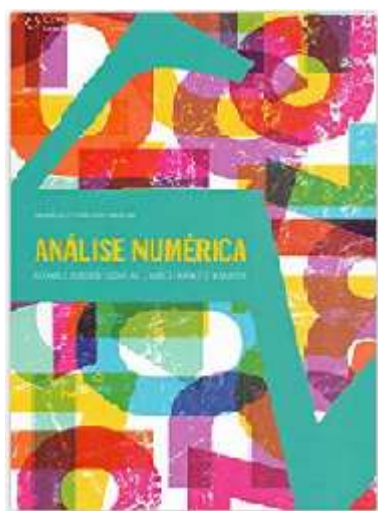


TEORIA: REPRESENTAÇÃO E ARITMÉTICA EM PONTO FLUTUANTE



Nossos **objetivos** nesta aula são:

- Conhecer a representação em ponto flutuante para implementação de algoritmos numéricos.
- Conhecer as bases da aritmética em ponto flutuante.
- Praticar com representação e aritmética de ponto flutuante.



Para esta semana, usamos como referência a **Seção 1.2 (Erros de Arredondamento e Aritmética no Computador)** do nosso livro da referência básica:

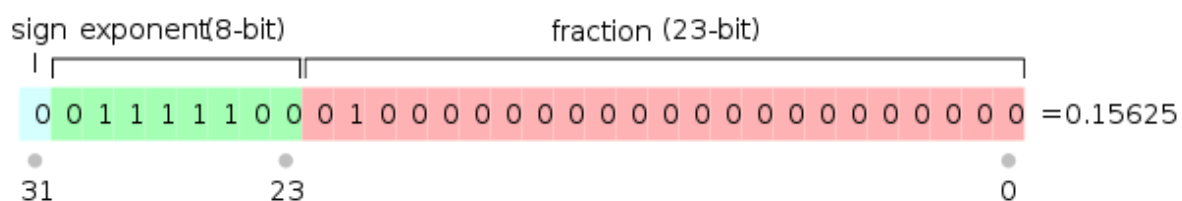
BURDEN, R.L., FAIRES, J.D. **Análise Numérica**. 10.ed. Norte Americana, São Paulo: Cengage Learning, 2017.

Não deixem de ler esta seção depois desta aula!

REPRESENTAÇÃO EM PONTO FLUTUANTE

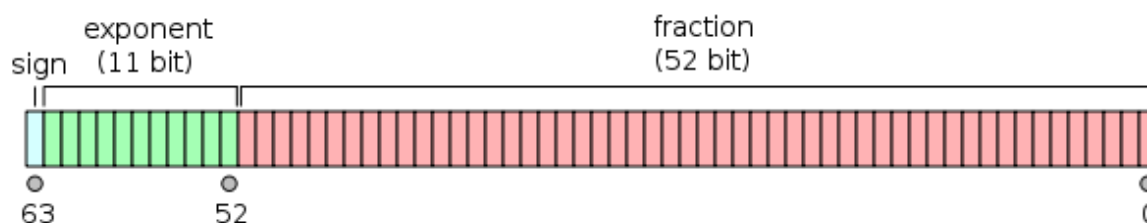
- Quando efetuamos cálculos em uma calculadora ou computador, os resultados podem ser diferentes daqueles que conhecemos em cursos tradicionais de álgebra ou cálculo. Por exemplo, sabemos de álgebra básica que $2+2=4$, $4*8=32$ e $(\sqrt{3})^2 = 3$.
- Na **aritmética computacional padrão**, esperamos resultados **exatos** para $2+2=4$ e $4*8=32$, mas não podemos garantir que $(\sqrt{3})^2 = 3$. Isto advém do fato de que, quando calculamos uma raiz quadrada na aritmética computacional padrão, temos **erros de arredondamento** devido à **representação e aritmética em ponto flutuante**.
- Espere **erros de arredondamento** sempre que forem feitos **cálculos usando números que não sejam potências de 2**. Manter estes **erros sob controle** é importante quando o número de cálculos for grande.

- Em 1985, o IEEE (*Institute of Electrical and Electronic Engineers*) publicou um relatório chamado **Binary Floating Point Arithmetic Standard 754-1985**, que especificava formatos para **precisão simples, dupla e estendida**. Isto é conhecido como **Padrão IEEE 754 (ou Padrão IEEE para Ponto Flutuante)** e é seguido por fabricantes de hardware e desenvolvedores do software que manipulam ponto flutuante.
- Comumente, utilizamos dois tipos deste padrão: **float (ponto flutuante de precisão simples)** e **double (ponto flutuante de precisão dupla)**.
- Na representação do tipo float, utilizamos **32 bits** conforme mostrado na representação do número real 0.15625:



Para o tipo **float**, o primeiro bit é um indicador de **sinal (sign)**, os próximos **8 bits** são chamados de **expoente (E)** ou **característica** e, os **23 bits** restantes, de **fração** ou **mantissa (M)**. A mantissa é normalizada na forma $0, d_1 d_2 \dots d_{23}$.

- Na representação do tipo double, utilizamos **64 bits** conforme mostrado no esquema a seguir:



Para o tipo **double**, o primeiro bit também é um indicador de **sinal (sign)**, os próximos **11 bits** são o **expoente** ou **característica** e, os **52 bits** restantes, a **fração** ou **mantissa**. A **mantissa é normalizada** na forma $0, d_1 d_2 \dots d_{52}$.

- Para o tipo **double**, em particular, pode-se **recuperar o número representado** pela seguinte forma:

$$(-1)^{\text{sinal}} 2^{\text{expoente}-1023} (1 + f)$$

Uma fórmula similar pode ser obtida para quantidades em diferentes tipos de formato (lista de exercícios, exercício 1 extra-classe).

- ```
0 10000000011 101110010001000000000000000000000000000000000000
```

- [illegible]

## TRUNCAMENTO/ARREDONDAMENTO EM PONTO FLUTUANTE

---

- Qualquer **número em ponto flutuante** pode ser **normalizado (na base decimal)** da seguinte forma:

$$\pm 0, d_1 d_2 \dots d_k d_{k+1} d_{k+2} \dots \times 10^n, \text{ com } 1 \leq d_1 \leq 9 \text{ e } 0 \leq d_i \leq 9, \text{ para } 2 \leq i \leq k$$

- Porém, devido ao tamanho dos registradores dos processadores, sabemos da **impossibilidade** de se **representar infinitas casas decimais** de um número. Assim, podemos utilizar duas operações possíveis para uma representação finita do número:

- **Truncamento:**  $\pm 0, d_1 d_2 \dots d_k \times 10^n$ , com  $1 \leq d_1 \leq 9$  e  $0 \leq d_i \leq 9$ , para  $2 \leq i \leq k$
- **Arredondamento:** se  $d_{k+1} \geq 5$ , somamos 1 em  $d_k$ ; se  $d_{k+1} < 5$ , mantemos o valor de  $d_k$

## EXERCÍCIO TUTORIADO

---

- 3) Sabemos que o número  $\pi$  tem uma expansão infinita da forma  $\pi = 3,1415926535 \dots$ . Calcule a representação normalizada com 5 algarismos para as operações de truncamento e arredondamento.

## ERROS EM TRUNCAMENTO/ARREDONDAMENTO EM PONTO FLUTUANTE

---

- Tanto as **operações de truncamento** quanto **arredondamento** em **operações numéricas em ponto flutuante** podem produzir **erros**.
- Denotando por  $fl(x)$  e  $fl(y)$  as representações em **ponto flutuante de  $x$  e  $y$** , as **operações em ponto flutuante** são definidas pelas seguintes igualdades:
  - soma:  $x \oplus y = fl(fl(x) + fl(y))$
  - subtração:  $x \ominus y = fl(fl(x) - fl(y))$
  - multiplicação:  $x \otimes y = fl(fl(x) \times fl(y))$
  - divisão:  $x \oslash y = fl(fl(x)/fl(y))$
- Se  $p$  é o **valor exato** de um número e  $p^*$  uma **aproximação** de  $p$ , definimos o **erro absoluto** da aproximação por  $|p - p^*|$  e o **erro relativo** da aproximação por  $\frac{|p - p^*|}{|p|}$  (com  $|p| \neq 0$ ).

## EXERCÍCIO TUTORIADO

---

4) Seja  $p = 0,54617$  e  $q = 0,54601$ . Calcule os erros absoluto e relativo produzidos na operação de **subtração** com **aritmética de quatro algarismos**, com os seguintes esquemas de aproximação:

(a) truncamento

(b) arredondamento

### EXERCÍCIO TUTORIADO EXTRA

---

- 1) Supor que  $x = \frac{5}{7}$  e  $y = \frac{1}{3}$ . Use truncamento com 5 algarismos para calcular  $x \oplus y$ ,  $x \ominus y$ ,  $x \otimes y$ ,  $x \oslash y$ .

## EXERCÍCIOS EXTRA-CLASSE

---

1. Em aula, obtivemos uma fórmula para recuperar a representação decimal de um número em ponto flutuante de precisão dupla (double) com a fórmula abaixo:

$$(-1)^{\text{ sinal }} 2^{\text{ expoente } - 1023} (1 + f)$$

2. Usando a aritmética de arredondamento, com **três algarismos**, calcule os erros absoluto e relativo com o **valor exato determinado com cinco algarismos** para os cálculos abaixo:

a.  $133 + 0,921$

b.  $133 - 0,499$

c.  $(121 - 0,327) - 119$

d.  $\left(\frac{2}{9}\right) \cdot \left(\frac{9}{7}\right)$

e.  $\frac{\pi - \frac{22}{7}}{\frac{1}{7}}$