

# Métodos Numéricos

Prof. Dr. Jonatha Costa

2024

# Organização

---

## ① Ponto flutuante, armazenamento de dados e erros

Conceitos

Representação EE 754/2008

Erros em soluções Numéricas

Desastres

# MN aplicados à Engenharia: Objetivo da aula

---

- Apresentar conteúdo de :
  - Aritmética de ponto flutuante
  - Armazenamento de dados no computador
  - Erros de arredondamento e de truncamento

# MN aplicados à Engenharia: Aritmética de Ponto flutuante

---

Sejam as grandezas:

- Massa do Elétron:  $9 \times 10^{-28}$  *gramas*
- Massa do Sol:  $2 \times 10^{33}$  *gramas*
- Faixa de variação:  $> 10^{60}$
- Representação de grandezas extremas
  - 000000000.1324468585133
  - 13676341235445403.341464684654

Como representar tais números de modo equânime?

# MN aplicados à Engenharia: Aritmética de Ponto flutuante

---

- Solução: notação científica:

$$\textit{Algarismo} \times \textit{Base}^{\textit{expoente}}$$

- Sistema de representação de maneira que a faixa de variação dos números seja independente do número de dígitos significativos dos números representados.
- Ponto flutuante

# MN aplicados à Engenharia: Aritmética de Ponto flutuante

## Representação em Ponto flutuante - base decimal

Para acomodar números grandes e pequenos, números reais são escritos na representação em ponto flutuante (também chamada de notação científica) e cuja a forma:

$$d, ddddd \times 10^p$$

Nessa representação, um algarismo é escrito à esquerda da vírgula decimal, e o resto dos algarismos significativos é escrito à direita da vírgula. O número 0, ddddddd é chamado de **mantissa**.

- 6519,23 é escrito como  $6,51923 \times 10^3$
- 0,00000391 é escrito como  $3,91 \times 10^{-6}$

A potência de  $10^p$ , representa a ordem de grandeza do número(O), a qual é estruturada como  $(p + 1)$  se a ordem for menor que 5.

### Exemplo:

- $3,91 \times 10^{-6}$  é da ordem de  $10^{-6}.O(10^{-6})$
- $6,51923 \times 10^3$  é da ordem de  $10^4.O(10^4)$ .

# MN aplicados à Engenharia: Aritmética de Ponto flutuante

## Representação em Ponto flutuante - base binária

A representação binária em ponto flutuante tem a forma:

$$1, bbbbb \times 2^{bbb}$$

Nessa forma, a mantissa é 0, bbbbbb, e a potência de 2 é chamada de expoente. Tanto a mantissa quanto o expoente são escritos na forma binária.

Para representar um número decimal em Ponto flutuante - base binária deve-se proceder a normalização do número e depois a conversão do número para a representação binária.

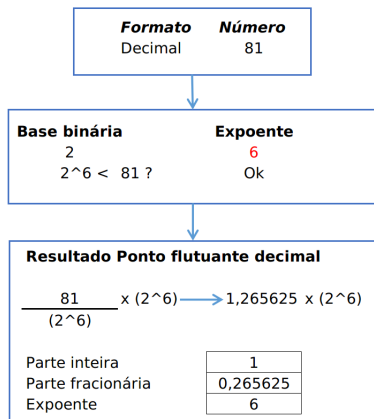
Um número  $n$ , portanto, deve ser dividido e multiplicado pela *maior potência de 2*, imediatamente menor que o próprio número  $n$ . Depois tanto a mantissa quanto o expoente são escritos na forma binária, posto que a parte inteira é 1.

**Exemplo:** Como é escrito  $50_{10}$  em binário? E na representação de ponto flutuante binária?

- ① Normalizar -  $50_{10} = \frac{50}{2^5} \times 2^5 = 1,5625 \times 2^5$
- ② Converter mantissa -  $0,5625_{10} = 1001_2$
- ③ Converter expoente -  $5_{10} = 101_2$
- ④ Compor o número -  $1,1001 \times 2^{101}$

# MN aplicados à Engenharia: Aritmética de Ponto flutuante

## Representação em Ponto flutuante - exemplo



### Ponto flutuante decimal para binário

$$1,265625 \times (2^6) \longrightarrow ?$$

#### Conversão para binário - parte inteira

$$1 = 1$$

#### Conversão para binário - mantissa

010001

0,265625	x 2 =	0,53125	0
0,53125	x 2 =	1,0625	1
0,0625	x 2 =	0,125	0
0,125	x 2 =	0,25	0
0,25	x 2 =	0,5	0
0,5	x 2 =	1	1
0	x 2 =	0	



#### Conversão para binário - expoente

$$6 \quad 110$$

### Resultado Ponto flutuante binário

$$1,265625 \times (2^6) \longrightarrow 1,010001 \times (2^{110})$$



# MN aplicados à Engenharia: Aritmética de Ponto flutuante

---

## Representação em Ponto flutuante - exemplo

### Método alternativo

- 1 Converter o número decimal para binário  
 $81_{10} = 1010001_2$
- 2 Deslocar a vírgula para direita e registrar o número de deslocamentos  
 $1,010001$  e 6 deslocamentos.
- 3 Converter a quantidade de deslocamentos para binário  
 $6_{10} = 110_2$
- 4 Reescrever o número unindo os termos acima  
 $1,010001 \times 2^{110}$

# MN aplicados à Engenharia: Aritmética de Ponto flutuante

## Representação em Ponto flutuante

Distribuição dos bits na representação IEEE 754/2008

Tipo	Sinal	Expoente	Mantissa
<b>Simplex (32 bits)</b>	bit 31	bits 30-23	bits 22-0
<b>Dupla (64 bits)</b>	bit 63	bits 62-52	bits 51-0

- Precisão simples 32 bits (dupla 64 bits)
- 1 bit para o sinal de número em ambas as precisões
- 8 bits para o expoente (11 bits para o expoente) + polarização (127 simples e 1023 dupla)
- 23 bits para a mantissa (52 bits para a mantissa)

A polarização é introduzida para se evitar o uso de um dos bits para representar o sinal do expoente (uma vez que o expoente pode ser positivo ou negativo).

### Resumo:

Diagram illustrating the IEEE 754 single-precision floating-point format. The 32-bit word is divided into three fields:

- Sign:** 1 bit.
- Exponent + bias:** 11 bits.
- Mantissa:** 52 bits.

The diagram shows the bit positions for the exponent (from  $2^{10}$  to  $2^{-52}$ ) and the mantissa (from  $2^{-50}$  to  $2^{-52}$ ). Red arrows indicate the bit positions for the Sign, Exponent, and Mantissa fields.

Fonte: GILAT,(2008)

1 bit para sinal, 11 bits para o expoente e 52 bits para a mantissa, bias = 1023

1 bit para sinal, 8 bits para o expoente e 23 bits para a mantissa, bias = 127

# MN aplicados à Engenharia: Aritmética de Ponto flutuante

Qual a representação do número 81 decimal na norma IEEE 754/2008 precisão de 32 e 64 bits?

- $81_{10}$  em ponto flutuante *binário* resulta em  $1,010001 \times (2^{110})$ , conforme já visto.
- Precisão simples (32 bits)
  - Bit 31 (sinal) = positivo 0;
  - Bits 23 ao 30 (expoente com bias de 127) =  $6_{10} + 127_{10} = 133_{10} = 10000101_2$
  - Bits 00 ao 22 (mantissa) =  $010001_2$
  - Logo: **0 | 10000101 | 0100010000000...0000 - 32bits**
- Precisão dupla (64 bits)
  - Bit 63 (sinal) = positivo 0;
  - Bits 52 ao 62 (expoente com bias de 1023) =  $6_{10} + 1023_{10} = 1029_{10} = 10000000101_2$
  - Bits 00 ao 51 (mantissa) =  $010001_2$
  - Logo: **0 | 10000000101 | 01000100000...0000 - 64bits**
- Nota:  
Lembre-se de que a sintaxe é : *sinal + expoente + mantissa!*

# MN aplicados à Engenharia: Aritmética de Ponto flutuante

## Representação em Ponto flutuante

### Considerações

- Mais bits para a mantissa fornece mais precisão ( $t$ )
- Mais bits para o expoente, aumenta o range de valores ( $e$ ).
- O tamanho da palavra do computador depende de características internas à arquitetura do mesmo. Em geral, os microcontroladores tem tamanho de palavra de 16 bits, os microcomputadores padrão PC tem tamanho de palavra de 32 bits, 64 bits ou mais.

Figura: Encapsulamentos de processadores



Fonte: Autor desconhecido

# MN aplicados à Engenharia: Erros em soluções numéricas

---

## Erros em soluções numéricas

- Erros de arredondamento
- Erros de truncamento
- Erro total

# MN aplicados à Engenharia: Erros em soluções numéricas

## Erros de arredondamento

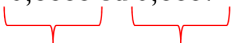
- Os números são representados em computador através de um número finito de bits.
- Os números podem ser encurtados e descartados ou arredondados.

### Exemplo:

Figura: Erros de arredondamento

$$2/3 = 0,666666\dots$$

$$2/3 = 0,6666 \text{ ou } 0,6667$$

  
Encurtado / arredondado

Fonte: DIAS,(2019)

# MN aplicados à Engenharia: Erros em soluções numéricas

---

## Erros de arredondamento:

**Exemplo:** Efetue a subtração entre dois números reais  $p = 9890,9$  e  $q = 9887,1$  mantendo o formato original, encurtando e arredondando os valores para três algarismos significativos.

- $p - q = 9890,9 - 9887,1 = 3,8$  (formato original)
- Se apenas três algarismos são permitidos na mantissa, tem-se a **representação em ponto flutuante**:
  - $p - q = 9,890 \times 10^3 - 9,887 \times 10^3 = 0,003 \times 10^3 = 3$  (corte)
  - $p - q = 9,891 \times 10^3 - 9,887 \times 10^3 = 0,004 \times 10^3 = 4$  (arredondamento)

A diferença verdadeira (exata) entre os números é de 3,8. Esses resultados mostram que, no presente problema, o arredondamento leva a um valor mais próximo à resposta verdadeira.



# MN aplicados à Engenharia: Erros em soluções numéricas

## Exemplo

- Considere a equação quadrática  $x^2 - 100,0001x + 0,01 = 0$ , para a qual as soluções exatas são  $x_1 = 100$  e  $x_2 = 0,0001$ .
- As soluções podem ser calculadas com a fórmula quadrática  $x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$  e  $x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$ , o que, em ambiente de simulador computacional Matlab/Octave resultando em:  
 $x_1 = 99.9999000000000000$  e  $x_2 = 1.000000000033197e - 004$ .
- O valor calculado no simulador para  $x_2$  não é exato devido a erros de arredondamento. Tais erros ocorrem no numerador da expressão de  $x_2$ . Como  $b$  é negativo, o numerador envolve a subtração de dois números que são quase iguais.
- Em alguns casos, a forma das expressões matemáticas pode ser mudada para uma forma diferente, menos propensa a erros de arredondamento como:  

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \left( \frac{-b + \sqrt{b^2 - 4ac}}{-b + \sqrt{b^2 - 4ac}} \right) = \frac{2c}{-b + \sqrt{b^2 - 4ac}}.$$
- O que resulta e  $x_2 = 1.0000000000000000e - 004$ .

# MN aplicados à Engenharia: Erros em soluções numéricas

---

## Erros de truncamento:

- Ocorrem quando os métodos numéricos usados na solução de um problema utilizam uma aproximação.
- **Exemplo:** A função  $\text{sen}(x)$  pode ser aproximada pela série de Taylor.

$$\text{sen}(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \frac{x^{11}}{11!} + \dots$$

- O valor exato do  $\text{sen}(\pi/6) = 0,5$
- Usando **um** termo da série de Taylor:  $\text{sen}(\pi/6) = (\pi/6) = 0,5235988$
- Usando **dois** termos  $\text{sen}(\pi/6) = \frac{\pi}{6} - \frac{(\pi/6)^3}{3!} = 0,4996742$

# MN aplicados à Engenharia: Erros em soluções numéricas

## Erro total (Erro real)

- A solução numérica é uma aproximação sempre inclui erros de arredondamento e, dependendo do método numérico utilizado, também pode incluir erros de truncamento.
- Juntos, os erros de arredondamento e de truncamento resultam no erro numérico total incluído na solução numérica.
- **Erro total** - ou erro real, é a diferença entre a solução verdadeira (exata) e a solução numérica sendo expresso por :

$$ErroReal = SolucaoExata - SolucaoNumerica$$

- **Erro relativo** - valor absoluto da razão entre o erro real e a solução exata sendo expresso por:

$$Erro\ Relativo\ Real = \frac{Erro\ Real}{Solucao\ Exata}$$

# MN aplicados à Engenharia: Erros em soluções numéricas

## Desastres causados por erros numéricos:

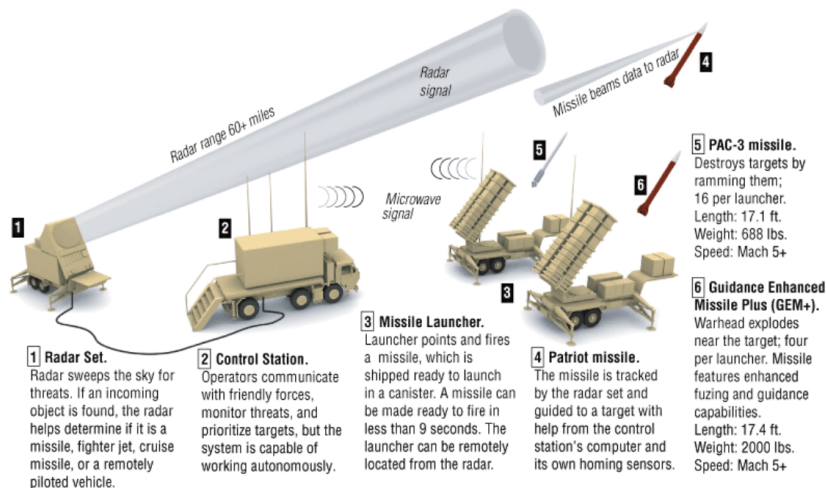
Figura: Defesa antiaérea - Patriot



Fonte: DIAS,(2019)

# MN aplicados à Engenharia

Figura: Defesa antiaérea - Patriot



Fonte: DIAS,(2019)

# MN aplicados à Engenharia: Erros em soluções numéricas

## Desastres causados por erros numéricos:

Figura: Míssel Iraniano



Fonte:DIAS,(2019)

# MN aplicados à Engenharia: Erros em soluções numéricas

---

## Desastres causados por erros numéricos:

O que aconteceu?

- Dia 25 de fevereiro de 1991 – Guerra do Golfo;
- Arábia Saudita – Sistema Patriot falhou na interceptação de mísseis SCUD;  
– Morte de 28 soldados americanos;
- Para prever onde mísseis estarão, ambos tempo e velocidade precisam ser números reais;
- O tempo em dezenas de segundos era multiplicado por  $1/10$  para produzir o tempo em segundos;

## MN aplicados à Engenharia: Erros em soluções numéricas

## Desastres causados por erros numéricos:

- 1/10 quando convertido para base binária resulta em uma dízima com período igual a 104 bits;
  - Patriot possuía um registrador de 24 bits;
- Conversão binária de 1/10  
0.000110011001100110011001100**11001100**...
  - O registrador do Patriot gravou  
0.000110011001100110011001100
  - Erro gerado  
0.00000000000000000000000000**11001100**...



# MN aplicados à Engenharia: Erros em soluções numéricas

---

## Desastres causados por erros numéricos

- Tempo depois de 100 horas  
 $0.000000095 \times 100 \times 60 \times 60 \times 10 = 0,34$  segundos
- Velocidade do Scud =  $1.676m/s$   
 Distância percorrida em 0,34 segundos =  $0,5km$
- Quem foi o responsável?  
 O projetista? O exército americano? Operadores?

# MN aplicados à Engenharia: Erros em soluções numéricas

## Desastres causados por erros numéricos

Figura: Ariane 5



### Explosão do foguete francês Ariane 5.

- Em 4 de junho de 1996, menos de um minuto após o lançamento, o foguete francês Ariane 501 se autodestruuiu.
- Foi indicada pelo CNES (Centro Nacional de Estudos Espaciais) e pela ESA (Agência Espacial Européia) uma comissão para investigar a ocorrência.
- Comissão indica um erro no software de controle como a origem na falha do lançamento.

Fonte: DIAS,(2019)

# MN aplicados à Engenharia: Erros em soluções numéricas

## Desastres causados por erros numéricos

Figura: Ariane 5



A investigação preliminar dos dados de voo revelou:

- Comportamento do lançador em condições nominais, até 36 segundos após a decolagem.
- Falha do sistema de referência inercial (SRI) secundário, seguido imediatamente de falha do sistema de referência inercial em operação.
- Guinada dos bocais dos sistemas de propulsão até o seu ângulo máximo, causando uma guinada abrupta do veículo.
- Autodestruição do lançador, disparada corretamente como consequência da ruptura das juntas entre os sistemas de propulsão sólidos e o primeiro estágio.

Fonte: DIAS,(2019)

# MN aplicados à Engenharia: Erros em soluções numéricas

## Desastres causados por erros numéricos

Figura: Ariane 5



A investigação preliminar dos dados de voo revelou:

- A origem da falha foi confinada ao sistema de controle, mais especificamente, aos dois sistemas referenciais inerciais (SRI), que nitidamente deixaram de funcionar, por volta de 36,7s após a decolagem.
- A anomalia interna de *software* do SRI ocorreu durante a execução de uma conversão de dados de um número de 64 bits em ponto flutuante para um inteiro de 16 bits com sinal. O valor do número em ponto flutuante era maior do que poderia ser representado pelo inteiro de 16 bits com sinal. O resultado foi um operando inválido.

Fonte: DIAS,(2019)

# Exercícios

---

- Veja a lista de exercícios na web
- Veja a lista de códigos em: <https://github.com/jonathacosta/NM>