Cálculo Numérico

Faculdade de Engenharia, Arquiteturas e Urbanismo – FEAU

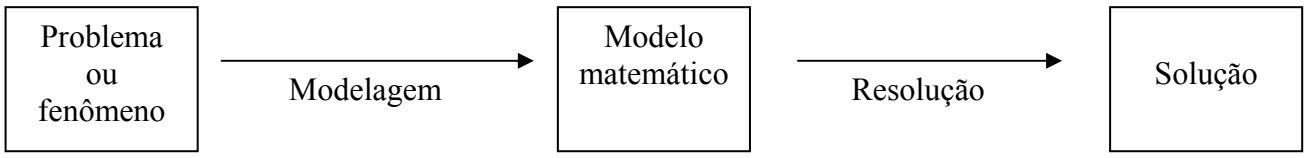
Prof. Dr. Sergio Pilling (IPD/ Física e Astronomia)

UNIVERSIDADE DO VALE DO PARAÍBA
São José dos Campos - SP

I – Representação dos números, aritmética de ponto flutuante e erros em máquinas digitais.

Objetivos: Alertar o aluno sobre as dificuldades numéricas que podem ocorrer ao se trabalhar com um computador (ou qualquer outra maquina digital); Erros inerentes ao processo de tradução de números decimais para números binários.

1 – O processo de modelagem de um fenômeno da natureza.

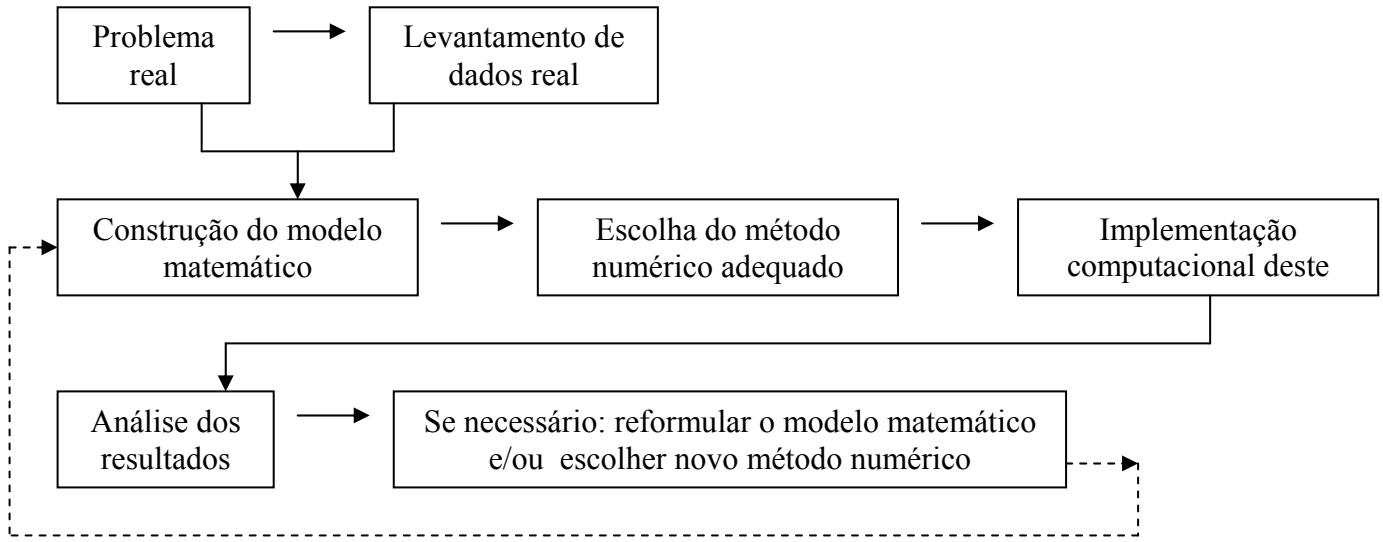


Modelagem – Fase de obtenção de um modelo matemático que descreve o comportamento do problema que se quer estudar.

Resolução – Fase de obtenção da solução do modelo matemático através da aplicação de métodos numéricos.

Obs: Ambas as fases acima estao passíveis de erros.

De forma mais detalhada temos:



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:

a) da precisão dos dados de entrada
b) da forma como esses dados são representados no computador
c) das operações numéricas efetuadas

2 – Representação dos números.

Os números empregados no calculo computacional podem ser de dois tipos: números inteiros e números em "ponto flutuante" (números reais da matemática, por exemplo $3.56 \rightarrow 0.356 \times 10^{-1}$). Os computadores atuais representam os números internamente no formato binário, como uma seqüência de 0s e 1s. Apesar dessa representação ser conveniente para as maquinas é antinatural para os seres humanos, cujo sistema de numeração é o decimal.

Obs. No passado o nosso sistema de numeração já foi também na base 12 (ex. contar nas falanges dos dedos) na base 60 (ex. sistema horário).

2.1 – Decomposição de um número num sistema de bases.

Em geral qualquer numero pode ser decomposto numa soma dos dígitos que o constitui (d) vezes potências da sua base (β) conforme indicado abaixo:

Atencão!

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

Onde os dígitos d_j pertencem aos números naturais e satisfazem a condição: $0 \le d_j \le (\beta-1)$

2.2 – Sistema de numeração decimal ou base 10.

Nesse caso todos os múltiplos e submúltiplos de um número são escritos com potencias de 10.

Ex1. $1537 = (1537)_{10} = 1 \times 10^3 + 5 \times 10^2 + 3 \times 10^1 + 7 \times 10^0$

$36{,}189 = (36{,}189)_{10} = 3 \times 10^1 + 6 \times 10^0 + 1 \times 10^{-1} + 8 \times 10^{-2} + 9 \times 10^{-3}$

$6{,}032\text{x}10^{23} = (6{,}032\text{x}10^{23})_{10} = 6 \times 10^{23} + 0 \times 10^{22} + 3 \times 10^{21} + 2 \times 10^{20}$

2.3 – Sistema de numeração binário ou base 2.

Nesse caso todos os múltiplos e submúltiplos de um número são escritos com potencias de 2.

Ex2. $(10111)_2 = 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

$(10{,}1)_2 = 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1}$

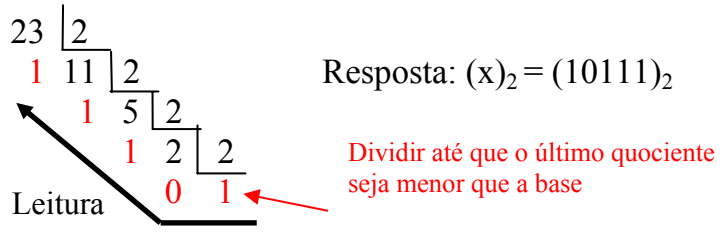
Obs. Os computadores digitais operam basicamente com dois tipos de sinais de tensão: Alto e baixo. Matematicamente, pode-se expressar esses valores por 0 (baixo) e 1 (alto).

3 – Conversão de números

3.1 – Conversão de números decimal → binário.

Para convertermos um numero decimal para um numero binário devemos aplicar um método para a parte inteira (divisões sucessivas) e um método para a parte fracionaria, se houver (multiplicações sucessivas).

Ex3. $(23)_{10} \rightarrow (x)_2$ Usando o método das divisões sucessivas.

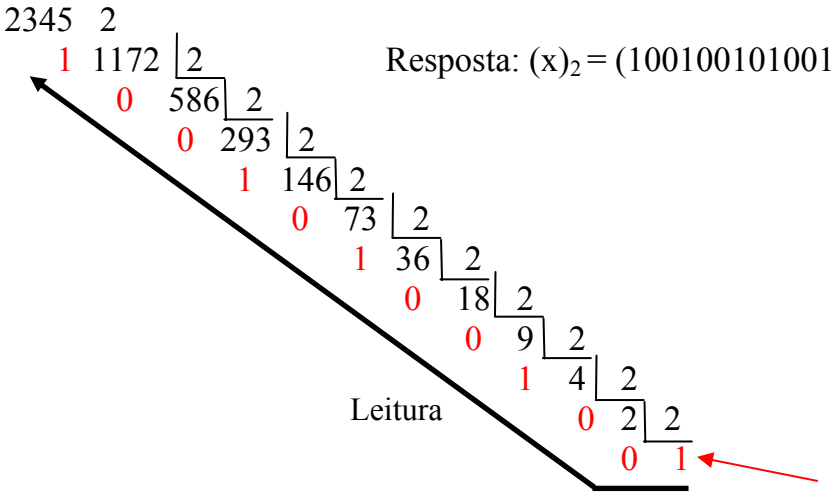


Resposta: $(x)_2 = (10111)_2$

Dividir até que o último quociente seja menor que a base

Portanto, a partir de uma seqüência de 0s e de 1s podemos expressar "qualquer" número decimal. Sera?

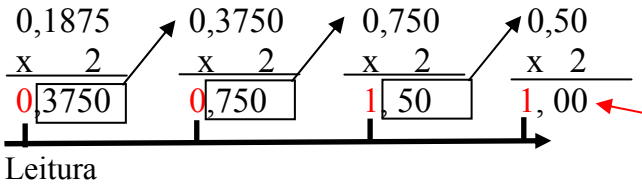
Ex4. $(2345)_{10} \rightarrow (x)_2$ Usando o método das divisões sucessivas.



Resposta: $(x)_2 = (100100101001)_2$

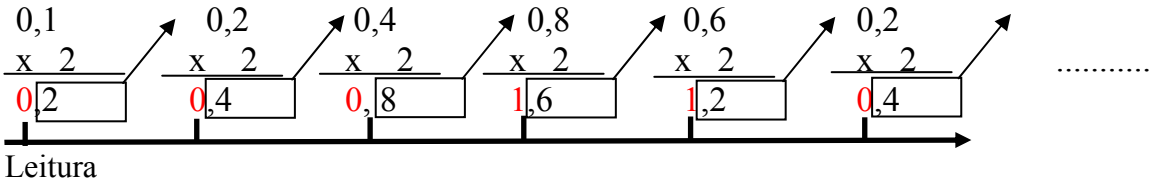
Para números fracionários utilizamos a regra da multiplicação.

Ex5. $(0,1875)_{10} \rightarrow (x)_2$



Resposta: $(x)_2 = (0,0011)_2$

Ex6. $(0,1)_{10} \rightarrow (x)_2$

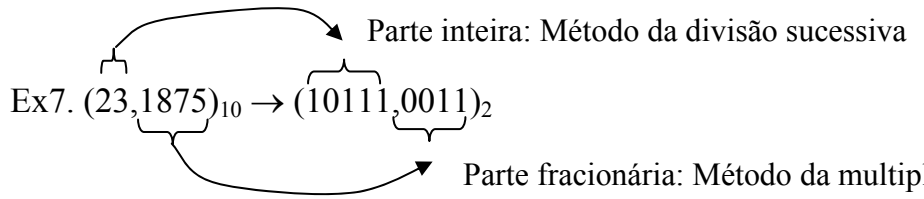


Resposta: $(x)_2 = (0,0001100110011.....)_2$

Repetições

Nesse caso concluímos que o número $(0,1)_{10}$ NÃO tem representação binária finita !!! → Por mais moderno que seja o computador ele nunca vai saber exatamente o que significa o numero $(0,1)_{10}$ pois sua conversão para binário sempre acarretará numa aproximação (truncamento u arredondamento)

Obs. O fato de um número não ter representação finita no sistema binário pode acarretar a ocorrência de erros aparentemente inexplicáveis nos cálculos dos dispositivos eletrônicos.



3.2 – Conversão de números binário → decimal.

Ex8. $(10111)_2 \rightarrow (x)_{10}$

$$(10111)_2 = \underbrace{1x2^4}_{16} + \underbrace{0x2^3}_{0} + \underbrace{1x2^2}_{4} + \underbrace{1x2^1}_{2} + \underbrace{1x2^0}_{1} = 23 = (23)_{10}$$

d_4 d_0 d_1

Ex9. $(110,11)_2 \rightarrow (x)_{10}$

$$(110,11)_2 = \underbrace{1x2^2}_{4} + \underbrace{1x2^1}_{2} + \underbrace{0x2^0}_{0} + \underbrace{1x2^{-1}}_{1/2} + \underbrace{1x2^{-2}}_{1/4} = 6,75 = (6,75)_{10}$$

d_2 d_{-1} d_0

Obs. Um numero inteiro decimal pode sempre ser representado exatamente por um inteiro binário. Mas isso não é verdade para os números fracionários. Pois, já vimos que:

$(0,1)_{10} = (0,0001100110011001100110011.....)_2$

Repetições

$(0,11)_{10} = (0,000111000010100011110101110000101000111101.....)_2$

Repetições

4 – Operações aritméticas entre números binários

A) Adição de binários.

Propriedades:
0+0=0
0+1=1
1+0=1
1+1= 0 e vai 1 (para somar ao digito imediatamente à esquerda)

```
Ex10:                Ex11:
     1                   11
     1100                 1100
  +   111              +  1111
    -----                -----
  = 10011              = 11011
```

Explicando: Os números binários são base 2, ou seja, há apenas dois algarismos: 0 (zero) ou 1 (um). Na soma de 0 com 1 o total é 1. Quando se soma 1 com 1, o resultado é 2, mas como 2 em binário é 10, o resultado é 0 (zero) e passa-se o outro 1 para a "frente", ou seja, para ser somado com o próximo elemento, conforme assinalado pelo asterisco,como no exemplo acima.

B) Subtração de binários.

Propriedades:
0-0=0
0-1= 1 e vai 1 (para subtrair ao digito imediatamente à esquerda)
1-0=1
1-1=0

Ex12:

```
      1 111
     1101110
-      10111
     -------
=    1010111
```

Explicando: Quando temos 0 menos 1, precisamos "pedir emprestado" do elemento vizinho. Esse empréstimo vem valendo 2 (dois), pelo fato de ser um número binário. Então, no caso da coluna 0 - 1 = 1, porque na verdade a operação feita foi 2 - 1 = 1. Esse processo se repete e o elemento que cedeu o "empréstimo" e valia 1 passa a valer 0. Os asteriscos marcam os elementos que "emprestaram" para seus vizinhos. Perceba, que, logicamente, quando o valor for zero, ele não pode "emprestar" para ninguém, então o "pedido" passa para o próximo elemento e esse zero recebe o valor de 1.

C) Multiplicação de binários

A multiplicação entre binários é similar à realizada com números decimais. A única diferença está no momento de somar os termos resultantes da operação:

Ex13:

```
          1 0 1 1
        x 1 0 1 0
        ---------
          0 0 0 0
  +     1 0 1 1
  +     0 0 0 0
  + 1 0 1 1
  ---------------
  = 1 1 0 1 1 1 0
```

Ex14:

```
              1 1 1
          x   1 1 1
          ---------
            1 1
          1 1 1 1 1
  +       1 1 1 1
  +     1 1 1 1
    ---------------
    =   1 1 0 0 0 1
```

C) Divisão de binários

Essa operação também é similar àquela realizada entre números decimais:

Ex15:

```
   110 |10
 - 10  |---
 ------|11
   010
 -  10
 ------
    00
```

Exercícios:

1) converta os números decimais em sua forma binária:
a) 2 b)10 c) 7550 d)13,25 e) 0,4217

2) Converta os números binários em sua forma decimal:
a) $(10100)_2$ b) $(1101)_2$ c)$(0,1101)_2$ d) $(11101,01)_2$

5 – Representação dos números no formato "ponto flutuante" e sua aritmética.

A representação de números reais mais utilizada em máquinas é a do ponto flutuante . Esse número tem três partes: o sinal, a parte fracionária (mantissa) e o expoente,

$$m = \pm\ ,d_1d_2d_3\ldots\ d_t \times \beta^e$$

sendo

$d_{i's}$: dígitos da parte fracionária, $d_1 \neq 0$, $0 \leq d_i \leq \beta\text{-}1$
β: base (em geral 2, 10 ou 16),
t: no de dígitos na mantissa.
e: expoente inteiro.

Equivalente à:
$(0,00011001100110011.....)_2$

Ex.

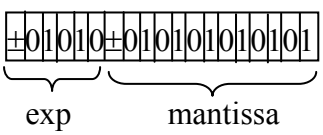
x=34,2 (decimal); β=10; t=4 $x=0,3420 \times 10^2$	x=0,1 (decimal) ; β=2; t=9 $x=0,110011001\times 2^{-3}$

Nas maquinas digitais, um digito binário é denominado BIT (do inglês, binary digit). Um grupo de oito bits corresponde a 1 byte. Dessa forma, percebemos que a representação dos números binários num computador é feita com um número finito de bits. A esse tamanho finito de bits é dado o nome ***palavra de computador.*** O tamanho da palavra do computador depende de características internas à arquitetura do mesmo. Em geral, os microcomputadores padrão PC tem tamanho de palavra de 16 e 32 bits. Computadores modernos tem palavras de 64 bits ou mais.

Quanto maior o tamanho da palavra do computador mais veloz e mais preciso será o computador.

Uma máquina digital (que opera em base 2) armazena um número internamente da seguinte forma esquematizada abaixo:

$(N)_{10}$ → Not. Ponto flutuante $\boxed{\begin{matrix}(\text{exp})_{10}\\(\text{mantissa})_{10}\end{matrix}}$ → Conversão para binário $\boxed{\begin{matrix}(\text{exp})_{2}\\(\text{mantissa})_{2}\end{matrix}}$ → Palavra do Computador



Parâmetros de aritméticas de ponto flutuante utilizadas em alguns computadores digitais.

Máquina e Aritmética	β	t	$e_{\min}$	$e_{\max}$
Cray-1 Precisão Simples	2	48	-8192	8191
Cray-1 Precisão Dupla	2	96	-8192	8191
DEC VAX formato G Dupla	2	53	-1023	1023
DEC VAX formato D Dupla	2	56	-127	127
Calculadoras HP 28 e 48G	10	12	-499	499
IBM 3090 Precisão Simples	16	6	-64	63
IBM 3090 Precisão Dupla	16	14	-64	63
IBM 3090 Precisão Extendida	16	28	-64	63
IEEE Precisão Simples	2	24	-126	127
IEEE Precisão Dupla	2	53	-1022	1023
PDP 11	2	24	-128	127
Control Data 6600	2	48	-976	1070

Mais detalhes sobre a aritmética de ponto flutuante:

Uma aritmética de ponto flutuante F é caracterizada por quatro números inteiros: $F(\beta, t, e_{\min}, e_{\max})$. Pode-se observar que F é um subconjunto dos números reais, ou seja $F \subset \Re$.

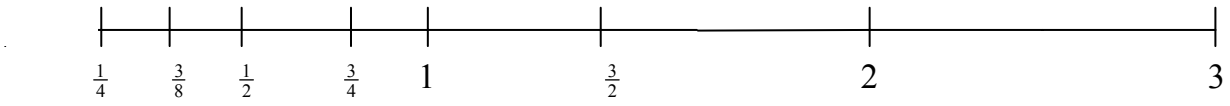
Ex:
Considere $F(2,2,-1,2)$, com número normalizado, isto é, $d_1 \neq 0$. Os números serão:
$\pm .10 \times 2^e$ ou $\pm .11 \times 2^e$, sendo $-1 \leq e \leq 2$.

Convertendo para decimal, temos:
$.10 = ½$ e $.11 = ¾$

Com isso, os únicos números positivos representáveis nesse computador são:

Mantissa	Expoentes
$1/2 \times 2^e$	
$3/4 \times 2^e$	para e= -1, 0, 1 e 2

Ou seja, ¼, ½, 1, 2, 3/8, 3/4, 3/2 e 3, que podem ser representados na reta numerada:



Alem desses números, os seus respectivos números negativos e o numero zero também serão representados.

5.1 - O número total de elementos de uma aritmética de ponto flutuante é dado por:

$$n.\ de\ elementos = 2(\beta - 1)\beta^{t-1}(e_{\max} - e_{\min} + 1) + 1$$

Para contabiliza os números negativos — Para o numero zero

Para o exemplo anterior temos que o número de elementos é 17. (8 positivos, 8 negativos e o zero).

O conjunto dos números de ponto flutuante é discreto, e não contínuo como os números reais. Não temos mais o conceito que entre dois números sempre existe um outro. Esse fato pode ter conseqüência desastrosa!

5.2 - Erros na representação dos números

O conjunto de números de números reais é infinito, entretanto, a sua representação em um sistema de ponto flutuante é limitada, pois é um sistema **finito**. Essa limitação tem duas origens:

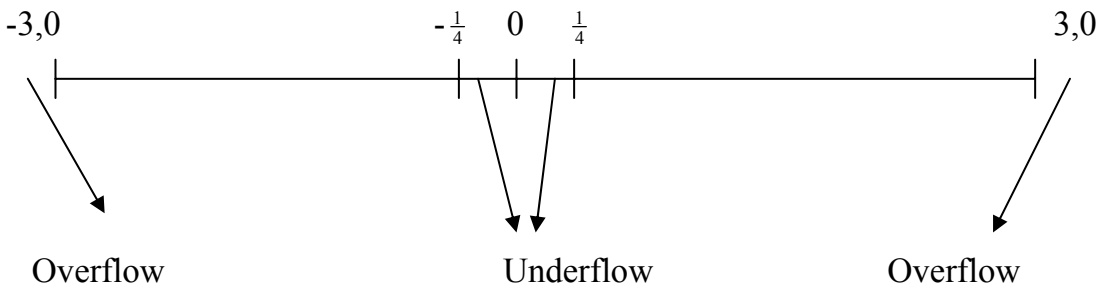
A) a faixa dos expoentes é limitada ($e_{\min} \le e \le e_{\max}$);

B) a mantissa representa um número finito de números ($\beta^{t-1} \le m \le \beta^{t-1}$)

Faixa dos expoentes é limitada ($e_{\min} \le e \le e_{\max}$);

Sempre que uma operação aritmética produz um número com expoente superior ao expoente máximo, tem-se o fenômeno de "**overflow**". De forma similar, operações que resultem em expoente inferior ao expoente mínimo tem-se o fenômeno de "**underflow**".

No caso do exemplo dado, pode-se observar qual as regiões que ocorrem o overflow e o underflow. Neste caso, considera-se a parte positiva e negativa da aritmética do exemplo.



Ex1: Considere uma aritmética de ponto flutuante *F*(10,2,-5,5)

-**overflow**: Sejam x =875 e y=3172 . Calcular x × y.
Primeiro, deve-se arredondar os números e armazena-los no formato indicado. A operação de multiplicação é efetuada usando 2t dígitos.
$x = 0.88 \times 10^{3}$ e $y = 0.32 \times 10^{4}$, $x \times y = 0.2816 \times 10^{7}$
Como o expoente é maior que 5, resulta em overflow

-**underflow**: Sejam x =0,0064 e y=7312 Calcular x ÷ y.
Primeiro, deve-se arredondar os números e armazená-los no formato indicado. A operação de divisão é efetuada usando 2t dígitos.
$x = 0.64 \times 10^{-2}$ e $y = 0.73 \times 10^{4}$, $x \div y = 0.8767 \times 10^{-6}$
O resultado dessa operação resultou em um valor menor que o computador pode armazenar, ou seja, resulta em **underflow**

Mantissa representando um número finito de números ($\beta^{t-1} \leq m \leq \beta^{t-1}$)

Ex1. Seja uma máquina que opere com apenas **6 dígitos** na mantissa, ou seja, que seja capaz de armazenar números no formato $m = \pm 0,d_1d_2d_3d_4d_5d_6 \times 10^e$. Como armazenaríamos número

$$(0,11)_{10} = (0,000111000010100011110101110000101000111101\overline{.....})_2$$

Repetições

nesta maquina?

Como o número $(0,11)_{10}$ que não tem representação binária finita, teremos neste caso:

$(0,11)_{10} \rightarrow (0,000111)_2 \rightarrow (0,109375)_{10}$

Para de armazenar.

Ex2. Considere a representação binária de 0,6 e 0,7.
0,6=0,100110011001... 0,7=0,1011001100110...
Se esses dois números forem representados na aritmética $F(2,2,-1,2)$ eles serão representados igualmente por 0.10×2^0. Esse número equivale a 0,5 em decimal. Portanto, tanto o 0,6 quanto o 0,7 serão considerados 0,5.

Ex3. Operações em $F(10,2,-5,5)$.
- Sejam x =4,32 e y=0,064 Calcular x + y.

A adição aritmética de PF requer o alinhamento dos pontos decimais dos dois números.
$x = 0.43x\ 10^{1}$ e $y = 0.0064 \times 10^{1} \rightarrow$ **$x+y = 0.4364 \times 10^{1}$**
Resultado com 2 dígitos : $x+y = 0.44 \times 10^{1}$

- Sejam x =372 e y=371 Calcular x - y.

$x = 0.37x\ 10^{3}$ e $y = 0.37 \times 10^{3} \rightarrow$ **$x-y = 0.00 \times 10^{0}$**
Resultado com 2 digitos **$x-y = 0.00 \times 10^{0}$**

- Sejam x =691 e y=2,71 Calcular x + y.

$x = 0.69x\ 10^{3}$ e $y = 0.0027 \times 10^{3} \rightarrow$ **$x+y = 0.6927 \times 10^{3}$**
Resultado com 2 dígitos : $x+y = 0.69 \times 10^{1}$

6 – Erros

Nenhum resultado obtido através de cálculos eletrônicos ou métodos numéricos tem valor se não tivermos conhecimento e controle sobre os possíveis erros envolvidos no processo.

A análise dos resultados obtidos através de um método numérico representa uma etapa fundamental no processo das soluções numéricas.

6.1. Número aproximado

Um número $\tilde{x}$ é dito uma aproximação para o número exato x se existe uma *pequena* diferença entre eles. Geralmente, nos cálculos os números exatos não são conhecidos e deste modo são substituidos por suas aproximações. Dizemos que $\tilde{x}$ é um número aproximado por falta do valor exato x se $\tilde{x} < x$. Se $\tilde{x} > x$ temos uma aproximação por excesso.

Exemplo

Como $1.41 < \sqrt{2} < 1.42$ temos que 1.41 uma aproximação de $\sqrt{2}$ por falta e 1.42 uma aproximação de $\sqrt{2}$ por excesso.

6.2 Erro absoluto EA_x

A diferença entre um valor exato x e sua aproximação $\tilde{x}$ é dito **erro absoluto** o qual denotamos por EA_x

$$EA_x = | x - \bar{x} |$$

Cota para o erro.

Na prática, o valor exato é quase sempre não conhecido. Como o erro é definido por $EA_x = | x - \bar{x} |$ conseqüentemente também será não conhecido. Uma solução para este problema é ao invés de determinar o erro determinar uma **cota** para o erro. Isso permitirá que, mesmo não conhecendo o erro, saber que ele está entre dois valores conhecidos.
Dizemos que um número $\epsilon > 0$ é uma cota para o erro EA_x se $EA_x < \epsilon$

$$\therefore \quad EA_x < \epsilon \iff |x - \tilde{x}| < \epsilon \iff \tilde{x} - \epsilon < x < \tilde{x} + \epsilon$$

Assim, mesmo não conhecendo o valor exato, podemos afirmar que ele esta entre $\tilde{x} - \epsilon$ e $\tilde{x} + \epsilon$ que são valores conhecidos.

É evidente que uma cota ϵ só tem algum valor prático se $\epsilon \approx 0$

6.3 Erro relativo ER_x

Considere : $x = 100$; $\tilde{x} = 100.1$ e $y = 0.0006$; $\tilde{y} = 0.0004$.

Assim $EA_x = 0.1$ e $e_y = 0.0002$.

Como EA_y é muito menor que EA_x poderiamos "imaginar" que a aproximação $\tilde{y}$ de y é *melhor* que a $\tilde{x}$ de x. Numa análise mais cuidadosa percebemos que as grandezas dos números envolvidos são muito diferentes.

Inspirados nessa observação definimos:

$$ER_x = \left| \frac{EA_x}{\bar{x}} \right| = \left| \frac{x - \bar{x}}{\bar{x}} \right|$$

que é denominado **erro relativo**. Temos então para os dados acima:

$ER_x = EA_x/\tilde{x} = 0.1/100.1 = 0.000999$

$ER_y = EA_y/\tilde{y} = 0.0002/0.0006 = 0.333333$

Agora podemos concluir que a aproximação $\tilde{x}$ de x é melhor que a $\tilde{y}$ de y pois $ER_x < |ER_y$.

6.4 Outras fontes de erros numa maquina digital

A) Erros relacionados à aproximações dos cálculos (números de iterações).

São os erros causados quando utilizamos num processo algorítmico infinito apenas uma parte finita do processo.

Exemplo 2.3.1

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{n=1}^{\infty} \frac{x^n}{n!}$$

Podemos assim usar $1 + x + x^2/2! + x^3/3!$ como uma aproximação para o valor exato e^x. Observe que para isso truncamos uma série infinita utilizando apenas uma parte finita dela. No exemplo utilizamos para a aproximação apenas quatro termos da série.

Usando a aproximação acima temos:

$\tilde{e} = 1+1+1/2+1/6 = 2.66666$ que é uma aproximação muito pobre para e

B) Erros devido ao armazenamento.

Erros de Arredondamento são os erros originados pela representação dos números reais utilizando-se apenas um número finito de *casas decimais*.
Como se sabe, desde a mais simples calculadora até o mais potente computador, utiliza apenas um número finito de *casas decimais* para representar um número real.(número real é denominado *número de ponto flutuante* nas linguagens de programação)
Dizemos então que os equipamentos eletrônicos utilizam nos cálculos a chamada *aritmética finita.*

Existem dois tipos de erros por arredondamento. O arredondamento truncado e o arredondamento simétrico (ou arredondamento propriamente dito).

Regra para o truncamento:

- Desprezam-se os algarismos que ficam acima da (t+1)-ésima casa decimal. Onde t representa o numero de dígitos da mantissa.

Regras para o arredondamento:

- *Se o valor do algarismo que fica na $(t+1)$-ésima casa decimal for menor do que 5 arredondamos o número desprezando-se todos algarismos após a t-ésima casa decimal;*
- *Se for maior ou igual a 5 soma-se 1 ao algarismo na t-ésima casa decimal e desprezam-se os algarismos restantes.*

Ex.

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)	UNDERFLOW
718235.82	(expoente maior que 4)	OVERFLOW

C – Valores limitantes para os erros de armazenamento.

Uma vez que as maquinas digitais apresentam erros devido ao tipo de armazenamento (truncamento e arredondamento), ao calcularmos os erros relativos (ou absolutos) devemos adicionar um termo para contabilizar esse erro extra (δ).

O erro em uma dada aproximação numérica (δ) em será no mínimo sempre maior do que um certo fator diferente de zero:

$$\delta < 10^{-t+1}$$
no caso do truncamento.

$$\delta < \tfrac{1}{2}\, 10^{-t+1}$$
no caso do arredondamento.

onde t = número de dígitos da mantissa.

Maiores detalhes podem ser obtidos no livro texto.

7 – Propagação dos erros.

A) Propagação dos erros absolutos.

Seja $\tilde{x}$ uma aproximação para x e $\tilde{y}$ uma aproximação para y ou seja $EA_x = x - \tilde{x}$ e $EA_y = y - \tilde{y}$ Então temos:

Soma e Subtração

$$EA_{(x \pm y)} = | EA_x \pm EA_y |$$

Multiplicação

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

Divisão

$$EA_{(x/y)} = \left| \frac{EA_x}{\bar{y}} - \frac{\bar{x} EA_y}{\bar{y}^2} \right|$$

B) Propagação dos erros relativos.

1. Soma e Subtração

$$ER_{(x\pm y)} = \frac{EA_x \pm EA_y}{\tilde{x} \pm \tilde{y}} = \frac{EA_x}{\tilde{x} \pm \tilde{y}} \pm \frac{EA_y}{\tilde{x} \pm \tilde{y}} = \frac{\tilde{x}}{\tilde{x} \pm \tilde{y}} ER_x \pm \frac{\tilde{y}}{\tilde{x} \pm \tilde{y}} ER_y$$

onde a ultima igualdade foi conseguida substituindo EA_x por $\tilde{x}ER_x$ e EA_y por $\tilde{y}ER_y$

$$\boxed{ER_{(x\pm y)} = \frac{\tilde{x}}{\tilde{x} \pm \tilde{y}} ER_x \pm \frac{\tilde{y}}{\tilde{x} \pm \tilde{y}} ER_y}$$

2. Multiplicação

$$ER_{xy} = \frac{EA(xy)}{\tilde{x}\tilde{y}} = \frac{\tilde{x}EA_y + \tilde{y}EA_x}{\tilde{x}\tilde{y}} = \frac{\tilde{x}EA_y}{\tilde{x}\tilde{y}} + \frac{\tilde{y}EA_x}{\tilde{x}\tilde{y}} = \frac{EA_y}{\tilde{y}} + \frac{EA_x}{\tilde{x}} = ER_x + ER_y$$

$$\boxed{ER_{xy} = ER_x + ER_y}$$

3. Divisão

$$ER_{(\frac{x}{y})} = \frac{EA(\frac{x}{y})}{\frac{\tilde{x}}{\tilde{y}}} = \frac{\tilde{y}}{\tilde{x}} EA(\frac{x}{y}) = \frac{\tilde{y}}{\tilde{x}}(\frac{EA_x}{\tilde{y}} - \frac{\tilde{x}}{(\tilde{y})^2} EA_y) = \frac{EA_x}{\tilde{x}} - \frac{EA_y}{\tilde{y}} = ER_x - ER_y$$

$$\boxed{ER_{(\frac{x}{y})} = ER_x - ER_y}$$

Concluindo, o erro relativo final das operações é obtido a partir da combinação dos erros relativos da operação adicionado ao erro devido a tipo de armazenamento numérico (δ):

Soma e Subtração

$$ER_{(x\pm y)} = \left| \frac{\bar{x}}{\bar{x} \pm \bar{y}} ER_x \pm \frac{\bar{y}}{\bar{x} \pm \bar{y}} ER_y \right| + \delta$$

Multiplicação

$$ER_{(xy)} = | ER_x + ER_y | + \delta$$

Divisão

$$ER_{(x/y)} = | ER_x - ER_y | + \delta$$

onde o fator δ, associado ao erro devido ao fato do computador trabalhar com números truncados ou arredondados é dado por:

$\delta = 10^{-t+1}$ (no caso de truncamento)

$\delta = ½\ 10^{-t+1}$ (no caso de arredondamento)

onde t = número de dígitos da mantissa.

Ex.

Um coreano ganhou de presente do pai uma máquina de calcular super moderna, capaz de armazenar 4 dígitos na mantissa utilizando **arredondamento**. Muito satisfeito, o ansioso rapaz efetuou duas operações em sua maquina nova envolvendo os números de arvores da plantação de seu pai (x=17534) e o número médio de frutas de cada arvore (y=21178).

a) Calcule os erros absolutos e relativos envolvido no processo de utilização da máquina digital para cada número x e y?

Resp: Devido ao tamanho da mantissa e ao arredondaremos termos:

$\bar{x} = 0{,}1753 \text{ x } 10^5$ e $\bar{y} = 0{,}2118 \text{ x } 10^5$

a) Calcule o erro absoluto e relativos das variáveis x e y.

Resp.

$$EA_x = |x - \bar{x}| = |17534\text{-}17530| = 4$$

$$EA_y = |y - \bar{y}| = |21178\text{-}21180| = 2$$

$$ER_x = \left|\frac{EA_x}{\bar{x}}\right| = 4/17530 = 2.281\text{E-}4$$

$$ER_y = \left|\frac{EA_y}{\bar{y}}\right| = 2/21180 = 9.442\text{E-}5$$

a) Após realizar as operações x+y e x×y percebeu que uma das duas operações resultava no erro relativo final maior. Qual foi?

Resp: $ER_{x+y} = \left|\frac{\bar{x}}{\bar{x}+\bar{y}} ER_x + \frac{\bar{y}}{\bar{x}+\bar{y}} ER_y\right| + \delta$, onde $\delta = ½\ 10^{-t+1}$ (arredondamento)

$= (17530/\ 38710)\ 2.281\text{E-}4 + (21180/\ 38710)\ 9.442\text{E-}5 + ½\ 10^{-4+1}$

$= 1.0329\text{E-}4 + 5.1661\text{e-}5 + 5\text{E-}4 = 6.5495\text{E-}4$

$ER_{xy} = ER_x + ER_y + \delta$, onde $\delta = ½\ 10^{-t+1}$ (arredondamento)

$= 2.281\text{E-}4 + 9.442\text{E-}5 + 5\text{E-}4 = 8.2242\text{E-}4$

Logo a operação xy apresentara o maior erro relativo final.

d) Calcule o erro relativo envolvido na operação x^4?

Resp: $ER_{xx} = ER_{x^2} = ER_x + ER_x + \delta$, onde $\delta = ½\ 10^{-t+1}$ (arredondamento)

$$ER_{xx^2} = ER_{x^3} = ER_x + ER_{x^2} + \delta = 3ER_x + 2\delta$$

$$ER_{xx^3} = ER_{x^4} = ER_x + ER_{x^3} + \delta = 4ER_x + 3\delta$$

logo,

$$ER_{x^4} = 4\text{x } 2.281\text{E-}4 + 3\text{x } 5\text{E-}4 = 2.4124\text{E-}3$$

Repita este exercício considerando o truncamento.

Exercícios Resolvidos

1- Considere uma maquina cujo sistema de representação de números é definido por: base decimal, 4 dígitos na mantissa (t=4), e expoentes no intervalo (-5,5). Pede-se:
A) Qual o menor e o maior número, em módulo, representados nesta maquina?
Resp. $m=0.1000\times10^{-5} = 10^{-6}$ $M=0.9999\times10^{5} = 99990$

B) Como será representado o número 73.758 nesta maquina, se for usado o arredondamento? E se for usado o truncamento?
Resp. 0.7375×10^{2} (trucamento) e 0.7376×10^{2} (arredondamento)

C) Se a=42450 e b=3 qual o resultado de a+b?
Resp. $a+b= 0.4245\times10^{5} + 0.00003\times10^{5} = 0.42453\times10^{5}$ (NAS OPERAÇÕES ENVOLVENDO NÚMEROS PONTO FLUTUANTES ESTES DEVEM TER A POTÊNCIA DO MAIOR DELES)

Mas o resultado será armazenado com 4 dígitos na mantissa portanto $a+b=0.4245\times10^{5}$

D) Qual o resultado da somas $S1 = 42450 + \sum_{k=1}^{10} 3$ e $S2= \sum_{k=1}^{10} 3 + 42450$ nesta maquina?

Obs. Obviamente o resultado deveria ser o mesmo. Contudo, as operações devem ser ralizadas na ordem em que aparecem as parcelas, o que conuzirá a resultados distintos.

Resp. $S1= 42450 + 3 + 3 + 3 + 3 + 3 + 3 + 3 + 3 + 3 + 3$

$= 0.4245\times10^{5} + 0.00003\times10^{5} = 0.42453\times10^{5} \rightarrow 0.4245\times10^{5}$ (representado na mantissa com 4 dígitos)

depois teremos

$0.4245\times10^{5} + 0.00003\times10^{5} = 0.42453\times10^{5}$ (idem)

depois teremos

$0.4245\times10^{5} + 0.00003\times10^{5} = 0.42453\times10^{5}$

.... até terminar a última soma individual
$S1 = 0.4245\times10^{5}$

$S2 = 3 + 3 + 3 + 3 + 3 + 3 + 3 + 3 + 3 + 3 + 42450$

$= 0.3000\times10^{1} + 0.3000\times10^{1} = 0.6000\times10^{1}$

$= 0.6000\times10^{1} + 0.3000\times10^{1} = 0.9000\times10^{1}$

$= 0.9000\times10^{1} + 0.3000\times10^{1} = 1.2000\times10^{1} = 0.1200\times10^{2}$

$= 0.1200\times10^{2} + 0.03000\times10^{2} = 0.1500\times10^{2}$

.... ate terminar o ultimo 3 que resultara no numero 0.3000×10^{2}. Depois é feita a soma com o número 42450 e no final teremos:
$0.3000\times10^{2} \rightarrow 0.0003\times10^{5}$
$S2=0.0003\times10^{5} + 0.4245\times10^{5} = 0.4248\times10^{5}$

Exercício Proposto

1- Seja um sistema de aritmética de ponto flutuante de quatro dígitos e base decimal. Dados os números: $x= 0{,}7237\times10^{4}$, $y = 0.2145\times10^{-3}$ e $z = 0.2585\times10^{-1}$ efetue as operações x+y+z e (xy)/z e obtenha o erro relativo em cada caso, supondo que x, y e z estão exatamente representados (Ea=Eb=Ec=0)