Ponto Flutuante

Paulo Ricardo Lisboa de Almeida

2021

1 Conteúdo da Aula

- Representação de Números reais em computadores
  - Overflow
  - Underflow

  Erros em operações aritméticas de Ponto Flutuante
  - Realização de operações de soma em ponto flutuante

2 Representação de Números Reais em Computadores

Computadores representam números reais através de *pontos flutuantes*, onde um número r_β é representado na forma:

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

onde:

β - base que a máquina opera (quase sempre base binária)

t - número de dígitos na mantissa, $0 \leq d_j \leq (\beta - 1), j = 1, ..., t, d_1 \neq 0$;

e - expoente no intervalo $[i, u]$

Considere o exemplo no sistema decimal

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

Os números são representados na forma:

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

Ao representar os seguintes números:

$x = 235.89 = 0.23589 \times 10^3$

O número possui 5 dígitos na mantissa, porém a arquitetura especificada suporta somente 3. Logo:

- O número pode ser representado como 0.235×10^3 se utilizarmos truncamento.
- O número pode ser representado como 0.236×10^3 se utilizarmos arredondamento

Truncamento e arredondamento de dígitos **são fontes de erro**.

Considere agora o seguinte exemplo, ainda para uma máquina onde $\beta = 10; t = 3; e \in [-5, 5]$:

$x = 0.00345 \times 10^{-5}$

Ao representar o valor em ponto flutuante, temos:

$x = 0.00345 \times 10^{-5} = 0.345 \times 10^{-7}$

Não podemos representar o valor na máquina de exemplo, onde $e \in [-5, 5]$ (note que o expoente -7 é menor que o menor expoente que a máquina comporta).

Quando isso acontece, temos um **underflow**.

Considere agora o seguinte valor:

$x = 875 \times 10^6$

Normalizando em ponto flutuante:

$x = 875 \times 10^6 = 0.875 \times 10^9$

O valor não pode ser representado na máquina, pois seu expoente é maior que 5.

Nesse caso, tivemos um **overflow**

Outro exemplo, agora com números binários:

Considere o número 3.5_{10}

Primeiro, convertendo para binário utilizando os métodos discutidos nas aulas passadas:

$3.5_{10} = 11.1_2$

Sabemos que $2^0 = 1$, então podemos escrever:

$11.1_2 = 11.1_2 \times 2^0$

Na base 10, ao multiplicar um número por 10^n, estamos "deslocando a vírgula" para a esquerda ou direita n vezes, dependendo se n é positivo ou negativo. O mesmo ocorre na base 2 ao se multiplicar por 2^n – ou em qualquer outra base β ao se multiplicar o valor por β^n. Sendo assim, normalizar o número binário em ponto flutuante é trivial:

$11.1_2 \times 2^0 = 1.11_2 \times 2^1 = 0.111_2 \times 2^2$

Considerando uma máquina $\beta = 2; t = 4; e \in [-3, 3]$, o valor armazenado será:

0.111×2^2

Considerando uma máquina $\beta = 2; t = 2; e \in [-3, 3]$, o valor armazenado será:

$0.111 \times 2^2 \approx 0.11 \times 2^2$ – precisamos truncar o valor (ou arredondar)

3 Erros Nas Operações Aritméticas de Ponto Flutuante

Considere A, a, B, b como valores exatos e aproximados, respectivamente, e E_a e E_b os erros dos operandos.

$A + B = (a + E_a) + (b + E_b) = a + b + E_a + E_b \qquad \therefore EA_{A+B} = E_a + E_b$

$A - B = (a + E_a) - (b + E_b) = a - b + E_a - E_b \qquad \therefore EA_{A-B} = E_a - E_b$

$A \times B = (a + E_a)(b + E_b) = ab + aE_b + bE_a + E_b * E_a \;\; \therefore EA_{A*B} = aE_b + bE_a + E_b \times E_a$

Devido aos erros na aritmética de ponto flutuante, a ordem que realizamos as operações pode mudar seus resultado.

Exemplo.

Suponto uma máquina $\beta = 10, t = 4, e \in [-4, 4]$, realizar os cálculos considerando $x_1 = 0,3491 \times 10^4$ e $x_2 = 0,2345 \times 10^0$.

$(x_2 + x_1) - x_1 =?$

$x_2 + (x_1 - x_1) =?$

Para realizar somas em ponto flutuante, devemos igualar o expoente do número de menor expoente com o de maior. Depois disso podemos somar as mantissas, mantendo os expoentes.

$x_2 = 0,2345 \times 10^0 = 0,00002345 \times 10^4$

Porém, a máquina possui $t = 4$, logo $0,00002345 \times 10^4$ será representado como $0,0000 \times 10^4$, considerando que o número foi truncado. Então:

$(x_2 + x_1) - x_1 = (0,2345 \times 10^0 + 0,3491 \times 10^4) - 0,3491 \times 10^4 \approx (0,0000 \times 10^4 + 0,3491 \times 10^4) - 0,3491 \times 10^4 = (0,3491 \times 10^4) - 0,3491 \times 10^4 = 0,0000 \times 10^0$

Fazendo agora $x_2 + (x_1 - x_1)$

$x_2 + (x_1 - x_1) = 0,2345 \times 10^0 + (0,3491 \times 10^4 - 0,3491 \times 10^4) = (0,2345 \times 10^0) + 0,0000 \times 10^0 = 0,2345 \times 10^0$

Dica

Multiplicações e divisões envolvem a multiplicação/divisão das mantissas, a soma/-subtração dos expoentes (da mesma forma que fazemos para valores em notação científica) e alguns passos extras de normalização que não serão vistos na disciplina introdutória de Cálculo Numérico. Você pode pesquisar esses detalhes em Patterson e Hennessy (2017).

Teste Você Mesmo

Volte na seção "Teste Você Mesmo" da aula inaugural, onde era pedido um programa que somava 0,1 10×. Coloque parêntesis em cada par de 0,1 sendo somado e note que o resultado muda. Isso se dá pelo mesmo motivo que acabamos de estudar.

---

curiosidade

A maioria das linguagens de programação utilizam a palavra chave *float* para identificar uma variável que pode receber um número fracionário. Isso por que a máquina utiliza o sistema de ponto flutuante (***float****ing-point*) discutido na aula de hoje.

O número de bits na mantissa e expoente das nossas máquinas é definido pelo padrão IEEE 754 (pesquise na internet ou em Patterson e Hennessy (2017)), e independe da

linguagem de programação que você está usando – está tudo embutido no hardware.

Por que temos variáveis que chamamos de *double*? O que significa *double*? Pesquise!

---

4 Exercícios

1) Considerando uma máquina $\beta = 10; t = 4; e \in [-3, 3]$, represente os números abaixo. Caso necessário, utilize o truncamento dos números, e caso não seja possível representá-los, indique um overflow ou underflow.

a) 2146

b) 0.00001

c) 890

d) 5.712389

2) Para uma máquina $\beta = 10, t = 4, e \in [-4, 4]$, realizar as operações. Qual a diferença (erro absoluto) do resultado nessa máquina pra uma máquina perfeita, capaz de representar qualquer número real ($\beta = 10; t = \infty; e \in [-\infty, +\infty]$) ? Considere que todos os valores apresentados estão na base 10, e os números são truncados caso não caibam na mantissa.

a) $0.5198 \times 10^3 + 0.0101 \times 10^1$

b) $0,437 \times 10^3 + 0.1021 \times 10^1$

5 Licença

Referências

PATTERSON, D.; HENNESSY, J. *Organização e Projeto de Computadores, 5ª Edição: Interface Hardware / Software*. [S.l.]: Elsevier Brasil, 2017.

RUGGIERO, M.; LOPES, V. da R. *Cálculo numérico: aspectos teóricos e computacionais*. [S.l.]: Makron Books do Brasil, 1996.

SANCHES, I.; FURLAN, D. C. C. *Métodos Numéricos*. [S.l.]: UFPR, 2007.