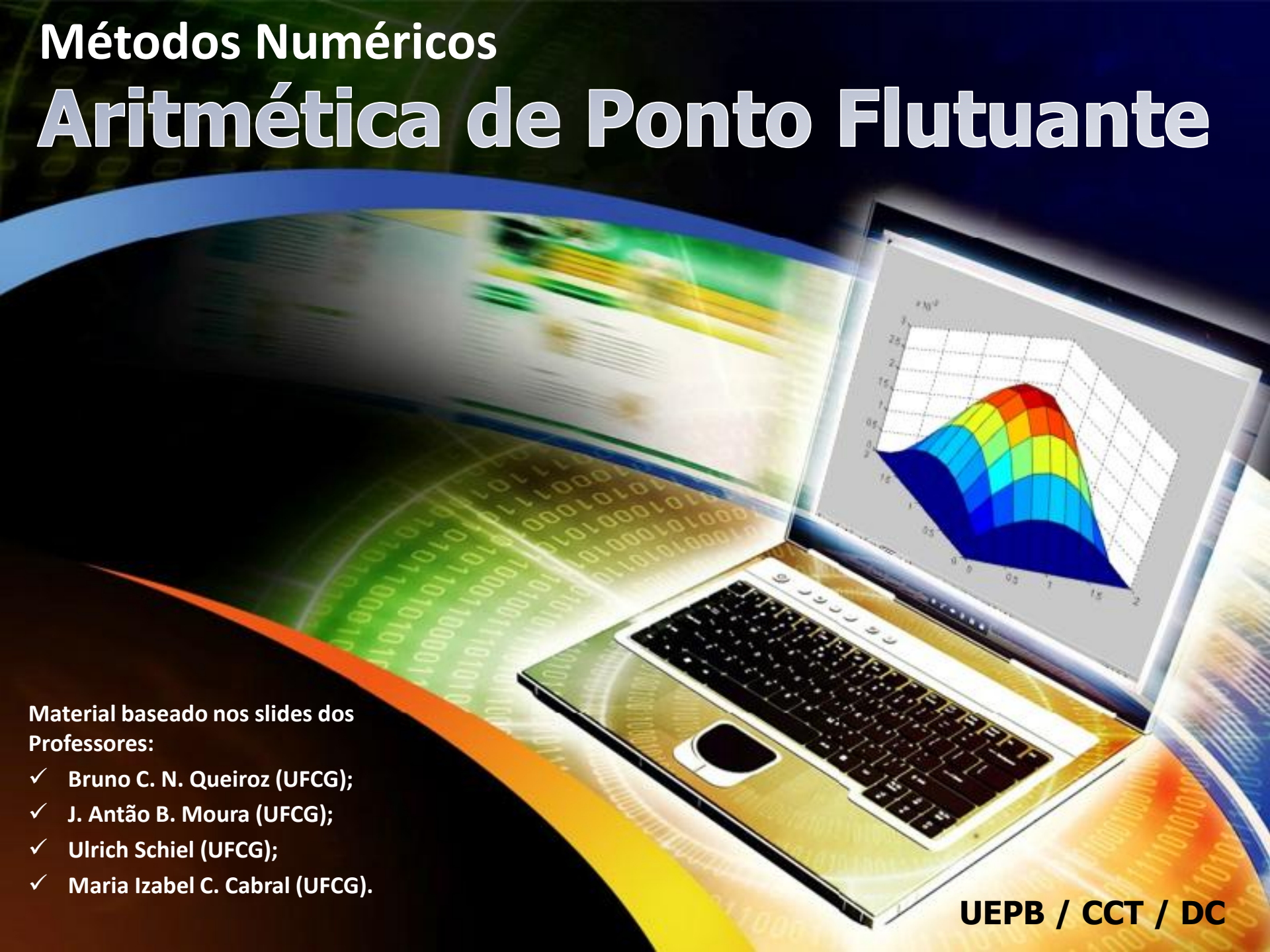
Métodos Numéricos

Aritmética de Ponto Flutuante



Material baseado nos slides dos Professores:

- ✓ **Bruno C. N. Queiroz (UFCG);**
- ✓ **J. Antão B. Moura (UFCG);**
- ✓ **Ulrich Schiel (UFCG);**
- ✓ **Maria Izabel C. Cabral (UFCG).**

UEPB / CCT / DC

Aritmética de Ponto Flutuante

Introdução

- Representação pode variar (“flutuar”) a posição da vírgula, ajustando potência da base.
  - $54,32 = 54,32 \times 10^{0} = 5,432 \times 10^{1} = 0,5432 \times 10^{2} = 5432,0 \times 10^{-2}$
  - Forma normalizada usa um único dígito antes da vírgula, diferente de zero
    - Exemplo: $5,432 \times 10^{1}$

Aritmética de Ponto Flutuante

Introdução

- No sistema binário:
  - $110101 = 110{,}101 \times 2^3 = 1{,}10101 \times 2^5 = 0{,}0110101 \times 2^7$
  - No caso dos números serem armazenados em um computador, os expoentes serão também gravados na base dois
    - $110{,}101 \times (10)^{11} = 1{,}10101 \times (10)^{101} = 0{,}0110101 \times (10)^{111}$
  - Na representação normalizada, há apenas um “1” antes da vírgula
    - Exemplo: $1{,}10101 \times (10)^{101}$

Aritmética de Ponto Flutuante

Introdução

- **Algumas definições**
  - No número **1,10101x**$(10)^{101}$, tomado como referência:
    - **1,10101** = **significando** (ou “mantissa”)
    - **101** = **expoente**
- **OBS:**
  - a base binária não precisa ser explicitada (o computador usa sempre esta)
  - O “1” antes da vírgula, na representação normalizada – se esta for adotada, também pode ficar implícito, economizando um bit (“bit escondido”).

Métodos Numéricos

Aritmética de Ponto Flutuante

Introdução

- **Representação genérica**

  – $\pm d_0,d_1d_2...d_t \mathbf{x}(b)^{exp}$,

    - **t** é o número de dígitos da mantissa
    - $d_1d_2...d_t$ = mantissa, com $0 \leq d_i \leq (b-1)$
    - **exp** = expoente (inteiro com sinal)

- **OBS:**
  – a base não precisa ser explicitada

Aritmética de Ponto Flutuante

Armazenamento de *floats*

- Na organização/arquitetura do computador, definir:
  - Número de bits da mantissa (precisão, ***p***)
  - Número de bits do expoente
  - Um bit de sinal (“0” para + e “1” para -) para o número (geralmente o primeiro, da esquerda)

Aritmética de Ponto Flutuante

Armazenamento de *floats*

- Ilustração

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Sinal	Expoente (+/-)			Significando			

- Sinal do número: 0 = + e 1 = -
- Expoentes: 8 combinações possíveis
  - 000 e 111 – especiais (ver adiante)
  - 011 (3_{10}) = expoente zero
  - 001 e 010 = expoente –2 e –1 (abaixo de zero)
  - 100, 101 e 110 = expoentes 1, 2 e 3 (acima zero)

Aritmética de Ponto Flutuante

Armazenamento de *floats*

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Sinal	Expoente (+/-)			Significando			

0 = +
1 = -

000 (especial)
001 (2^{-2})
010 (2^{-1})
011 (2^{0})
100 (2^{1})
101 (2^{2})
110 (2^{3})
111 (especial)

1,0000
1,0001
....
....
1,1111

1 = bit escondido

Aritmética de Ponto Flutuante

Armazenamento de *floats*

- Ainda os expoentes na ilustração...
  - **Maior número positivo é (lembre do bit escondido)**
    - 0 110 1111 = + 2^3 x 1,1111 = 2^3 x (2- 2^{-4}) = 1111,1 = 15,5 decimal
  - Menor número positivo é (lembre do bit escondido)
    - 0 001 0000 = + 2^{-2} x 1,0000 = 2^{-2} x 2^0 = 0,01 ou 0,25 decimal

Aritmética de Ponto Flutuante

Armazenamento de *floats*

- Combinações especiais dos expoentes na ilustração...
  - 000 – representação NÃO normalizada
    - Significando passa a ser 0,_ _ _ ...
    - Expoente (000) = -2
    - **Menor número positivo passa a ser**
    - $0\ 000\ 0001 = 2^{-2} \times 0{,}0001 = 2^{-2} \times 2^{-4} = 2^{-6} = 0{,}015625$

Aritmética de Ponto Flutuante

Armazenamento de *floats*

- Ainda as combinações especiais...
  - Normalização não permite representar zero!
  - 000 – representação NÃO normalizada
    - 00000000 = + 0 decimal
    - 10000000 = - 0 decimal
    - São iguais em comparações

Aritmética de Ponto Flutuante

Armazenamento de *floats*

- Ainda as combinações especiais...
  - 111 – representações de infinito
    - 01110000 = + infinito
    - 11110000 = - infinito
    - 11111000 = indeterminação
    - Outras combinações 11111_ _ _ = Not A Number (NANs)

Aritmética de Ponto Flutuante

Armazenamento de *floats*

- Exemplo: Armazenar 2,75

  - Resposta: 01000110

Aritmética de Ponto Flutuante

Padrão IEEE para *floats*

- O padrão IEEE 754 para ponto (vírgula) flutuante é a representação mais comum para números reais em computadores de hoje, incluindo PC's compatíveis com Intel, Macintosh, e a maioria das plataformas Unix/Linux.

Aritmética de Ponto Flutuante

Padrão IEEE para *floats*

- O padrão (ou norma) IEEE 754 define dois formatos básicos para os números em ponto flutuante:
  - o formato ou precisão simples, com 32 bits; e,
  - o duplo com 64 bits.

Aritmética de Ponto Flutuante

Padrão IEEE para *floats*

	Sinal	Expoente(+/-)	Significando
Simples (32bits)	1 [bit31]	8 [bits30-23]	23 [bits22-00]
Dupla (64 bits)	1 [bit63]	11 [bits62-52]	52 [bits51-00]

- Sinal: 0 = + e 1 = -
- Combinações Sinal + Expoente + Significando

Aritmética de Ponto Flutuante

IEEE 754 com precisão simples

- Expoentes na precisão simples c/256 combinações
  - 1111 1111
    - sinal=1 e significando = 0...0 : -infinito
    - sinal=0 e significando = 0...0 : +infinito
    - sinal=1 e significando =10...0: indeterminado
    - c/outras combinações: NAN

Aritmética de Ponto Flutuante

IEEE 754 com precisão simples

- Expoentes na precisão simples c/256 combinações
  - 0111 1111 (127_{10}) = expoente zero (*bias* = polarização)
  - 0000 0001 = menor expoente = –126 (abaixo de um)
  - 1111 1110 = maior expoente = +127 (acima de um)

    OBS: Expoente vale (Número em binário MENOS 127)
  - 0000 0000
    - sinal=1 e significando = 0...0 : -zero
    - sinal=0 e significando = 0...0 : +zero

Aritmética de Ponto Flutuante

IEEE 754 com precisão simples

- Expoentes na <u>precisão simples</u> c/256 combinações

(0) 0000 0000 (especial)
(1) 0000 0001 (2^{-126}) menor expoente
.....
0111 1100
(125) 0111 1101 (2^{-2})
(126) 0111 1110 (2^{-1})
(127) 0111 1111 (2^{0})
(128) 1000 0000 (2^{1})
(129) 1000 0001 (2^{2})
1000 0010
.....
(254) 1111 1110 (2^{127}) maior expoente
(255) 1111 1111 (especial)

Aritmética de Ponto Flutuante

IEEE 754 com precisão simples

- Menor número positivo (lembre do bit escondido e não normalizada)
  - 0 00000000 00....01 = $2^{-126} \times 2^{-23} = 2^{-149}$
- Maior número positivo (lembre do bit escondido)
  - 0 1111110 11...11 = $2^{127} \times (2-2^{-23})$

Aritmética de Ponto Flutuante

IEEE 754 com precisão dupla

- No formato (precisão) duplo, o menor expoente é representado por 00000000001, valendo -1022, e o maior expoente é representado por 11111111110, valendo +1023. Em ambos os casos, o expoente vale o número representado em binário menos 1023 (este é o valor da *bias* = zero).

Aritmética de Ponto Flutuante

IEEE 754 com precisão dupla

Verifique:

- Menor número positivo (lembre do bit escondido e não normalizada)
  - 0 00000000000 00….01 = $2^{-1022} \times 2^{-52} = 2^{-1074}$
- Maior número positivo (lembre do bit escondido)
  - 0 1111110 11...11 = $2^{1023} \times (2-2^{-52})$

Aritmética de Ponto Flutuante

IEEE 754 com precisão dupla

- Expoentes na precisão dupla c/2048 combinações

(0) 00000000000 (especial)
(1) 00000000001 (2^{-1022}) menor expoente
.....
01111111100
01111111101 (2^{-2})
(1022) 01111111110 (2^{-1})
(1023) 01111111111 (2^{0})
(1024 10000000000 (2^{1})
10000000001 (2^{2})
10000000010
.....
(2046) 11111111110 (2^{1023}) maior expoente
(2047) 11111111111 (especial)

Aritmética de Ponto Flutuante

Quadro resumo IEEE 754

	Não normalizado	Normalizado	Decimal
Simples	$\pm 2^{-149}$ a $(1-2^{-23}) \times 2^{-126}$	$\pm 2^{-126}$ a $(2-2^{-23}) \times 2^{127}$	$\pm \sim 10^{-44.85}$ a $\sim 10^{38.53}$
Dupla	$\pm 2^{-1074}$ a $(1-2^{-52}) \times 2^{-1022}$	$\pm 2^{-1022}$ a $(2-2^{-52}) \times 2^{1023}$	$\pm \sim 10^{-323.3}$ a $\sim 10^{308.3}$

Aritmética de Ponto Flutuante

Erro na representação de *floats*

- Número finito de bits na representação (número é apenas “maior” na precisão dupla), implica em “truncamento” (ou arredondamento) do número real a ser representado. Truncamento introduz erro na representação. Casos especiais:
  - *Overflow*: número a representar é maior que maior número possível de ser representado
  - *Underflow*: número a representar é menor que menor número possível de ser representado

Aritmética de Ponto Flutuante

Limite no erro na representação de um *float*

- A forma normalizada do número **N** é $1,\mathbf{n} \times 2^{\mathbf{e}}$
  - Supõe-se que **e** esteja dentro dos limites dessa representação (ou ocorreria *overflow*).
  - Se **n** não couber no número de bits da representação (precisão) do significando, **p**, haverá truncamento, introduzindo erro.

Aritmética de Ponto Flutuante

Limite no erro na representação de um *float*

- A forma normalizada do número **N** é 1,**n** x 2^{e}
- **Ex: N** = 1,101011110100101… x 2^{e} e que **p** número de bits (precisão) do significando seja 4.
  - A representação de **N** seria 1,1010 x 2^{e} gerando um $Erro_{N}$ = 0,11110100101… x 2^{e-4}
  - O erro relativo é definido como $E_{N} = Erro_{N} / \mathbf{N}$, ou:

    0,11110100101… x 2^{e-4} / 1,101011110100101… x 2^{e}

    = 0,11110100101… x 2^{-4} / 1,101011110100101…