Tópico 1 - Representação Numérica e Erros de Aproximação

João B. Florindo

Instituto de Matemática, Estatística e Computação Científica
Universidade Estadual de Campinas - Brasil
jbflorindo@ime.unicamp.br

Outline

Introdução

Cálculo numérico está na interface entre a Matemática, a Ciência da Computação e as Aplicações

O Cálculo tradicional trabalha no contínuo e mesmo lá muitos problemas não possuem solução fechada. Ex.: $\int_0^1 \mathrm{e}^{-x^2} dx$

Além disso, o mundo real é discreto e a modelagem contínua envolve algum tipo de aproximação. Ex.: Tenho os valores (domínio discreto) da velocidade instantânea e quero saber a distância percorrida.

Introdução

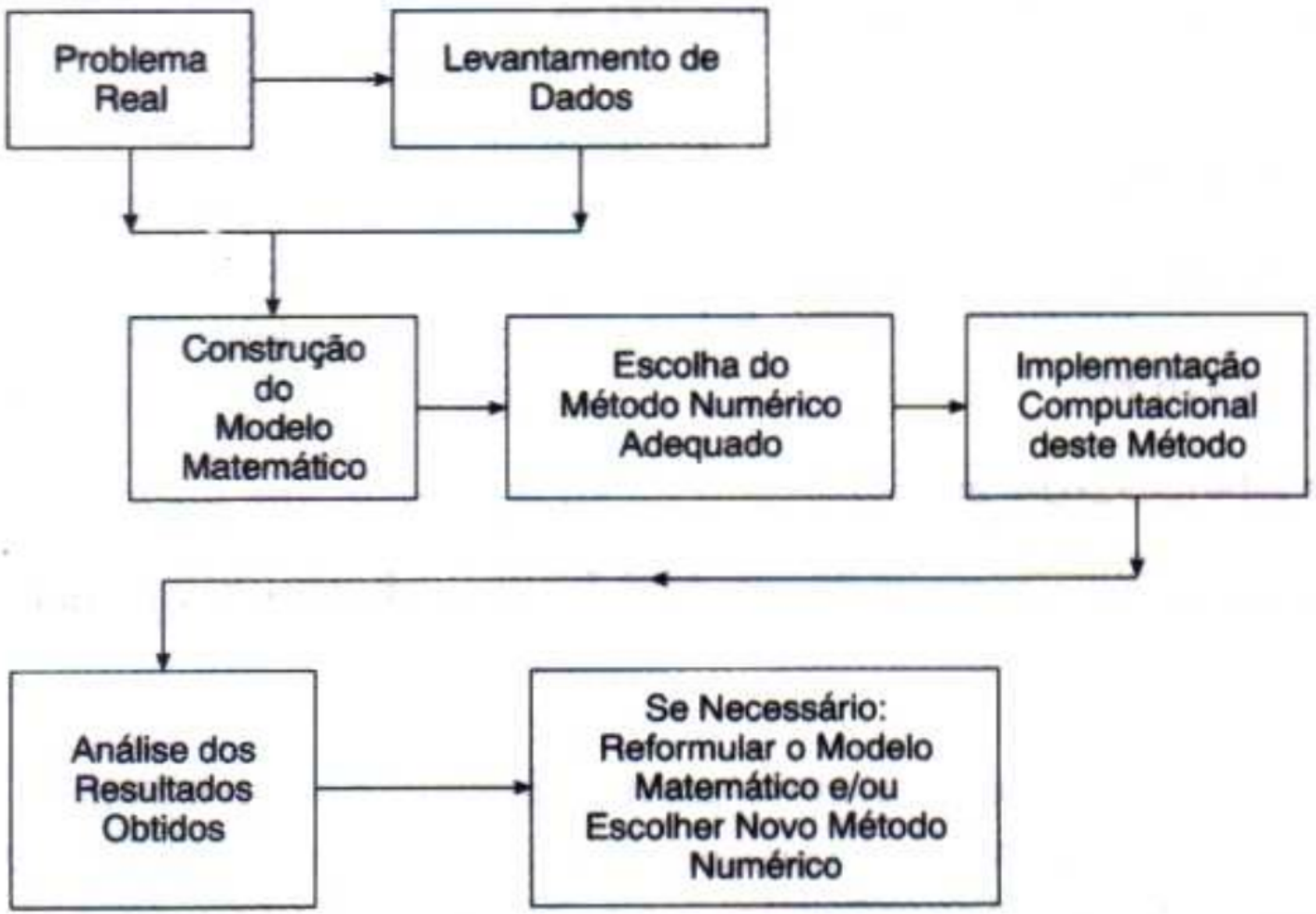

Figura: Fonte: Ruggiero&Lopes

Introdução

Problema: o mundo real é discreto e a modelagem contínua envolve alguma aproximação = ERRO!

Experimental ou de Modelo: Sabemos que na eletricidade $V = RI$. Medimos $V = 10$ e $I = 5$. Em seguida, dobramos a voltagem $V = 20$ e medimos $I = 11$!? A lei de Ohm não funciona no circuito (existem outros elementos que não foram contados na modelagem)? Ou temos um erro experimental? Como quantificar e se adequar à imprecisão do aparelho de medida?

Representação no computador: representação finita em sistema binário. Erros de arredondamento. Ex. no Octave: $0.55 + 0.55 + 1 + 1 - 3.1 = 0$. Mas: $1 + 1 + 0.55 + 0.55 - 3.1 = -4.4409e-16$ (??!!) Erro pode parecer pequeno mas se propaga por operações (tente multiplicar por $1e16$!!!). Ordem importa!

Introdução

- Resultados numéricos incorretos são comuns:
  - Dados de entrada imprecisos
  - Representação de números em computadores
  - Características inerentes a cada operação

- Representação de números em computadores, aritmética e erros associados

$$S(x) = \sum_{i=1}^{30000} x$$

Na calculadora: $S(0.5) = 15000$, $S(0.11) = 3300$
No computador: $S(0.5) = 15000$, $S(0.11) = 3299.99691!!??$ $\Rightarrow$ Problema:
$(0.11)_2 = 0.000\overline{1110000101000111101}0\ldots$

Outline

Representação Numérica

- No dia-a-dia usamos representação decimal

$$832 = 8 \cdot 10^2 + 3 \cdot 10^1 + 2 \cdot 10^0$$

- Bases 12, 60, etc. já foram usuais no passado e outras como 8 e 16 são ainda úteis
- Computadores usam base binária (2)

$$10010 = 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0 = 18$$

Representação Numérica

- Em geral, temos o seguinte polinômio:

$$N = a_j\beta^j + a_{j-1}\beta^{j-1} + \cdots + a_2\beta^2 + a_1\beta^1 + a_0\beta^0$$

representado $(a_j a_{j-1} \ldots a_2 a_1 a_0)_\beta$, para $0 \leq a_k \leq (\beta - 1)$

Algoritmo 1: Converter de uma base β qualquer para decimal:

$b_j \leftarrow a_j$;
$b_{j-1} \leftarrow a_{j-1} + b_j\beta$;
$b_{j-2} \leftarrow a_{j-2} + b_{j-1}\beta$;
...
$b_0 \leftarrow a_0 + b_1\beta$;
retorna b_0;

Representação Numérica

Algoritmo 2: Conversão de número em base genérica β para decimal:

$N \leftarrow a_j$;
para $k = j - 1$ **até** 0 **faça**
  $N \leftarrow a_k + N\beta$;
fim
retorna N;

Representação Numérica

Demonstração (polinômios aninhados):

$N = (a_j\beta^{j-1} + a_{j-1}\beta^{j-2} + \cdots + a_2\beta + a_1)\beta + a_0 =$

$((a_j\beta^{j-2} + a_{j-1}\beta^{j-3} + \cdots + a_2)\beta + a_1)\beta + a_0 =$

$\ldots$

$N = (((a_j\beta + a_{j-1})\beta + a_{j-2})\beta + a_{j-3})\beta \cdots + a_0$

Demonstração de cima para baixo:

$b_j = a_j$

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

$b_{j-2} = a_{j-2} + b_{j-1}\beta = a_{j-2} + (a_{j-1} + a_j\beta)\beta = a_{j-2} + a_{j-1}\beta + a_j\beta^2$

$b_{j-3} = a_{j-3} + b_{j-2}\beta = a_{j-3} + (a_{j-2} + a_{j-1}\beta + a_j\beta^2)\beta =$

$a_{j-3} + a_{j-2}\beta + a_{j-1}\beta^2 + a_j\beta^3$

...

Algoritmo 1:

Converter $(1101)_2$ para decimal
$b_3 = 1$
$b_2 = a_2+b_3*2 = 1+1*2 = 3$
$b_1 = a_1+b_2*2 = 0+3*2 = 6$
$b_0 = a_0+b_1*2 = 1+6*2 = 13$

Converter $(187)_{10}$ para binário
$b_2 = (1)_2$
$b_1 = a_1+b2*(1010)_2 = (1000)_2 + (1)_2*(1010)_2 = (10010)_2$
$b_0 = a_0+b_1*(1010)_2 = (111)_2 + (10010)_2*(1010)_2$

```
  1000
+1010
________
10010
```

```
          1 0 0 1 0
          x 1 0 1 0
      ______________

          0 0 0 0 0
        1 0 0 1 0 +
      0 0 0 0 0 + +
    1 0 0 1 0 + + +
    _______________
    1 0 1 1 0 1 0 0
```

```
     1
10110100
+       111
________
10111011
```

```
          1 1 1
        x 1 1 1
_______________
    10  10  1
          1 1 1
        1 1 1 +
      1 1 1 + +
_______________
     11  0 0 0 1
```

$(110001)_2 = 2^5+2^4+2^0 = 49$

Representação Numérica

- Algoritmo 1 pode converter decimal para binário usando-se aritmética binária
- Uso de muitos dígitos complica aritmética binária. Preferível: decimal $\rightarrow$ octal $\rightarrow$ binário. Ou então, aplicar algoritmo abaixo

Algoritmo 3: Conversão do número decimal N para binário:

$k \leftarrow 0$;
$N_k \leftarrow N$;
enquanto $q_k \neq 0$ **faça**
    Obter q_k e r_k tal que $N_k = 2q_k + r_k$;
    $a_k \leftarrow r_k$;
    $N_k \leftarrow q_k$;
    $k \leftarrow k + 1$;
fim
retorna $a_n a_{n-1} \ldots a_0$;

Representação Numérica

Demonstração
Usamos novamente a forma aninhada:
$N = a_0 + 2(a_1 + 2(a_2 + 2(a_3 + 2(\cdots + 2(a_{j-1} + 2a_j)))))$
Note como a cada divisão por 2 o quociente é a parte dentro dos parêntesis e o resto são examente os coeficientes $a_0, a_1, \ldots, a_{j-1}$.
O último coeficiente a_j é o próprio quociente.
Como $a_j < 2$ o quociente da próxima divisão $a_j/2$ é 0, o resto é a_j e o algoritmo para.

Algoritmo 2:
Converter $(187)_{10}$ para binário
$k = 0, N_0 = 187$
Passo 1:
$q_0 = 93, r_0 = 1$
$a_0 = 1, N_0 = 93, k = 1$
Passo 2:
$q_1 = 46, r_1 = 1$
$a_1 = 1, N_1 = 46, k=2$
Passo 3:
$q_2 = 23, r_1 = 0$
$a_2 = 0, N_2 = 23, k=3$
Passo 4:
$q_3 = 11, r_3 = 1$
$a_3 = 1, N_3 = 11, k=4$
Passo 5:
$q_4 = 5, r_4 = 1$
$a_4 = 1, N_4 = 5, k=5$
Passo 6:
$q_5 = 2, r_5 = 1$
$a_5 = 1, N_5 = 2, k=6$
Passo 7:
$q_6 = 1, r_6 = 0$
$a_6 = 0, N_6 = 1, k=7$
Passo 8:
$q_7 = 0, r_7=1$
$a_7 = 1$ PARAR
$(a_7a_6...a_0)_2 = (10111011)_2$

Outline

Representação Numérica Fracionária

- E os números não-inteiros?
- Número real positivo x
- Parte inteira x_I: maior inteiro $\leq x$
- Parte fracionária $x_F = x - x_I$

$$x_F = \sum_{j=1}^{\infty} b_j 10^{-j}$$

Representação Numérica Fracionária

- A parte fracionária é finita se $b_j = 0$ para todo j maior que determinado inteiro
- Por exemplo, a representação de $\frac{1}{3}$ não termina:

$$\frac{1}{3} = 3 \cdot 10^{-1} + 3 \cdot 10^{-2} + 3 \cdot 10^{-3} + \ldots$$

- Números binários são representados da mesma forma

$$x_F = \sum_{j=1}^{\infty} b_j 2^{-j}$$

$$x = (a_n a_{n-1} \ldots a_0 . b_1 b_2 b_3 \ldots)_2$$

Representação Numérica Fracionária

Algoritmo 4: Dado x entre 0 e 1, e chamando as partes inteira e fracionária de $()_I$ e $()_F$, sempre podemos obter sua representação em uma base genérica β:

$c_0 \leftarrow x$;
$j \leftarrow 0$;
enquanto $c_j \neq 0$ **faça**
  $b_{j+1} \leftarrow (\beta c_j)_I$, $c_{j+1} \leftarrow (\beta c_j)_F$;
  $j \leftarrow j + 1$;
fim
retorna $x \leftarrow (.b_1 b_2 b_3 \ldots)_\beta = \sum_{j=1}^{\infty} b_j \beta^{-j}$;

- Manualmente, melhor fazer decimal $\rightarrow$ octal $\rightarrow$ binário
- Mesmo algoritmo para converter de binário para decimal usando aritmética binária, representando β como $(1010)_2$

Representação Numérica Fracionária

Demonstração

Forma aninhada novamente, agora para x entre 0 e 1:

$x = b_1\beta^{-1} + b_2\beta^{-2} + b_3\beta^{-3} + b_4\beta^{-4} + \cdots + b_j\beta^{-j} =$

$= \beta^{-1}(b_1 + \beta^{-1}(b_2 + \beta^{-1}(b_3 + \beta^{-1}(b_4 + \beta^{-1}(\cdots + \beta^{-1}(b_{j-1} + b_j\beta^{-1}))))))$

Note-se aqui a semelhança com o algoritmo anterior, da divisão por 2

Multiplicando-se por β no primeiro passo obtemos:

$\beta c_0 = \beta x =$

$b_1 + \beta^{-1}(b_2 + \beta^{-1}(b_3 + \beta^{-1}(b_4 + \beta^{-1}(\cdots + \beta^{-1}(b_{j-1} + b_j\beta^{-1})))))$

A parte inteira contém b_1 pois sabemos que b_1 é um inteiro entre 0 e $\beta - 1$. Já o lado direito (em vermelho) sabemos ser fracionário pois é equivalente a $b_2\beta^{-1} + b_3\beta^{-2} + \ldots$, que obviamente é menor do que 1.

Repetindo o procedimento de multiplicar a parte fracionária por β sempre encontramos b_k na parte inteira até o momento em que a parte fracionária for 0 e então todos os b_k's seguintes serão sempre 0 e não terão mais nenhum significado.

ALGORITMO 3
Converter $(0.625)_{10}$ para binário.
$c_0 = 0.625$
$j = 0$
Passo 1
$b_1 = 1, c_1 = 0.25$
$j=1$
Passo 2
$b_2 = 0, c_2 = 0.5$
$j=2$
Passo 3
$b_3 = 1, c_3 = 0$
PARAR
$(0.625)_{10} = (0.101)_2$

Converter $(0.11)_{10}$ para binário.
$c_0 = 0.11$
$j = 0$
Passo 1
$b_1 = 0, c_1 = 0.11$
$j=1$
Passo 2
$b_2 = 0, c_2 = 0.22$
$j=2$
Passo 3
$b_3 = 0, c_3 = 0.44$
$j=3$
Passo 4
$b_4 = 0, c_4 = 0.88$
$j=4$
Passo 5
$b_5 = 1, c_5 = 0.76$
$j=5$
Passo 6
$b_6 = 1, c_6 = 0.52$
$j=6$
Passo 7
$b_7 = 1, c_7 = 0.04$
$j=7$
Passo 8
$b_8 = 0, c_8 = 0.08$
$j=8$
Passo 9
$b_9 = 0, c_9 = 0.16$
$j=9$
Passo 10
$b_{10} = 0, c_{10} = 0.32$
$j=10$

Representação Numérica Fracionária

Por que $\sum_{i=1}^{30000} 0.11 \neq 3300$?

- 0.11 possui representação infinita no sistema binário
- Supondo um computador com 6 dígitos decimais:

$$(0.11)_{10} \approx (0.000111)_2$$

Mas:

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

Outline

Aritmética de Ponto Flutuante

- Representação clássica em cálculos científicos

$$x = \pm(.d_1 d_2 \ldots d_n)_\beta \beta^e,$$

sendo $(.d_1 d_2 \ldots d_n)_\beta$ a mantissa e e o expoente.

- **Normalizado** se $d_1 \neq 0$ ou $d_1 = d_2 = \cdots = d_n = 0$
- $\beta = 2$ (binário): PC; $\beta = 10$ (decimal): calculadoras
- Valor de n define a precisão: *single*, *double*, ...
- Expoente deve ser delimitado:

$$m < e < M$$

Aritmética de Ponto Flutuante

- Conjunto de todos os pontos flutuantes possíveis para os parâmetros dados é denotado por

$$F(\beta, n, m, M)$$

ATENÇÃO: Alguns textos usam a letra t em lugar de n.

EXEMPLO:

No sistema $F(10, 4, -5, 5)$, representar os seguintes números:

$73.75 = 0.7375 \times 10^2$
$132700 = 0.1327 \times 10^6$ (OVERFLOW!)
$0.0000008391 = 0.8391 \times 10^{-6}$ (UNDERFLOW!)
Qual o maior valor representável?
0.9999×10^5
E o menor (excluindo 0)?
$0.1000 \times 10^{-5} = 10^{-6}$

Aritmética de Ponto Flutuante

- Em geral, tratamento especial se $|x| \geq \beta^M$ (*overflow*) ou $0 < |x| \leq \beta^{m-1}$ (*underflow*)
- O *zero* normalmente é representado pelo menor expoente possível para evitar perdas de dígitos significativos na adição

Exemplo de problema com representação de 0 por mantissa nula:

$x + y = 0.0000 \times 10^0 + 0.8732 \times 10^{-2} = 0.0000 \times 10^0 + 0.0087 \times 10^0 = 0.0087 \times 10^0 = 0.87 \times 10^{-2}$
PERDEU DÍGITO SIGNIFICATIVO DEVIDO AO **CANCELAMENTO**!

Outline

Padrão IEEE

- Padrão IEEE 754: 1 bit para sinal, 11 para expoente (**característica**) e 52 para fração (**mantissa**):

$$x = (-1)^s(1.d_{51}d_{50}\ldots d_0) \times 2^{e-1023} \Rightarrow F = (2, 52, -1023, 1024)$$

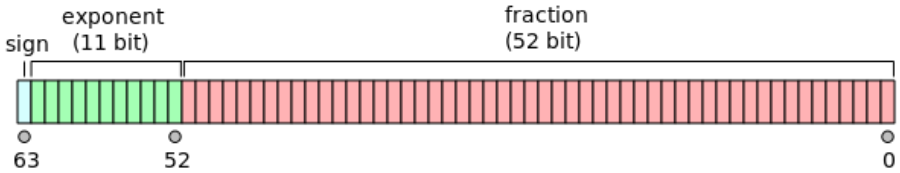

Figura: IEEE 754 para *double*. Fonte: Wikipedia

Padrão IEEE

Considere o exemplo:
$x =$0 10000000011 10111001000100000000000000000000000000000000000000
$s = 0$ (número positivo), $c =$ 10000000011
$e = 1 \cdot 2^{10} + 0 \cdot 2^9 + \cdots + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 1024 + 2 + 1 = 1027$
$f = 1 \cdot \left(\frac{1}{2}\right)^1 + 1 \cdot \left(\frac{1}{2}\right)^3 + 1 \cdot \left(\frac{1}{2}\right)^4 + 1 \cdot \left(\frac{1}{2}\right)^5 + 1 \cdot \left(\frac{1}{2}\right)^8 + 1 \cdot \left(\frac{1}{2}\right)^{12}$
$x = (-1)^s 2^{e-1023}(1+f) =$
$(-1)^0 \cdot 2^{1027-1023}\left(1 + \left(\frac{1}{2} + \frac{1}{8} + \frac{1}{16} + \frac{1}{32} + \frac{1}{256} + \frac{1}{4096}\right)\right) = 27.56640625$
Note-se que evidentemente números reais são "perdidos" nesta representação pois o número imediatamente menor que x é
0 10000000011 10111001000011111111111111111111111111111111111111
$\approx 27.566406249999998\ldots$
e o maior
0 10000000011 10111001000100000000000000000000000000000000000001
$\approx 27.566406250000002\ldots$
x representa metade deste intervalo

Padrão IEEE

Dois números são reservados: $e = 0$ (tudo 0 no vetor binário da característica) para 0 com sinal se $f = 0$ ou subnormais (zeros nas primeiras posições da mantissa $[0.f]$ usados para preencher lacuna entre 0 matemático e 0 da máquina mesmo perdendo precisão) se $f \neq 0$ e $e = 2047$ (tudo 1 em binário) para ∞ se $f = 0$ ou NAN se $f \neq 0$

Deste modo, o menor número que pode ser representado com $s = 0$, $e = 1$ e $f = 0$:

$$2^{-1022} \cdot (1 + 0) \approx 2.2251 \times 10^{-308}$$

e o maior com $s = 0$, $e = 2046$ e $f = 1 - 2^{-52}$:

$$2^{1023} \cdot (1 + 1 - 2^{-52}) \approx 1.7977 \times 10^{308}$$

Outline

Conversão para Ponto Flutuante

- **Arredondamento:** $fl(x)$ é o ponto flutuante normalizado mais próximo de x
- **Truncamento:** $fl(x)$ é o ponto flutuante normalizado mais próximo entre 0 e x
  Exemplo: $\pi \approx 0.314159265\cdots \times 10^1$
  Truncamento 5 dígitos: $fl(\pi) = 0.31415 \times 10^1$
  Arredondamento 5 dígitos: $fl(\pi) = 0.31416 \times 10^1$

Conversão para Ponto Flutuante

Representar 73.758 em $F(10, 4, -5, 5)$

Truncamento: 0.7375×10^2
Arredondamento: 0.7376×10^2

$\Rightarrow$ **Erro de aproximação:** diferença entre x e $fl(x)$ dependente de x:

$$fl(x) = x(1 + \delta(x))$$

Conversão para Ponto Flutuante

Axioma dos Pontos Flutuantes

Se ω^* é a operação em PF correspondente à operação analítica ω:

$$x\omega^* y = fl(x\omega y)$$

Operações alternativas

$$x \oplus y = fl(fl(x) + fl(y) \quad x \ominus y = fl(fl(x) - fl(y)$$
$$x \otimes y = fl(fl(x) \times fl(y) \quad x \oslash y = fl(fl(x) \div fl(y)$$

Operações em PF costumam gerar resultados com comprimentos diferentes. Exemplo:

$$x = 1 = (.10)10^1;\ y = 0.0021 = (.21)10^{-2}$$
$$x + y = (.10021)10^1$$

Conversão para Ponto Flutuante

ARITMÉTICA DE PONTO FLUTUANTE
Representar 73.758 em F(10,4,-5,5)
Truncamento: 0.7375×10^2
Arredondamento: 0.7376×10^2

OPERAÇÕES EM PONTOS FLUTUANTES
$a = 0.4523 \times 10^4$, $b = 0.2115 \times 10^3$, $c = 0.2583 \times 10^1$, $d = 0.9857 \times 10^2$
Considerar arredondamento

$(a+b)+c = (0.4523 \times 10^4 + 0.0211 \times 10^4) + 0.0002 \times 10^4 = 0.4736 \times 10^4$
$a+(b+c) = 0.4523 \times 10^4 + (0.2115 \times 10^3 + 0.0026 \times 10^3) = 0.4523 \times 10^4 + 0.2141 \times 10^3 =$
$= 0.4523 \times 10^4 + 0.0214 \times 10^4 = 0.4737 \times 10^4$ (NÃO É ASSOCIATIVO)

$axb = 0.4523 \times 10^4 \times 0.2115 \times 10^3 = (0.4523 \times 0.2115) \times 10^7 = 0.09566145 \times 10^7 = 0.9566 \times 10^6$
(OVERFLOW!!! Máximo = 0.9999×10^5)

$bx(c+d) = 0.2115 \times 10^3 \times (0.2583 \times 10^1 + 0.9857 \times 10^2) = 0.2115 \times 10^3 \times 0.1011 \times 10^3 = 0.02138265 \times 10^6 =$
0.2138×10^5
$bxc + bxd = 0.5463 \times 10^3 + 0.2085 \times 10^5 = 0.2140 \times 10^5$ (NÃO É DISTRIBUTIVO)

Outline

Erro de Aproximação

- Seja x o valor exato e $\overline{x}$ o aproximado
- Erro absoluto: $x - \overline{x}$
- Erro relativo: $(x - \overline{x})/\overline{x}$ ((Conte & deBoor) e (Burden & Faires) dividem por x)
  Exemplos:
  $x = 0.3100 \times 10^1, \overline{x} = 0.3000 \times 10^1 \Rightarrow EA = 0.1, ER = 0.3333 \times 10^{-1}$
  $x = 0.3100 \times 10^{-3}, \overline{x} = 0.3000 \times 10^{-3} \Rightarrow EA = 0.1 \times 10^{-4}, ER = 0.3333 \times 10^{-1}$
  $x = 0.3100 \times 10^4, \overline{x} = 0.3000 \times 10^4 \Rightarrow EA = 0.1 \times 10^3, ER = 0.3333 \times 10^{-1}$
  Note como erro absoluto é enganoso!!!

Erro de Aproximação

Para compreender melhor o efeito da perda de dígitos significativos em números contendo uma parte inteira e uma fracionária na base decimal com n dígitos, vamos re-escrevê-los da seguinte forma:

$$x = f_x 10^e + g_x 10^{e-n}, \text{ dado que } 0.1 \leq f_x < 1 \text{ e } 0 \leq g_x < 1$$

Demonstração:
Seja $x = a_1 a_2 \ldots a_{k_I} . b_1 b_2 \ldots b_{k_F}$
Na representação em ponto flutuante, a_1 vem para logo após o ponto decimal:
$x = 0.a_1 a_2 a_3 \ldots$
Portanto andei k_I casas para a esquerda com meu ponto e preciso multiplicar por 10^{k_I}
Portanto $e = k_I$
Mas eu só posso representar n dígitos. Quantos ficam de fora?
R.: $k_I + k_F - n$

Erro de Aproximação

Qual a potência de 10 que multiplica o primeiro elemento a ficar de fora?
Lembre-se que o último elemento mais à direita (b_{k_F}) é multiplicado por 10^{-k_F}, o segundo da direita para a esquerda por 10^{-k_F+1} e assim sucessivamente
Portanto a potência do elemento cuja posição a partir da direita é $k_I + k_F - n$ só pode ser $10^{-k_F+k_I+k_F-n-1} = 10^{k_I-n-1} = 10^{e-n-1}$.
Como g_x é menor que 1, então a potência deve ser multiplicada por 10 para compensar ficando 10^{e-n}.c.q.d.

Erro de Aproximação

Como exemplo, a representação de $x = 6327.293$ para $n = 5$:

$$x = 0.63272 \times 10^4 + 0.93 \times 10^{-1}$$

Dada a limitação de dígitos, o termo $g_x 10^{e-n}$ não pode ser incorporado explicitamente. Neste caso, o truncamento descarta o termo $g_x 10^{e-n}$ e o arredondamento descarta o mesmo termo se $|g_x| < 0.5$ ou soma 10^{e-n} se $|g_x| \geq 0.5$ (no caso acima, $x = 0.63273 \times 10^4$, ou seja, soma-se 10^{-1})

Erro de Aproximação

Neste caso temos uma redefinição para erro absoluto (EA_x) e erro relativo (ER_x) para uma aproximação $\overline{x}$:

$$|EA_x| = |x - \overline{x}| \text{ e } |ER_x| = \frac{|EA_x|}{|\overline{x}|}$$

e valem os seguintes limitantes

$$\text{Truncamento} \begin{cases} |EA_x| < 10^{e-n} \\ |ER_x| < 10^{-n+1} \end{cases} \qquad \text{Arredondamento} \begin{cases} |EA_x| \leq 0.5 \times 10^{e-n} \\ |ER_x| < 0.5 \times 10^{-n+1} \end{cases}$$

Erro de Aproximação

Demonstração Truncamento:
$|EA_x| = |f_x 10^e + g_x 10^{e-n} - f_x 10^e| = |g_x 10^{e-n}| = |g_x| 10^{e-n}$. Como $|g_x| < 1$ temos $|EA_x| < 10^{e-n}$

$$|ER_x| = \frac{|EA_x|}{|\overline{x}|} = \frac{|g_x| 10^{e-n}}{|f_x| 10^e}$$

Como $|g_x|$ tem seu máximo em 1 e $|f_x|$ tem seu mínimo em 0.1, a fração acima é maximizada por $\frac{10^{e-n}}{0.1 \times 10^e} = \frac{10^{e-n}}{10^{e-1}} = 10^{-n+1}$

Logo: $|ER_x| < 10^{-n+1}$ (note que a desigualdade é estrita pois o valor 10^{-n+1} nunca á atingido de fato porque g_x é sempre estritamente menor que 1)

Erro de Aproximação

Demonstração Arredondamento:

$$\overline{x} = \begin{cases} f_x \times 10^e & \text{se } |g_x| < 0.5 \\ f_x \times 10^e + 10^{e-n} & \text{se } |g_x| \geq 0.5 \end{cases}$$

Portanto, se $|g_x| < 0.5$:

$$|EA_x| = |x - \overline{x}| = |g_x| \times 10^{e-n} < 0.5 \times 10^{e-n}$$

$$|ER_x| = \frac{|EA_x|}{|\overline{x}|} < \frac{0.5 \times 10^{e-n}}{|f_x| \times 10^e} < \frac{0.5 \times 10^{e-n}}{0.1 \times 10^e} = 5 \times 10^{-n} = 0.5 \times 10^{-n+1}$$

Se $|g_x| \geq 0.5$:

$$|EA_x| = |f_x \times 10^e + g_x \times 10^{e-n} - f_x \times 10^e - 10^{e-n}| = |g_x - 1| \times 10^{e-n} \leq 0.5 \times 10^{e-n}$$

Erro de Aproximação

Demonstração Arredondamento:

$$|ER_x| \leq \frac{0.5 \times 10^{e-n}}{|f_x| \times 10^e + 10^{e-n}}$$

Como sabemos que 10^{e-n} é positivo, ao ser somado no denominador ele produz um resultado menor na divisão. Assim:

$$|ER_x| < \frac{0.5 \times 10^{e-n}}{|f_x| \times 10^e} < \frac{0.5 \times 10^{e-n}}{0.1 \times 10^e} = 5 \times 10^{-n} = 0.5 \times 10^{-n+1}$$

Resumindo, considerando o maior erro possível em ambos os casos no arredondamento:

$$|EA_x| \leq 0.5 \times 10^{e-n} \qquad |ER_x| < 0.5 \times 10^{-n+1}$$

Erro de Aproximação

Os erros de uma operação podem ser estimados:

Aritmética de erros

$$
\begin{array}{ll}
EA_{x+y} = EA_x + EA_y & ER_{x+y} = ER_x \left(\frac{\overline{x}}{\overline{x}+\overline{y}}\right) + ER_y \left(\frac{\overline{y}}{\overline{x}+\overline{y}}\right) \\
EA_{x-y} = EA_x - EA_y & ER_{x-y} = ER_x \left(\frac{\overline{x}}{\overline{x}-\overline{y}}\right) - ER_y \left(\frac{\overline{y}}{\overline{x}-\overline{y}}\right) \\
EA_{xy} \approx \overline{x} EA_y + \overline{y} EA_x & ER_{xy} \approx ER_x + ER_y \\
EA_{x/y} \approx \frac{\overline{y} EA_x - \overline{x} EA_y}{\overline{y}^2} & ER_{x/y} \approx ER_x - ER_y
\end{array}
$$

Erro de Aproximação

Demonstração

Adição:

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

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

Subtração:

$$x - y = (\overline{x} + EA_x) - (\overline{y} + EA_y) = (\overline{x} - \overline{y}) + (EA_x - EA_y)$$

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

Multiplicação:

$$xy = (\overline{x} + EA_x)(\overline{y} + EA_y) = \overline{xy} + \overline{x}EA_y + \overline{y}EA_x + EA_xEA_y$$

Como se supõe EA_xEA_y pequeno: $EA_{xy} \approx \overline{x}EA_y + \overline{y}EA_x$

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

Erro de Aproximação

Divisão:

$\frac{x}{y} = \frac{\overline{x}+EA_x}{\overline{y}+EA_y}$

Colocando $\overline{y}$ em evidência no denominador:

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

Lembremos agora da série geométrica: $\frac{1}{1-x} = \sum_{n=0}^{\infty} x^n$ se $|x| < 1$

Para a série de $\frac{1}{1+x}$ basta trocar x por $-x$ e lembrar que agora potências ímpares mudam sinal:

$\frac{1}{1+x} = 1 - x + x^2 - x^3 + x^4 - x^5 + \ldots$ e portanto:

$\frac{1}{1+\frac{EA_y}{\overline{y}}} = 1 - \frac{EA_y}{\overline{y}} + \left(\frac{EA_y}{\overline{y}}\right)^2 - \left(\frac{EA_y}{\overline{y}}\right)^3 + \ldots$. Desprezando potências maiores que 1:

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

Como EA_xEA_y é muito pequeno:

$EA_{x/y} \approx \frac{EA_x}{\overline{y}} - \frac{\overline{x}EA_y}{\overline{y}^2} = \frac{\overline{y}EA_x - \overline{x}EA_y}{\overline{y}^2}$

Erro de Aproximação

Divisão:

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

Outline

Propagação de Erro em Operações Específicas

Problema do CANCELAMENTO: subtração implica em erro (relativo) grande se x e y forem próximos (Repare na fórmula para o erro relativo!!).
Exemplo de problema:
$x = \frac{5}{7}$ e $y = 0.714251$ em 5 dígitos truncados:
$fl(x) = 0.71428 \times 10^0$ e $fl(y) = 0.71425 \times 10^0$
$|(x - y) - (x \ominus y)| = |(x - y) - (fl(fl(x) - fl(y))| =$
$|\left(\frac{5}{7} - 0.714251\right) - (fl(0.71428 \times 10^0 - 0.71425 \times 10^0))| =$
$|0.347143 \times 10^{-4} - fl(0.00003 \times 10^0)| = 0.47143 \times 10^{-5}$
Erro absoluto pequeno mas erro relativo grande:
$|\frac{0.47143 \times 10^{-5}}{0.347143 \times 10^{-4}}| \leq 0.136$
Erro pode se propagar se continuarmos com outras operações!

Propagação de Erro em Operações Específicas

- Operações específicas podem aumentar o erro. Ex.: $\overline{z} = \overline{x} - \overline{y}$
- Re-arranjo de expressões pode atenuar o problema. Ex.: $f(x) = 1 - \cos(x)$ para $x \approx 0$

  Re-escrito como $f(x) = \frac{1-\cos^2(x)}{1+\cos(x)} = \frac{\sin^2(x)}{1+\cos(x)}$

Raiz da equação de 2º grau com sinal $+$:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

Quando $b^2 \gg 4ac$ temos problema com arredondamento. Solução:

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}\left(\frac{-b - \sqrt{b^2 - 4ac}}{-b - \sqrt{b^2 - 4ac}}\right) = \frac{b^2 - (b^2 - 4ac)}{2a(-b - \sqrt{b^2 - 4ac})} =$$

$$= \frac{-2c}{b + \sqrt{b^2 - 4ac}}.$$

Propagação de Erro em Operações Específicas

Identidades tipicamente usadas:

$x^2 - y^2 = (x + y)(x - y)$

$cos^2(x) + sin^2(x) = 1$

$\sqrt{x} - \sqrt{y} = \frac{x-y}{\sqrt{x}+\sqrt{y}}$ (supondo $x > 1$ e $y > 1$ tal que $|x - y| > |\sqrt{x} - \sqrt{y}|$)

$\cos^2(x) - \sin^2(x) = cos(2x)$ (quando $x \to (2n + 1)\frac{\pi}{4}$)

$\log(x) - \log(y) = \log(x/y)$

$\sin(x) - \sin(y) = 2\cos(\frac{x+y}{2})\sin(\frac{x-y}{2})$

$\cos(x) - \cos(y) = -2\sin(\frac{x+y}{2})\sin(\frac{x-y}{2})$

Propagação de Erro em Operações Específicas

Forma aninhada reduz o número de operações e atenua o erro:
$f(x) = x^3 - 6.1x^2 + 3.2x + 1.5$ em $x = 4.71$ com 3 dígitos
Exato: $f(4.71) = 104.487111 - 135.32301 + 15.072 + 1.5 = -14.263899$
Número de multiplicações: $2\ (x^3) + 1\ (x^2) + 1\ (6.1x^2) + 1\ (3.2x) = 5$
Número de adições/subtrações: 3

Truncamento: $f(4.71) = ((104. - 134.) + 15.0) + 1.5 = -13.5$
Arredondamento: $f(4.71) = ((105. - 135.) + 15.1) + 1.5 = -13.4$
Erros:
$|\frac{-14.263899+13.5}{-13.5}| \approx 0.0566$ para truncamento e $|\frac{-14.263899+13.4}{-13.4}| \approx 0.0645$ para arredondamento.

Propagação de Erro em Operações Específicas

Forma aninhada:
$f(x) = x^3 - 6.1x^2 + 3.2x + 1.5 = ((x - 6.1)x + 3.2)x + 1.5 \Rightarrow$
Número de multiplicações: 2
Número de adições/subtrações: 3

$f(4.71) = -14.2$ (truncamento) ou $f(4.71) = -14.3$ (arredondamento)e os erros agora são
$|\frac{-14.263899+14.2}{-14.2}| \approx 0.0045$ para truncamento e $|\frac{-14.263899+14.3}{-14.3}| \approx 0.0025$ para arredondamento.

Outline

Condicionamento e Instabilidade

- Conceitos de **condicionamento** e **instabilidade**
- Condicionamento é a sensibilidade da função $f(x)$ a mudanças no argumento x:

$$C = \max \left\{ \left| \frac{f(x) - f(\overline{x})}{f(x)} \right| / \left| \frac{x - \overline{x}}{x} \right| : |x - \overline{x}| \text{ pequeno} \right\}$$

Usando a notação costumeira para derivadas chamamos $\overline{x}$ de $x + h$
Se pensarmos na variação absoluta temos pelo teorema do valor médio para derivadas que:
$f(x + h) - f(x) = f'(\xi)h$ com $x < \xi < x + h$
Já pensando em desvio relativo como acima:
$C = \left| \frac{f(x+h)-f(x)}{f(x)} \right| / \left| \frac{h}{x} \right| \approx \left| \frac{hf'(x)}{f(x)} \right| / \left| \frac{h}{x} \right| = \left| \frac{f'(x)x}{f(x)} \right|$

Condicionamento e Instabilidade

- Ex.: $f(x) = \sqrt(x)$ é bem-condicionada:

$$\left|\frac{f'(x)x}{f(x)}\right| = \left|\frac{\frac{1}{2\sqrt{x}}x}{\sqrt{x}}\right| = \frac{1}{2}$$

- Já $f(x) = \frac{10}{1-x^2}$ é mal-condicionada devido ao problema com $|x| \approx 1$:

$$\left|\frac{f'(x)x}{f(x)}\right| = \left|\frac{10\frac{2x}{(1-x^2)^2}x}{\frac{10}{1-x^2}}\right| = \left|\frac{\frac{20x^2}{(1-x^2)^2}}{\frac{10}{(1-x^2)}}\right| = \frac{2x^2}{|1-x^2|}$$

Condicionamento e Instabilidade

- **Instabilidade:** Sensibilidade a erros de arredondamento
- Mais difícil de ser detectado
- Quebrar função em passos menores e verificar condicionamento de cada passo em relação ao condicionamento de f em si
- Ex.: $f(x) = \sqrt{x+1} - \sqrt{x}$ para $x \approx 10^4$:

$$\left| \frac{f'(x)x}{f(x)} \right| = \frac{1}{2} \frac{x}{\sqrt{x+1}\sqrt{x}} \approx \frac{1}{2}$$ MUITO BOM!!!

- Mas: para $x = 12345$ e 6 dígitos de precisão, $f(x) = 0.005$ enquanto o valor exato seria $f(x) = 0.0045000...$ (Erro relativo de 10%!)

Condicionamento e Instabilidade

- Quebrando em partes:

$$
\begin{aligned}
x_0 &\leftarrow 12345 \\
x_1 &\leftarrow x_0 + 1 \\
x_2 &\leftarrow \sqrt{x_1} \\
x_3 &\leftarrow \sqrt{x_0} \\
x_4 &\leftarrow x_2 - x_3
\end{aligned}
$$

A última função $f_3(t) = x_2 - t$ possui o seguinte condicionamento:

$$\left| \frac{f'(x)x}{f(x)} \right| = \left| \frac{t}{x_2 - t} \right|,$$

o qual é bem-condicionado exceto para $t \approx x_2$, o que ocorre aqui, pois $x_2 - t = 0.005$ e o condicionamento é 22222 (40 mil vezes maior que o condicionamento de f!!!).

Outline

Precisão de Máquina

- Como vimos anteriormente, o maior erro relativo possível no arredondamento é $\frac{1}{2}\beta^{-n+1}$. Este valor é chamado de ϵ da máquina
- Menor número positivo ϵ representado em ponto flutuante para o qual $1 + \epsilon > 1$

EX.:
$1 + 0.0_1 0_2 \ldots 0_{n-1} 5_n = 0.1_1 0_2 \ldots 0_{n-1} 0_n \times 10^1 + 0.0_1 0_2 \ldots 0_{n-1} 5_n \times 10^0$
IGUALANDO EXPOENTES:
$0.1_1 0_2 \ldots 0_{n-1} 0_n \times 10^1 + 0.0_1 0_2 \ldots 0_n 5_{n+1} \times 10^1$
MAS POR ARREDONDAMENTO:
$0.0_1 0_2 \ldots 0_n 5_{n+1} \times 10^1 = 0.0_1 0_2 \ldots 1_n \times 10^1$
SOMA: $0.1_1 0_2 \ldots 0_{n-1} 1_n \times 10^1 \neq 1$
A SOMA É DIFERENTE DE 1, o que não ocorreria se o primeiro dígito não-nulo de ϵ fosse menor que $\beta/2$ (5 no caso)

Precisão de Máquina

Algoritmo 5: Calcula precisão de máquina ϵ multiplicando por β^{-1} seguidamente, ou equivalentemente, dividindo por 2:

$s \leftarrow 1$;
$A \leftarrow 2$;
enquanto $s > 1$ **faça**
    $A \leftarrow A/2$;
    $s \leftarrow 1 + A$;
fim
retorna $\epsilon = 2A$;

Note que quando o algoritmo retorna do "loop" o valor de A é visto como 0 pela máquina. Por isso, retorna-se $2A$.