto: A, B, C, etc. para representar, respectivamente, 10, 11, 12, etc. A Tabela 1.2 mostra a representação dos números de *zero* a *vinte* nas bases de 10 a 16, onde A, B, C, D, E e F representa, nessa ordem, 10, 11, 12, 13, 14 e 15.

	Base de numeracão						
Número	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Zero	0	0	0	0	0	0	0
Um	1	1	1	1	1	1	1
Dois	2	2	2	2	2	2	2
Três	3	3	3	3	3	3	3
Quatro	4	4	4	4	4	4	4
Cinco	5	5	5	5	5	5	5
Seis	6	6	6	6	6	6	6
Sete	7	7	7	7	7	7	7
Oito	8	8	8	8	8	8	8
Nove	9	9	9	9	9	9	9
Dez	10	A	A	A	A	A	A
Onze	11	10	B	B	B	B	B
Doze	12	11	10	C	C	C	C
Treze	13	12	11	10	D	D	D
Quatorze	14	13	12	11	10	E	E
Quinze	15	14	13	12	11	10	F
Dezesseis	16	15	14	13	12	11	10
Dezessete	17	16	15	14	13	12	11
Dezoito	18	17	16	15	14	13	12
Dezenove	19	18	17	16	15	14	13
Vinte	20	19	18	17	16	15	14

Tabela 1.2. Representação dos números de zero a vinte, nas bases de 10 a 16.

A base 16, também conhecida como hexadecimal, é bastante utilizada no armazenamento de informações em máquinas que processam os dados em base 2, pois o processamento em base 2 é rápido, mas requer mais memória no armazenamento do que em base 16, por exemplo, considere o número *vinte* cuja representação em base 2 é *10100* (veja Tabela 1.1), e em base 16 é *14* (veja Tabela 1.2), assim, serão necessários cinco espaços na memória para armazenar o *vinte* em base 2 e apenas dois, em base de 16. Por outro lado, o processamento é mais vantajoso em base 2, pois existem apenas duas possibilidade distintas para cada um dos cinco espaços de memória ocupado pelo número dois em binário, totalizando 32 possibilidades. Já em base 16,

temos dezesseis possibilidades para cada um dos dois espaços, constituindo $16^2 = 256$ possibilidades. Assim, uma tomada de decisão é mais rápida em base binária e o armazenamento é mais vantajoso em base hexadecimal. A base hexadecimal não possui vantagem significativa em relação a base decimal, porém a conversão de binário para hexadecimal (e vice-versa) é mais rápida do que de binário para base decimal, pois as duas (binária e hexadecimal) são potências de 2. Portanto em se tratando de processamento é vantajoso processar os dados em base 2 e armazená-los em base hexadecimal.

Definição 1.1

Seja $\beta \geq 2$ um número natural, se $d_0, d_1, ...,d_n, d_{-1}, d_{-2}, ...,d_{-m} \in \{0, 1, ... , \beta - 1\}$, então $(d_n d_{n-1}...d_0,d_{-1}d_{-2}...d_{-m})_\beta$ é a representação, em base β, do número decimal

$$d_n \times \beta^n + d_{n-1} \times \beta^{n-1} + ... + d_0 \times \beta^0 + d_{-1} \times \beta^{-1} + d_{-2} \times \beta^{-2} + ... + d_{-m} \times \beta^{-m} .$$

onde:

$(d_n d_{n-1}...d_0)_\beta = d_n \times \beta^n + d_{n-1} \times \beta^{n-1} + ... + d_0 \times \beta^0$ é a parte inteira

$(0,d_{-1}d_{-2}...d_{-m})_\beta = d_{-1} \times \beta^{-1} + d_{-2} \times \beta^{-2} + ... + d_{-m} \times \beta^{-m}$ é a parte fracionária

EXEMPLO 2

$(3412,05)_7$ é a representação de um número em base $\beta = 7$. De acordo com a Definição 1.1:

$$d_0 = 2, \ d_1 = 1, d_2 = 4, d_3 = 3, d_{-1} = 0, d_{-2} = 5$$

E assim,

$$(3412,05)_7 = 3 \times 7^3 + 4 \times 7^2 + 1 \times 7^1 \ 2 \times 7^0 + 2 \times 7^0 + 0 \times 7^{-1} + 5 \times 7^{-2}$$
$$\cong 1234,102041.$$

Ou seja, o número que tem representação em base 7 dada por $(3412,05)_7$ corresponde, aproximadamente, ao número decimal 1234,102041.

Por questões de simplificação de pronúncia e escrita, iremos tratar o número por sua representação, por exemplo, ao invés de dizermos "*o número cuja sua representação em base 7 é* $(3412,05)_7$" iremos dizer apenas "*o número 3412,05 na base 7*". Para melhorar a notação, quando possível omitiremos os parênteses, deixando apenas o índice no final do número, e quando se tratar da *base 10* o índice também poderá ser omitido, por exemplo, $(3412,05)_7$ será denotado simplesmente por $3412,05_7$ e $(1234,102041)_{10}$ por 1234,102041.

Proposição 1.1

Se multiplicarmos o número $(d_n d_{n-1}...d_1 d_0,d_{-1}d_{-2}...d_{-m})_\beta$ por β obtemos o número $(d_n d_{n-1} ... d_1 d_0 d_{-1},d_{-2}...d_{-m})_\beta$, isto é, *a vírgula se desloca um algarismo à direita*. E ainda, se dividirmos o mesmo número por β (multiplicarmos por β^{-1}), obtemos $(d_n d_{n-1}...d_{1,} d_0 d_{-1} d_{-2} ... d_{-m})_\beta$, isto é, *a vírgula se desloca um algarismo para a esquerda*.

Demonstração

De acordo com a Definição 1.1, podemos escrever,

$(d_n d_{n-1}...d_1 d_0,d_{-1}d_{-2}...d_{-m})_\beta = d_n \times \beta^n + d_{n-1} \times \beta^{n-1} + ... + d_0 \times \beta^0 + d_{-1} \times \beta^{-1} + d_{-2} \times \beta^{-2} + ... + d_{-m} \times \beta^{-m}$

Multiplicando, ambos os membros por β temos:

$d_n \times \beta^{n+1} + d_{n-1} \times \beta^n + ... + d_0 \times \beta^1 + d_{-1} \times \beta^0 + d_{-2} \times \beta^{-1} + ... + d_{-m} \times \beta^{-m+1} = (d_n d_{n-1} ... d_0 d_{-1},d_{-2}...d_{-m})_\beta$, Isso demonstra a primeira parte.

Analogamente, dividindo

$$d_n \times \beta^n + d_{n-1} \times \beta^{n-1} + ... + d_1 \times \beta^1 + d_0 \times \beta^0 + d_{-1} \times \beta^{-1} + d_{-2} \times \beta^{-2} + ... + d_{-m} \times \beta^{-m}$$

por β, obtemos:

$$d_n \times \beta^{n-1} + d_{n-1} \times \beta^{n-2} + ... + d_1 \times \beta^0 + d_0 \times \beta^{-1} + d_{-1} \times \beta^{-2} + d_{-2} \times \beta^{-3} + ... + d_{-m} \times \beta^{m-1}$$

que é a representação do número $(d_n d_{n-1} ... d_1, d_0 d_{-1} d_{-2} ... d_{-m})_\beta$ em base β

■

1.1 Mudança de Base

Mudar um número de uma base α para uma base β significa encontrar sua representação no sistema β, a partir da sua representação em α. Para isso estudaremos os seguintes casos:

Mudança de uma base β para a base 10

Essa mudança é feita com a aplicação imediata da Definição 1.1

EXEMPLO 4

Considere o número $31{,}402_5$. A mudança deste número, que se encontra na base $\beta = 5$, para a base 10 é, segundo a Definição 1.1, dada por:

$$31{,}402_5 = 3 \times 5^1 + 1 \times 5^0 + 4 \times 5^{-1} + 0 \times 5^{-2} + 2 \times 5^{-3} \cong 16{,}816$$

Mudança da base 10 para uma base β

Na busca de um método para fazer a conversão de um número decimal para uma base β, vamos propor o seguinte problema: *converter o número decimal 6,625 para a base $\beta = 2$ (base binária)*.

Separamos a parte inteira da fracionária 6,625 = 6 + 0,625. E Dividimos o problema em duas partes:

i) vamos encontrar a representação binária da parte inteira

Suponha que a representação binária de 6 (parte inteira), seja dada por

$$d_n d_{n-1} \ldots d_2 d_1 d_0$$

Pela Definição 1.1 temos:

$$6 = d_n \times 2^n + d_{n-1} \times 2^{n-1} + \ldots + d_2 \times 2^2 + d_1 \times 2^1 + d_0$$

Colocando 2 em evidência no segundo membro e fazendo a divisão inteira de 6 por 2, isto é, $6 = 2 \times 3 + 0$. Temos:

$$2 \times 3 + 0 = 2 \times (d_n \times 2^{n-1} + d_{n-1} \times 2^{n-2} + \ldots + d_2 \times 2^1 + d_1) + d_0,$$

Comparando ambos os membros (com observância ao algoritmo da divisão [x]) obtemos:

$$d_0 = 0 \quad \text{e} \quad 3 = d_n \times 2^{n-1} + d_{n-1} \times 2^{n-2} + \ldots + d_2 \times 2^1 + d_1.$$

Repetindo o processo, isto é, colocando 2 em evidência no segundo membro e escrevendo $3 = 2 \times 1 + 1$, temos:

$$2 \times 1 + 1 = 2 \times (d_n \times 2^{n-2} + d_{n-1} \times 2^{n-3} + \ldots + d_2) + d_1$$

Logo,

$$d_1 = 1 \quad \text{e} \quad 1 = d_n \times 2^{n-2} + d_{n-1} \times 2^{n-3} + \ldots + d_3 \times 2 + d_2$$

De modo análogo, escrevemos $1 = 2 \times 0 + 1$, e assim,

$$2 \times 0 + 1 = 2\times(d_n \times 2^{n-3} + d_{n-1} \times 2^{n-4} + ... + d_3) + d_2$$

Obtemos:

$$d_2 = 1 \text{ e } 0 = d_n \times 2^{n-3} + d_{n-1} \times 2^{n-4} + ... + d_4 \times 2 + d_3$$

Como $0 = 2 \times 0 + 0$, escrevemos:

$$2 \times 0 + 0 = 2 \times (d_n \times 2^{n-4} + d_{n-1} \times 2^{n-5} + ... + d_4) + d_3$$

Isso implica que,

$$d_3 = 0 \text{ e } 0 = d_n \times 2^{n-4} + d_{n-1} \times 2^{n-5} + ... + d_4.$$

Observe que, $d_3 = d_4 = d_5 = ...= d_n = 0$, pois a divisão inteira de 0 por 2 resulta sempre em $0 = 2 \times 0 + 0$. E assim, concluímos que:

$$d_0 = 0,$$
$$d_1 = 1$$
$$d_2 = 1$$
$$d_3 = d_4 = d_5 = ... = d_n = 0$$

Portanto,

$$6 = 0...00110_2.$$

Como zeros à esquerda não têm *valor significativo*, podemos escrever simplesmente,

$$6 = 110_2.$$

Se observarmos bem, d_2 é o resto da divisão de 6 por 2, d_1 é o resto de divisão de 3 (quociente da divisão anterior) por 2, e d_0 é o resto da divisão de 1 (quociente da divisão anterior) por 2. Portanto, podemos obter 6 em base 2, usando o dispositivo prático da divisão, observando que o primeiro resto é o último algarismo (da esquerda para a direita), o segundo é o penúltimo, e assim por diante, veja a Figura 1.1.

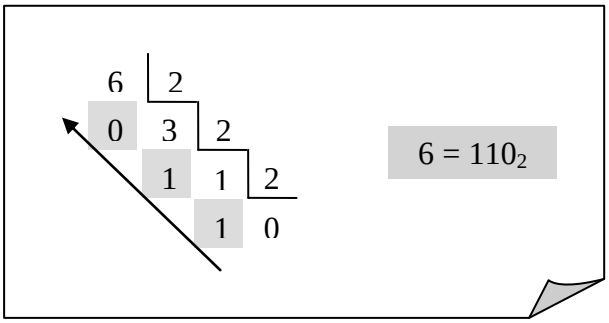

Figura 1.1. Parte inteira da mudança de base.

ii) Encontrar a representação binária da parte fracionária

Suponha que a representação binária de 0,625 (parte fracionária) seja $0,d_{-1}d_{-2}...d_{-m}$, isto é,

$$0,625 = 0,d_{-1}d_{-2}...d_{-m}$$

Multiplicando ambos os membros dessa igualdade por 2 e usando a Proposição1.1, no segundo membro, obtemos:

$$1,25 = d_{-1},d_{-2}...d_{-m} \Leftrightarrow$$

$$1 + 0,25 = d_{-1} + 0,d_{-2}...d_{-m}.$$

Comparando ambos os membros,

$$d_{-1} = 1$$

E

$$0,25 = 0,d_{-2}d_{-3}...d_{-m}.$$

Aplicando novamente o mesmo procedimento na última igualdade, isto é, multiplicando ambos os membros por 2 (usando a proposição 1.1 no segundo membro),

$$0,5 = d_{-2},d_{-3}...d_{-m} \Leftrightarrow$$

$$0 + 0,5 = d_{-2} + 0,d_{-3}...d_{-m}$$

O que nos dá

$$d_{-2} = 0$$

E

$$0,5 = 0,d_{-3}d_{-4}...d_{-m}.$$

Repetindo novamente esse procedimento,

$$1,0 = d_{-3},d_{-4}...d_{-m} \Leftrightarrow$$

$$1 + 0,0 = d_{-3} + 0,d_{-4}d_{-5}...d_{-m}$$

Logo,

$$d_{-3} = 1$$

e

$$0,0 = 0,d_{-4}d_{-5}...d_{-m}.$$

Como o primeiro membro é nulo, a repetição do processo descrito anteriormente, nos dá $d_{-4} = d_{-5} =...=d_{-m} = 0$. E assim,

$$0,625 = 0,10100...0_2.$$

Como os zeros que aparecem à direita, neste caso, são insignificantes, podemos escrever simplesmente,

$$0{,}625 = 0{,}101_2.$$

Observe que d_{-1}, d_{-2}, ..,d_{-m} são obtidos dos produtos de 2 pelo número dado e pelas respectivas partes fracionárias obtidas continuamente:

$$\begin{array}{rrrl} 0{,}625 & 0{,}25 & 0{,}5 & \\ \times\ \ 2 & \times\ \ 2 & \times\ 2 & \\ \hline 1{,}25 & 0{,}5 & 1{,}0 & \longrightarrow\ \mathbf{0{,}101_2} \end{array}$$

Observação

Neste exemplo, repetimos o processo até que a parte fracionária se tornasse nula, mas em muitas situações pode ser que isso nunca aconteça, pois muitas vezes um número que possui uma representação finita em base decimal, poderá ter uma representação infinita em binário, porém para que isso aconteça, certa seqüência de algarismos deverão se repetir indefinidamente, formando uma *dízima periódica*. Por exemplo, o número decimal 0,3 tem a seguinte representação binária,

$$0{,}010011001100110011\ldots_2$$

Observe que os algarismos 0011 se repetem indefinidamente. Nesses casos o processo termina quando detectamos a repetição ou até obtermos a quantidade de algarismos desejada. Como notação, utilizaremos uma barra sobre a parte que se repete,

$$0{,}3 = 0{,}01\overline{0011}_2$$

Portanto, juntando, em representação binária, a parte inteira com a parte fracionária encontramos $110{,}101_2$, ou seja,

$$6{,}625 = 110{,}101_2.$$

■

O processo utilizado para mudança de um número da base decimal para base binária poderá ser feito para qualquer base β, bastando proceder da mesma forma, só alterando o valor 2 para o valor β, isto é, para mudar um número da base 10 para a base β devemos:

a) *Parte inteira*: dividimos o número (divisão inteira) e os respectivos quocientes obtidos, por β, até obter um quociente igual à zero. Devemos tomar como resultado os restos encontrados (*em ordem invertida)*.

b) *Parte fracionária*: multiplicamos a parte fracionária do número e as novas partes fracionárias obtidas, por β, até obtermos uma parte fracionária nula ou até que ela comece a se repetir ou então até obtermos a quantidade de algarismos desejada.

Mudança de uma base α para uma base β

Essa mudança é feita passando primeiro o número da base α para a base 10, usando a Definição 1.1, e em seguida da base 10 para a base β, utilizando os procedimentos descritos anteriormente.

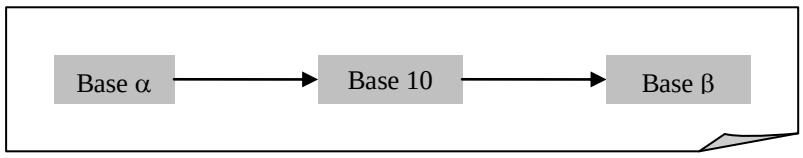

Figura 1.2. Mudança de uma base α para uma base β.

EXEMPLO 5

Passar o número $6{,}5_8$ para a base 2.

1º passo*) Mudança da base 8 para a base 10:*

Pela Definição 1.1 temos: $6{,}5_8 = 6\times8 + 5\times8^{-1} = 6{,}625$

2º passo) *Mudança da base 10 para a base 2:*

Isso já foi feito anteriormente e obtivemos $110{,}101_2$.

Portanto, $6{,}5_8 = 110{,}101_2$

1.2 Método prático para mudança de base

Através de exemplos mostraremos como é possível fazer, de forma prática, a mudança de base com o uso de uma calculadora científica comum.

EXEMPLO 6

Vamos passar o número 4325 para a base 3.

Quando dividimos um número por 3, os possíveis restos são: 0, 1 ou 2., porém a calculadora divide o resto também. A divisão de cada um dos possíveis restos citados por 3, gera os respectivos números: 0; 0,333... e 0,666... . Assim, o que aparecer após o "ponto", na calculadora, indica qual é o resto da divisão, e antes do "ponto" temos o quociente. Aplicando essa idéia ao número 4325, temos:

4325 ÷ 3	=	1441.666666	⇒	resto = **2**
1441 ÷ 3	=	480.3333333	⇒	resto = **1**
480 ÷ 3	=	160	⇒	resto = **0**
160 ÷ 3	=	53.33333333	⇒	resto = **1**
53 ÷ 3	=	17.66666666	⇒	resto = **2**
17 ÷ 3	=	5.666666666	⇒	resto = **2**

$5 \div 3 = 1.666666666 \Rightarrow$ resto = **2**

$1 \div 3 = 0.333333333 \Rightarrow$ resto = **1**

Assim, os restos obtidos são: 2, 1, 0, 1, 2, 2, 2, 1. Invertendo a ordem, obtemos o número 12221012_3.

Se usarmos a calculadora, anotando só os restos obtidos, podemos fazer mudanças da base 10 para a uma base β em um tempo bastante considerável.

EXEMPLO 7

Vamos passar 0,378 para a base 4.

Usando uma calculadora, multiplicamos 0,378 por 4 e as partes fracionárias obtidas, ou seja, os valores que aparecem após o "ponto", na calculadora.

0.378	×	4	=	**1**.512,	subtraindo	1 :	1.512 – 1	=	0.512
0.512	×	4	=	**2**.048,	subtraindo	2 :	2.048 – 2	=	0.048
0.048	×	4	=	**0**.192,	não subtrai, pois já é fracionário.				
0.192	×	4	=	**0**.768,	também não subtrai				
0.768	×	4	=	**3**.072,	subtraindo	3 :	3.072 – 3	=	0.072
0.072	×	4	=	**0**.288,	não subtrai				
0.288	×	4	=	**1**.152,	subtraindo	1 :	1.152 – 1	=	0.152
0.152	×	4	=	**0**.608,	não subtrai				
0.608	×	4	=	**2**.432,	subtraindo	2 :	2.432 – 2	=	0.432
0.432	×	4	=	**1**.728,	subtraindo	1 :	1.728 – 1	=	0.728
0.728	×	4	=	**2**.912,	subtraindo	2 :	2.912 – 2	=	0.912
0.912	×	4	=	**3**.648,	subtraindo	3 :	3.648 – 3	=	0.648
0.648	×	4	=	**2**.592,	subtraindo	2 :	2.592 – 2	=	0.592
0.592	×	4	=	**2**.368,					

Tomando-se as partes inteiras, obtidas em cada produto, temos:

$$0{,}378 = 0{,}12003010212322\ldots_4 .$$

EXEMPLO 8

Vamos passar 2105,322 para a base 5.

Parte inteira: (2105)

Possíveis restos na divisão por 5: $0 \leftrightarrow 0$; $1 \leftrightarrow 0.4$; $3 \leftrightarrow 0.6$; $4 \leftrightarrow 0.8$.

Usando a calculadora:

$$\begin{aligned} 2105 \div 5 &= 421 \Rightarrow \text{resto} = \mathbf{0} \\ 421 \div 5 &= 84.2 \Rightarrow \text{resto} = \mathbf{1} \\ 84 \div 5 &= 16.8 \Rightarrow \text{resto} = \mathbf{4} \\ 16 \div 5 &= 3.2 \Rightarrow \text{resto} = \mathbf{1} \\ 3 \div 5 &= 0.6 \Rightarrow \text{resto} = \mathbf{3} \end{aligned}$$

Logo, $2105 = 31410_5$

Parte fracionária: (0,322)

$$\begin{aligned} 0.332 \times 5 &= \mathbf{1}.61, \quad \text{subtraindo 1,} \quad 1.61 - 1 = 0.61 \\ 0.61 \times 5 &= \mathbf{3}.05, \quad \text{subtraindo 3,} \quad 3.05 - 3 = 0.05 \\ 0.05 \times 5 &= \mathbf{0}.25 \\ 0.25 \times 5 &= \mathbf{1}.25, \quad \text{subtraindo 1,} \quad 1.25 - 0.25 = 0.25 \\ 0.25 \times 5 &= \mathbf{1}.25 \end{aligned}$$

Observe que obtivemos uma dízima periódica. Logo,

$0{,}322 = 0{,}1301111\ldots_5 = 0{,}130\overline{1}$.

Portanto,

$$2105{,}322 = 31410,\ 130\overline{1}_5$$

EXEMPLO 9

Passar $43610{,}1052_7$ para a base hexadecimal.

i) Vamos passar $43610{,}1052_7$ para a base 10:

$$43610{,}1052_7 = 4 \times 7^4 + 3 \times 7^3 + 6 \times 7^2 + 1 \times 7^1 + 1 \times 7^{-1} + 5 \times 7^{-3} + 2 \times 7^{-4}$$
$$= 10934{,}15827$$

ii) Vamos passar da base 10 para a base 16 (hexadecimal)

Parte inteira: (10934)

Possíveis restos:

0	↔	0;	1	↔	0.0625;	2	↔	0.125;	3	↔	0.1875;
4	↔	0.25;	5	↔	0.3125;	6	↔	0.375;	7	↔	0.4375;
8	↔	0.5;	9	↔	0.5625;	10 = A	↔	0.625;	11 = B	↔	0.6875;
12 = C	↔	0.75;	13 = D	↔	0.8125;	14 = E	↔	0.875;	15 = F	↔	0.9375

Usando a calculadora:

10934	÷	16	=	683.375	⇒	resto = 6
683	÷	16	=	42.6875	⇒	resto = 11 = B
42	÷	16	=	2.625	⇒	resto = 10 = A
2	÷	16	=	0.125	⇒	resto = 2

Logo, $10934 = 2AB6_{16}$.

Parte fracionária: (0,15827)

Usando a calculadora:

0.15827	×	16	=	**2**.53232,	subtraindo	2;	2.53232 –	2	=	0.53232
0.53232	×	16	=	**8**.51712,	subtraindo	8;	8.51712 –	8	=	0.51712
0.51712	×	16	=	**8**.27392,	subtraindo	8;	8.27392 –	8	=	0.27392
0.27392	×	16	=	**4**.38272,	subtraindo	4;	4.38272 –	4	=	0.38272
0.38272	×	16	=	**6**.12352,	subtraindo	6;	6.12352 –	6	=	0.12352

0.12352 × 16 = **1**.97632, subtraindo 1; 1.97632 – 1 = 0.97632
0.97632 × 16 = **15**.62112, subtraindo 15; 15.62112 – 15 = 0.62112
0.62112 × 16 = **9**.93792, Subtraindo 9; 9.93792 – 9 = 0.93792

Logo, $0{,}15827 = 0{,}288461F9..._{16}$.

Portanto, $43610{,}1052_7 = 10934{,}15827 = 2AB6{,}288461F9..._{16}$.

1.3 Armazenamento na Máquina

As máquinas (calculadoras digitais e computadores) armazenam e operam os números em uma notação específica, denominada *notação de ponto flutuante*. Assim, como na *notação científica* (bastante conhecida por alunos de matemática e engenharia), a notação de ponto flutuante é uma forma de escrever um número impondo-lhes certas propriedades, como definiremos a seguir.

Definição 1.2

Seja $\beta \geq 2$ um número inteiro e $a_1, a_2, ..., a_t \in \{0, 1, ..., \beta - 1\}$, dizemos que um número real $r \neq 0$ está representado em ponto flutuante de base β e mantissa $a_1a_2...a_t$ se, r tem a forma $\pm (0,a_1a_2...a_t)_\beta \times \beta^e$, com $a_1 \neq 0$ e *e* é um número inteiro escrito em base β.

EXEMPLO 10

a) $0{,}3457 \times 10^{23}$ é um número escrito em ponto flutuante de base 10, com mantissa 3457 e expoente 23.

b) $0{,}10221 \times 3^{201_3}$ é um número em ponto flutuante de base 3, com mantissa 10221_3 e expoente 201_3.

c) $0{,}043_5 \times 5^{203_5}$ $0{,}043 \times 5^{203}$ **não** está em ponto flutuante de base 5, pois $a_1 = 0$.

Como já citamos anteriormente o armazenamento e as operações em máquina, segue uma normatização baseada na notação de ponto flutuante. A IEEE (Institute for Electrical and Eletronic Engineers) publicou normas de pontos flutuantes binários que atualmente é seguida pela maioria dos fabricantes computadores. Duas categorias foram intituladas, as de *precisão simples* que trabalham com 32-bits e as de *precisão dupla* de 64-bits. Vamos descrever como essas normas foram definidas para ambos os casos e em seguida, faremos uma generalização dessa idéia para uma dada base $\beta \geq 2$ com uma precisão de t - bits.

Precisão Simples

Nesse sistema um número é representado por uma seqüência de 32 bits binários. O bit mais a esquerda é usado para indicar o sinal da mantissa, 0 (zero) para positivo e 1 (um) para negativo. Os oito bits seguintes representam o expoente e os 23 restantes à mantissa.

Os oito bits do expoente correspondem a um número de 0 a 255 formando 256 expoentes possíveis (de 0 a 255) onde: De 0 a 126 representam os expoentes negativos, de 128 a 255 para os positivos e o número 127 representa o expoente zero. O valor, em base dez, correspondente aos oito bits do expoente armazenado denomina-se característica e é denotada por c. Assim, o número real representando na máquina é dado por:

$$c - 127.$$

EXEMPLO 11

Se uma máquina de precisão simples armazena o expoente 01001101, sua característica é,

$$c = 0 \times 2^7 + 1 \times 2^6 + 0 \times 2^5 + 0 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = 77,$$

Logo, $c - 127 = 77 - 127 = -50$, ou seja, o número real correspondente ao expoente armazenado na máquina é –50.

Note que em ponto flutuante de base 2 o primeiro bit sempre é 1, assim, todo número em ponto flutuante de base 2 é da forma,

$$\pm 0{,}1a_2a_3...a_t \times 2^e \quad \textit{que é equivalente a} \quad \pm 1{,}a_2a_3...a_t \times 2^{e-1}$$

E que corresponde ao número decimal:

$$(1 \times 2^0 + a_2 \times 2^{-1} + a_3 \times 2^{-2} + ... + a_t \times 2^{-3}) \times 2^{c-127}.$$

A característica c é a representação decimal de $e - 1$ e só a parte da mantissa $f = a_2a_3...a_t$ é representada na máquina, como mostra a figura 1.3.

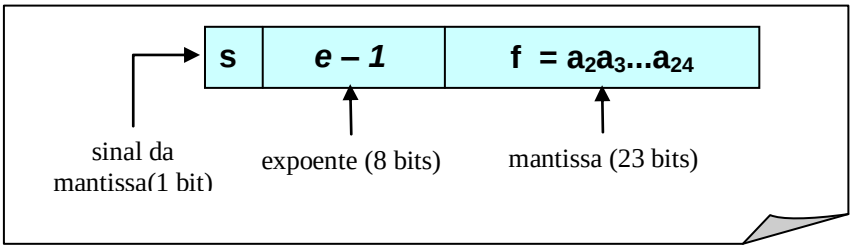

Figura 1.3 – armazenamento em precisão simples.

Portanto, um número de máquina dado pela figura 1.3 corresponde ao número:

$$(-1)^s \times (1,f)_2 \times 2^{c-127}$$

ou

$$(-1)^s \times (1 \times 2^0 + a_2 \times 2^{-1} + a_3 \times 2^{-2} + ... + a_{24} \times 2^{-23}) \times 2^{c-127}$$

EXEMPLO 12

Considere o número na máquina, representado por:

1	10000100	01101110100000000000000

s = 1

e – 1 = $10000100_2 \Rightarrow c = 2^7 + 2^2 = 132 \Rightarrow c - 127 = 132 - 127 = 5$.

f = - $01101110100000000000000_2$.

Logo o número representado na máquina é:

$(-1)^1 \times 1{,}01101110100000000000000_2 \times 2^{132-127} =$

$(-1)^1 \times (2^0 + 1\times 2^{-2} + 1\times 2^{-3} + 1\times 2^{-5} + 1\times 2^{-6} + 1\times 2^{-7} + 1\times 2^{-9}) \times 2^5 = -45{,}8185$.

Ou seja, o número armazenado na máquina é – 45,8185

EXEMPLO 13

Considere o número armazenado na máquina

0	01100001	10000011000000000000000

s = 0, $c = 2^6+2^5+2^0 = 97$ e f = 10000011000000000 $\Rightarrow$

1,f = 1,10000011000000000000000 $\Rightarrow (1,f)_{10} = 1{,}511711875$. Logo, o número armazenado na máquina é:

$(-1)^0 \times 1{,}511711875 \times 2^{97-127} = 1{,}407891395501792430877685546875 \times 10^{-9} \approx$ 0,0000000014078913955.

EXEMPLO 14

Vamos representar 4692,34 em uma máquina de precisão simples, de acordo com o IEEE.

$4692 = 1001001010100_2$ e $0{,}34 \approx 0{,}010101110000_2 \Rightarrow$

$4692{,}34 = 1001001010100{,}010101110000_2 = 1{,}00100101010001010111000_2 \times 2^{12}$

$(-1)^0 \times 1{,}00100101010001010111000_2 \times 2^{12}$.

Assim,

f = 00100101010001010111000 e $c - 127 = 12 \Rightarrow c = 139$. Logo,

$e - 1 = 139 = 10001011_2$. Portanto, a representação de 4692,34 em precisão simples é:

0	10001011	00100101010001010111000

Precisão Dupla

O sistema de precisão dupla, definido pela IEEE, é análogo ao sistema de precisão simples, porém esse sistema é composto por 64 bits, sendo 1 para o sinal da mantissa, 11 para o expoente e 52 para a mantissa, como mostra a Figura 1.4.

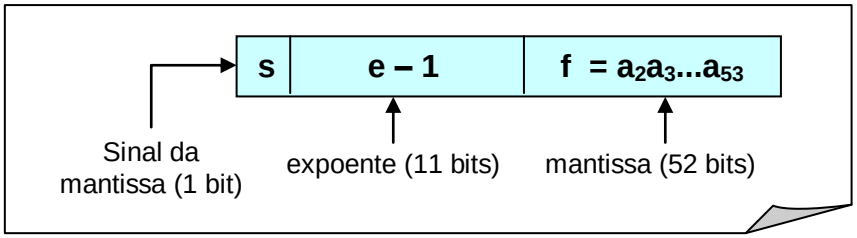

Figura 1.4 – Precisão Dupla

Um número representado em precisão dupla de acordo com a Figura 1.4 corresponde ao decimal

$$(-1)^s \times (1,f) \times 2^{c-1023}$$

ou

$$(-1)^s \times (1 \times 2^0 + a_2 \times 2^{-1} + a_3 \times 2^{-2} + ... + a_{53} \times 2^{-52}) \times 2^{c-1023}$$

Precisão de k- bits

A idéia usada para precisão simples e dupla pode ser estendida para um caso geral de k bits. Além de consideramos k bits, iremos considerar também uma base β qualquer.

Se 0, $a_1a_2...a_t \times \beta^e$ é um número em ponto flutuante de base β, sua representação no sistema k-bits é dada pela Figura 1.5.

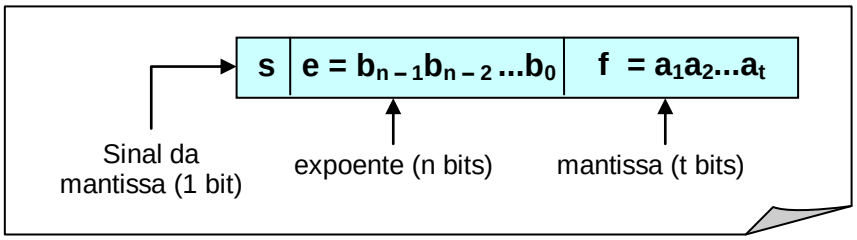

Figura 1.5 – Precisão k-bits

Observe que nesse sistema a_1 deve ser considerado, pois a base não é necessariamente 2. Além disso, precisamos conhecer n e t, isto é, a quantidade de bits do expoente e da mantissa, respectivamente.

Estendendo a idéia usada na formação da precisão simples e dupla para a precisão de k-bits podemos encontrar o número decimal correspondente ao número de máquina da Figura 1.5 pela expressão

$$(-1)^s \times 0{,}f \times \beta^{c-\lambda}$$

ou

$$(-1)^s \times (a_1 \times \beta^{-1} + a_2 \times \beta^{-2} + a_3 \times \beta^{-3} + ... + a_t \times \beta^{-t}) \times \beta^{c-\lambda}$$

Onde:

- c é a característica, isto é, a representação decimal do expoente *e*.
- λ é, em base dez, a parte inteira da metade do maior expoente representável na máquina, ou seja,

$$\lambda = \text{parte inteira de } \frac{(\beta-1)\times\beta^0 + (\beta-1)\times\beta^1 + \cdots + (\beta-1)\times\beta^{n-1}}{2}$$

EXEMPLO 15

Considere um sistema 11–bits, sendo t = 7 e n = 4 com β = 3. Vamos encontrar o número que possui a seguinte representação na máquina.

0	2101	1022001

Primeiro encontraremos o valor de λ.

λ = parte inteira de $\frac{(\beta-1)\times\beta^0+(\beta-1)\times\beta^1+\cdots+(\beta-1)\times\beta^{n-1}}{2}$ = parte inteira de $\frac{2\times3^0+2\times3^1+2\times3^2+2\times3^3}{2}$ = 40.

Para esse sistema os números têm a forma

$$(-1)^s \times (0,f)_3 \times 2^{c-40}.$$

Calculando a característica c, obtemos:

$c = 2 \times 3^3 + 1 \times 3^2 + 0 \times 3^1 + 1 \times 3^0 = 64$.

Portanto, o número decimal armazenado na máquina é

$$(-1)^0 \times (0,1022001)_3 \times 2^{64-40} = 0,4325560128 \times 3^{24} \approx 122166594,2 \times 10^3$$

Memória Excedida

Como vimos anteriormente, $c - \lambda$ é o expoente de um número representado em uma máquina de precisão k–bits. Esse expoente será positivo se $c > \lambda$, negativo se $c < \lambda$ e zero se $c = \lambda$.

Se I e S são respectivamente, o menor e maior e menor expoente armazenável na máquina, então I acontece quando $c = 0$, (pois $c \geq 0$) isto é, quando $I = c - \lambda = 0 - \lambda = -\lambda$. S ocorre quando c assumir o maior valor possível, isto é, quando $c = (\beta - 1) \times \beta^0 + (\beta - 1) \times \beta^1 + \ldots + (\beta - 1) \times \beta^{n-1}$, logo podemos obter S por:

$S = (\beta - 1) \times \beta^0 + (\beta - 1) \times \beta^1 + \ldots + (\beta - 1) \times \beta^{n-1}$ – parte inteira de $\frac{(\beta-1)\times\beta^0+(\beta-1)\times\beta^1+\cdots+(\beta-1)\times\beta^{n-1}}{2}$ = parte inteira de $[(\beta - 1) \times \beta^0 + (\beta - 1) \times \beta^1 + \ldots + (\beta - 1) \times \beta^{n-1} - \frac{(\beta-1)\times\beta^0+(\beta-1)\times\beta^1+\cdots+(\beta-1)\times\beta^{n-1}}{2}]$ = parte inteira de $\frac{(\beta-1)\times\beta^0+(\beta-1)\times\beta^1+\cdots+(\beta-1)\times\beta^{n-1}}{2} = \lambda$.

Portanto, os expoentes representáveis em uma máquina de precisão k – bits estão no intervalo [I, S] = [–λ, λ].

Se a máquina tentar armazenar um expoente menor que I ou maior que S, acontece uma sobrecarga de memória denominada, respectivamente, UNDERFLOW E OVERFLOW.

EXEMPLO 16

Em um sistema 32 – bits binário com n = 5 e t = 7, podemos calcular λ por:

λ = parte inteira de $\frac{1\times2^0+1\times2^1+1\times2^2+1\times2^3+1\times2^4+1\times2^5+1\times2^6}{2}$ = 63. Logo, todos os números na forma $\pm(1,f)_2 \times 2^e$, gera um UNDERFLOW se e < – 63 e OVERFLOW se e > 63.

1.4 Erros de Máquina

Como vimos anteriormente, as máquinas armazenam e operam os números em sistema de ponto flutuante que é limitado, e isso pode gerar erros no processo de armazenamento e nas operações numéricas. Ao usarmos a máquina para processamentos numéricos estamos submetendo os resultados a certa precisão, que depende das suas limitações físicas e que em geral é especificada pelo fabricante. Assim, em muitas situações é importante analisar as dimensões dos erros produzidos pela máquina e julgar se tais erros são aceitáveis. A seguir, faremos um estudo sobre os erros de arredondamento que podem acontecer na utilização da máquina.

Considere o número real *r*, escrito da seguinte forma:

$$r = 0,b_1 b_2 \ldots b_t b_{t+1} \ldots \times 10^e,\ b_i \in \{0, 1, 2, \ldots, 9\},\ i = 0, 1, 2, \ldots, \text{com } b_1 \neq 0.$$

Indicaremos por $fl_t(r)$ a representação de r em ponto flutuante de base 10 com mantissa de t bits. Essa representação é obtida limitando-se a mantissa de r em apenas t dígitos. Existem duas formas de se obter $fl_t(r)$:

i) Por truncamento de t dígitos, que consiste em desprezar todos os bits a partir de b_{t+1}.

ii) Por arredondamento para t dígitos, que consiste em adicionar $5 \times 10^{e-(t+1)}$ a *r* e fazer o truncamento no resultado obtido. Se observarmos bem, esse processo é equivalente a: adicionar 1 em b_t se $b_{t+1} \geq 5$ e em seguida fazer o truncamento de t dígitos ou simplesmente fazer o truncamento de t dígitos se $b_{t+1} < 5$.

EXEMPLO 17

$\sqrt{113}$ = 10,63014581... = 0,1063014581... $\times 10^2$.

Por truncamento:

$fl_7(\sqrt{113}) = 0{,}1063014 \times 10^2$.

Por arredondamento:

$5 \times 10^{e-(t+1)} = 5 \times 10^{2-(7+1)} = 5 \times 10^{-6} = 0{,}000005 = 0{,}00000005 \times 10^2$. Logo, 0,1063014581... $\times 10^2$ + 0,00000005 $\times 10^2$ = 0,1063015081... . Fazendo o truncamento com 7 dígitos, obtemos

$fl_7(\sqrt{113}) = 0{,}1063015$.

Assim, dizemos que o truncamento e arredondamento de $\sqrt{113}$ com 7 dígitos significativos são, respectivamente 10,63014 e 10,63015.

O erro cometido ao se trocar r por $fl_t(r)$ é denominado erro de *arredondamento* (mesmo que $fl_t(r)$ seja obtido por truncamento). O termo "*erro de truncamento"*, em geral, é usado para se referir ao erro cometido ao se calcular uma aproximação de uma série infinita, simplesmente desprezando uma infinidade de termos dela, a partir de um termo fixado, e maiores detalhes fogem ao nosso propósito. Portanto, o erro cometido pela máquina sempre será referido como *arredondamento.*

Essa idéia pode ser generalizada para outras bases.

Erro Absoluto e Relativo

O erro cometido pela aproximação em ponto flutuante pode ser medido como absoluto ou relativo. O erro absoluto está relacionado com a a distância, na reta numérica, entre o valor aproximado; e o erro relativo,compara proporcionalmente a dimensão dessa distância com o valor exato.

Definição 1.3

Se r é um número real, não nulo, definimos respectivamente, erro absoluto e relativo gerado por $f_t l(r)$ por :

$$E_A = |r - f_1 l(r)| \qquad \text{e} \qquad E_R = \frac{|r - f_1\, l(r)|}{|r|}$$

Seja $r = 0b_1b_2 \ldots b_tb_{t+1} \ldots x\ 10^e$ ($b1 \neq 0$). Se $f_t l(r) = 0, b_1b_2 \ldots b_t \times 10^e$ é obtido pelo método de truncamento, então

$$E_R = \frac{|r - f_1\, l(r)|}{|r|} = \frac{|0{,}b_1b_2 \ldots b_t \ldots x10^e - 0{,}b_1b_2 \ldots b_t \ldots x10^e|}{|0{,}b_1b_2 \ldots b_t \ldots x10^e|} = \frac{|0{,}b_{t+1}b_{t+2} \ldots|}{|0{,}b_1b_2 \ldots|} \times 10^{-1}$$

Como, $|0, b_{t+1}b_{t+2}| < 1$ e $|0, b_1b_2 \ldots| \geq 0{,}1$ ($b \neq 0$), temos:

$$E_R \leq \frac{1}{0{,}1} x\ 10^{-1} \Rightarrow E_R \leq 10^{-t+1}$$

Considerando agora que $f_t l(r)$ seja obtido pelo método de arredondamento, temos:

$f_t l(r) = 0{,}b_1b_2 \ldots b_t \ldots x10^e + 5 \times 10^{e-(t+1)} = 0{,}b_1b_2 \ldots b_t \ldots x10^e + 0{,}5 \times 10^e \times 10^{-t.}$

Logo,

$E_R = \frac{|r - f_1\ l(r)|}{|r|} = \frac{|0,b_1b_2...b_t...x10^e - (0,b_1b_2...b_t...x10^e + 0,5 \text{ x } 10^e \text{ x } 10^{-1}|}{|0,b_1b_2...b_t...x10^e|} =$

$\frac{|0,b_{t+1}b_{t+2} - 0,5 \text{ x } 10^{-1}|}{|0,b_1b_2...b_t|} = \frac{|0,b_{t+1}b_{t+2} - 0,5\ |}{|0,b_1b_2...b_t|} x\ 10^{-1.}$

Como

$|0, b_{t+1}b_{t+2}| < 1$ temos $|0, b_{t+1}b_{t+2} ... - 0,5| < 0,5$

E ainda $|0, b_1b_2 ...| \geq 0,1, pois(\ b\ \neq 0)$,

Assim,

$$E_R \leq \frac{0,5}{0,1} \text{ x } 10^{-1} \implies E_R \leq 0,5 \text{ x } 10^{-t+1}$$

Portanto:

$E_R \leq 10^{-t+1}$ pelo método de truncamento

$E_R \leq 0,5 \text{ x } 10^{-t+1}$ pelo método de arredondamento

Para estimar o erro relativo em outras bases basta seguir o mesmo processo, não é necessário saber como efetuar as operações nessas bases. Em binário encontramos:

$E_R \leq 2^{1-t}$ pelo método de truncamento

$E_R \leq 2^{-t}$ pelo método de arredondamento

Definição 1.4

Dizemos que r* se aproxima de r com k dígitos significativos se k é o maior inteiro não negativo tal que

$$\frac{|r - r*|}{|r|} < 5 \times 10^{-k}$$

EXEMPLO 18

Os números reais r* que se aproximam de r = 2,7182 com três algarismos significativos satisfaz a inequação:

$\frac{|2,7182-r^*|}{|2,7182|} < 5 \times 10^{-3} \Rightarrow |2,7182 - r^*| < 0,013591 \Rightarrow$ 2,704609 < x < 2,731791.

1.4 Exercícios Propostos

1) Escreva o número r na base β especificada.

a) r = 2,3459 e β = 2
b) r = -2342,3459 e β = 3
c) r = 888,777 e β = 4
d) r = 45764,9876 e β = 16
e) r = $10010,001101_2$ e β = 10
f) r = $1000101,00011_2$ e β = 16
g) r = $10221,201221_3$ e β = 2
h) r = $12342,3401_5$ e β = 3
i) r = A01B,34CF e β = 10
j) r = 2E3B,5E1 e β = 2
k) r = $100110,01101_2$ e β = 16