Cálculo Numérico

Um guia prático com Python

Prof. Dr. Rogério Vargas[1]

[1]Centro de Estudos do Mar
Universidade Federal do Paraná

2026

rogerio.in

UFPR

Encontro 4: Aritmética de Ponto Flutuante

Objetivos

Ao final da aula, o estudante deverá ser capaz de:

- converter números entre as bases decimal e binária;
- utilizar notação científica;
- identificar sinal, expoente e mantissa;
- aplicar truncamento e arredondamento;
- compreender por que surgem erros numéricos;
- observar esses erros em Python.

Os computadores erram?

Ideia central

O computador possui memória finita e, por isso, não consegue representar exatamente todos os números reais.

No sistema decimal:

$$\frac{1}{3} = 0{,}333333\ldots$$

Com apenas quatro casas:

$$\frac{1}{3} \approx 0{,}3333$$

No sistema binário:

$$0{,}1_{10} = 0{,}000110011\ldots_2$$

O computador armazena uma aproximação.

Conclusão

O computador calcula corretamente com os valores que conseguiu armazenar, mas esses valores podem já ser aproximações.

Um experimento em Python

Soma

```
>>> 0.1 + 0.2
0.30000000000000004
```

Comparação

```
>>> 0.1 + 0.2 == 0.3
False
```

- $0{,}1$, $0{,}2$ e $0{,}3$ não possuem representação binária finita.
- O erro é muito pequeno, mas pode se acumular.
- Alguns métodos numéricos podem amplificar esses erros.

Comparação com tolerância

Em cálculos numéricos, frequentemente verificamos

$$|x - y| < \varepsilon$$

em vez de comparar diretamente $x = y$.

De onde vêm os erros numéricos?

Erro nos dados
Medições e dados fornecidos possuem precisão limitada.

Erro de modelagem
O modelo matemático simplifica a realidade.

Erro de truncamento
Um processo infinito é substituído por um número finito de etapas.

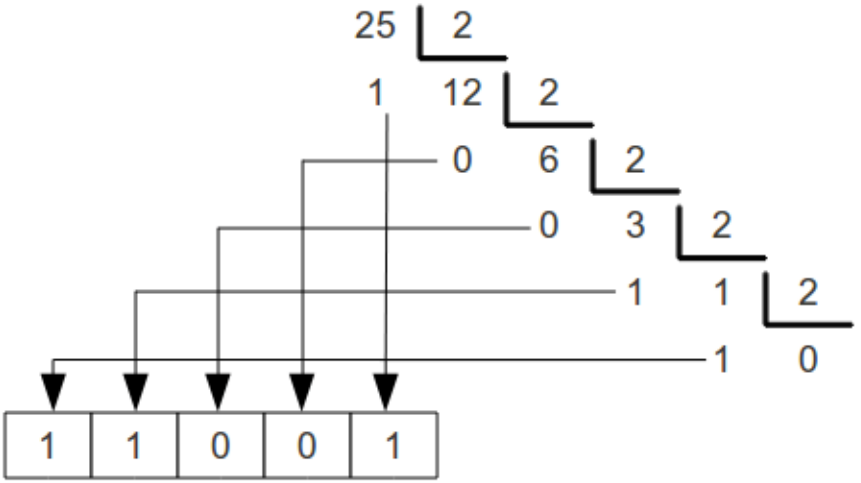
Erro de arredondamento
O computador armazena uma quantidade limitada de dígitos.

Caminho da aula

Para compreender os erros de representação, seguiremos esta ordem:

1. conversão entre as bases decimal e binária;
2. notação científica;
3. representação com sinal, expoente e mantissa;
4. truncamento e arredondamento;
5. underflow e overflow;
6. experimentos em Python.

Conversão de decimal para binário



Conversão de binário para decimal

$$2^6 \times 1 + 2^5 \times 0 + 2^4 \times 1 + 2^3 \times 1 + 2^2 \times 0 + 2^1 \times 1 + 2^0 \times 1 = 91$$

Exercícios: conversão de bases

Converta cada número para a base indicada:

Para binário:

1. 2_{10}
2. 123_{10}
3. 1111_{10}
4. $27{,}75_{10}$

Para decimal:

1. 10110_2
2. 1111_2
3. $1100{,}01_2$

Notação científica

Definição

Um número em notação científica é escrito como

$$x = a \cdot 10^n,$$

em que $1 \leq |a| < 10$ e n é um número inteiro.

- a: mantissa ou significando;
- 10: base;
- n: expoente.

Exemplo

$$125000 = 1{,}25 \cdot 10^5$$

Exercícios: notação científica

Converta para decimal:

1. $1{,}2 \cdot 10^{5}$
2. $9{,}2 \cdot 10^{-5}$
3. $-7{,}2123 \cdot 10^{7}$

Converta para notação científica:

1. 1250000
2. 500000000000
3. $0{,}0000256$
4. $-0{,}0000003$
5. $0{,}0100003$

Representação em ponto flutuante

A ideia da notação científica também é usada na base binária:

$$x = (-1)^s \cdot m \cdot 2^e$$

- s: sinal do número;
- m: mantissa ou significando;
- e: expoente;
- 2: base utilizada pelo computador.

Exemplo

$$101{,}1_2 = 1{,}011_2 \cdot 2^2$$

Assim, a mantissa é $1{,}011_2$ e o expoente é 2.

Um modelo didático com 8 bits

- **Sinal:** indica se o número é positivo ou negativo.
- **Expoente:** indica a posição da vírgula.
- **Mantissa:** guarda os algarismos mais importantes.

Modelo didático
Este formato simplificado ajuda a compreender o princípio, mas não representa todos os detalhes do padrão IEEE 754.

Por que a mantissa produz aproximações?

O número $0{,}1$ possui representação binária infinita:

$$0{,}1_{10} = 0{,}0001100110011\ldots_2$$

Normalizando:

$$0{,}1_{10} = 1{,}100110011\ldots_2 \cdot 2^{-4}$$

Com apenas três bits para a parte fracionária da mantissa:

$$1,\ \underbrace{100}_{\text{armazenados}}\ \underbrace{110011\ldots}_{\text{descartados}} \cdot 2^{-4}$$

Consequência

O valor armazenado fica muito próximo de $0{,}1$, mas não é exatamente $0{,}1$.

Truncamento

Definição

No truncamento, conservamos a quantidade desejada de algarismos significativos e descartamos os demais.

Exemplo com três algarismos significativos

$$3{,}141592\ldots \longrightarrow 3{,}14$$

Os algarismos após o terceiro significativo são simplesmente retirados.

Arredondamento

Definição

Conservamos a quantidade desejada de algarismos significativos e observamos o primeiro algarismo descartado.

- de 0 a 4: o último algarismo mantido não muda;
- de 5 a 9: acrescentamos uma unidade ao último algarismo mantido.

Exemplo com três algarismos significativos

$$3{,}141592\ldots \longrightarrow 3{,}14$$

$$3{,}146592\ldots \longrightarrow 3{,}15$$

Exercício: truncamento e arredondamento

Represente os números abaixo com dois algarismos significativos, utilizando truncamento e arredondamento:

1. $x_1 = 0{,}437$
2. $x_2 = 0{,}144$
3. $x_3 = -0{,}495$
4. $x_4 = 0{,}314159265 \cdot 10^1$

Underflow

Definição

Ocorre quando a magnitude do resultado é menor do que o menor número positivo normal representável:

$$0 < |x| < \text{fmin}_N \, .$$

O resultado pode ser representado como um número subnormal ou ser aproximado para zero.

Exemplo em Python

```
>>> 1e-300 * 1e-300
0.0
```

Overflow

Definição

Ocorre quando a magnitude do resultado é maior do que o maior número finito representável:

$$|x| > \mathrm{fmax}\,.$$

Dependendo da operação e do sistema utilizado, o resultado pode ser representado como infinito ou provocar um erro.

Exemplo em Python

```
>> 1e308 * 10
inf
```

Atividade prática em Python

Vamos desenvolver dois programas:

1. receber um número decimal e apresentá-lo em notação científica;
2. receber um número real e uma quantidade de algarismos significativos, apresentando:
   - o resultado por truncamento;
   - o resultado por arredondamento.

Fechamento

- A memória do computador é finita.
- Nem todo número decimal possui representação binária finita.
- A mantissa limita a quantidade de algarismos significativos.
- Truncamento e arredondamento introduzem pequenas diferenças.
- Os métodos numéricos precisam considerar e controlar esses erros.

Mensagem principal

O computador não trabalha com todos os números reais, mas com um conjunto finito de números representáveis.