DCC008 - Cálculo Numérico

Representação de Números

Bernardo Martins Rocha

Departamento de Ciência da Computação
Universidade Federal de Juiz de Fora
bernardomartinsrocha@ice.ufjf.br

Conteúdo

Introdução

Como estamos estudando métodos numéricos para resolver problemas de ciências e engenharias, é de extrema importância estudar e entender como os números são representados no computador e como as operações aritméticas são feitas.

Vamos então estudar como representar números no computador e os erros cometidos por conta da representação e durante as operações aritméticas.

Vamos começar revisando sistemas de numeração, conversão de bases, e depois estudamos como são representados os números inteiros e reais no computador.

Em seguida vamos estudar as operações aritméticas em ponto flutuante e alguns efeitos numéricos como propagação do erro, cancelamento, etc.

Sistema Decimal

No dia a dia usamos números baseados no sistema decimal. O número 257, por exemplo, pode ser escrito como

$$\begin{aligned} 257 &= 2 \cdot 100 + 5 \cdot 10 + 7 \cdot 1 \\ &= 2 \cdot 10^2 + 5 \cdot 10^1 + 7 \cdot 10^0 \end{aligned}$$

Sistema Decimal

No dia a dia usamos números baseados no sistema decimal. O número 257, por exemplo, pode ser escrito como

$$\begin{aligned} 257 &= 2 \cdot 100 + 5 \cdot 10 + 7 \cdot 1 \\ &= 2 \cdot 10^2 + 5 \cdot 10^1 + 7 \cdot 10^0 \end{aligned}$$

Chamamos de 10 de **base** deste sistema (decimal). Qualquer número inteiro pode ser expresso por um polinômio na base 10 com coeficientes entre 0 e 9.

Sistema Decimal

No dia a dia usamos números baseados no sistema decimal. O número 257, por exemplo, pode ser escrito como

$$\begin{aligned} 257 &= 2 \cdot 100 + 5 \cdot 10 + 7 \cdot 1 \\ &= 2 \cdot 10^2 + 5 \cdot 10^1 + 7 \cdot 10^0 \end{aligned}$$

Chamamos de 10 de **base** deste sistema (decimal). Qualquer número inteiro pode ser expresso por um polinômio na base 10 com coeficientes entre 0 e 9.

Usaremos a notação

$$\begin{aligned} N &= (a_n a_{n-1} \ldots a_0)_{10} \\ &= a_n \cdot 10^n + a_{n-1} \cdot 10^{n-1} + \ldots + a_0 \cdot 10^0 \end{aligned}$$

para representar qualquer inteiro no sistema decimal, onde os coeficientes são tais que $0 \leq a_i \leq 9$.

Sistema Binário

No passado outras civilizações usaram sistemas de numeração com bases diferentes como 12, 20 e 60.

Nos computadores modernos, os componentes elétricos possuem apenas dois estados "*on*"e "*off*". Portanto é mais conveniente representar números no computador usando o sistema binário, cuja base é 2.

Sistema Binário

No passado outras civilizações usaram sistemas de numeração com bases diferentes como 12, 20 e 60.

Nos computadores modernos, os componentes elétricos possuem apenas dois estados "*on*"e "*off*". Portanto é mais conveniente representar números no computador usando o sistema binário, cuja base é 2.

Neste sistema um número não-negativo é representado por

$$\begin{aligned} N &= (a_n a_{n-1} \ldots a_0)_2 \\ &= a_n \cdot 2^n + a_{n-1} \cdot 2^{n-1} + \ldots + a_0 \cdot 2^0 \end{aligned} \tag{1}$$

com a_k assumindo os valores 0 ou 1.

Sistema Hexadecimal

O sistema hexadecimal usa base 16 e portanto temos 16 dígitos diferentes. As vezes programadores usam o sistema hexadecimal pois o sistema binário pode ser cansativo e longo para representar algum valor. Exemplo:

$$\begin{aligned}(1234)_{16} &= 1 \times 16^3 + 2 \times 16^2 + 3 \times 16^1 + 4 \times 16^0 \\ &= 4096 + 512 + 48 + 4\end{aligned}$$

A representação hexadecimal usa as letras de A até F para os seis dígitos adicionais (10, 11, 12, 13, 14, 15). Outros exemplos:

$$(234)_{16} \qquad (BEEF)_{16} \qquad (DEAD)_{16} \qquad (0AFB)_{16}$$

A conversão entre os sistemas binário e hexadecimal é simples, uma vez que trabalhamos com grupos de 4 bits para cada dígito do sistema hexadecimal.

Sistema Hexadecimal

Binário	Hexadecimal	Binário	Hexadecimal
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

Podemos trabalhar no sistema octal, cuja base é 8, de forma análoga ao sistema hexadecimal. Quando queremos converter entre binário e octal, basta lembrar que cada grupo de 3 bits é um dígito do sistema octal.

Conversão binário para decimal

Pode ser obtida diretamente pelo uso da Equação (1).

Exemplos

$$\begin{aligned}(11)_2 &= 1 \cdot 2^1 + 1 \cdot 2^0 \\ &= 2 + 1 = 3\end{aligned}$$

Conversão binário para decimal

Pode ser obtida diretamente pelo uso da Equação (1).

Exemplos

$$(11)_2 = 1 \cdot 2^1 + 1 \cdot 2^0$$
$$= 2 + 1 = 3$$

$$(1101)_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0$$
$$= 8 + 4 + 0 + 1 = 13$$

Conversão decimal para binário

O procedimento é dividir o número por 2, a seguir continuar dividindo o quociente por 2, até que o quociente seja igual a 0. O número na base 2 é obtido tomando-se o resto das divisõees anteriores.

Conversão decimal para binário

O procedimento é dividir o número por 2, a seguir continuar dividindo o quociente por 2, até que o quociente seja igual a 0. O número na base 2 é obtido tomando-se o resto das divisõees anteriores.

Exemplo

Converter $(13)_{10}$ para binário.

Conversão decimal para binário

O procedimento é dividir o número por 2, a seguir continuar dividindo o quociente por 2, até que o quociente seja igual a 0. O número na base 2 é obtido tomando-se o resto das divisõees anteriores.

Exemplo

Converter $(13)_{10}$ para binário.

$$\begin{aligned} 13 \div 2 = 6, &\quad \text{resto=}1 \\ 6 \div 2 = 3, &\quad \text{resto=}0 \\ 3 \div 2 = 1, &\quad \text{resto=}1 \\ 1 \div 2 = 0, &\quad \text{resto=}1 \quad \uparrow \end{aligned}$$

Portanto

$$(13)_{10} = (1101)_2$$

Algoritmo conversão decimal para binário

entrada: numero inteiro decimal d
saída: numero binário b
i = 0;
q = d;
faça
    b[i] = q mod 2 ;
    q = q div 2 ;
    i = i + 1 ;
enquanto *(q ≠ 0)*;
inverteVetor(b);

Representação de números inteiros no computador

Inteiros são armazenados usando uma palavra de 32 bits no computador.

- Se estamos interessados em números **não negativos**, a representação é simples:

$(71)_{10} \Rightarrow$

00000000	00000000	00000000	01000111

Os valores que podemos representar vão de 0 a $2^{32} - 1$.

Representação de números inteiros no computador

Inteiros são armazenados usando uma palavra de 32 bits no computador.

- Se estamos interessados em números **não negativos**, a representação é simples:

$(71)_{10} \Rightarrow$

00000000	00000000	00000000	01000111

Os valores que podemos representar vão de 0 a $2^{32} - 1$.

- Para números inteiros com sinal, uma possibilidade, é reservar 1 bit para indicar qual o sinal do número. Exemplo com palavra de 8 bits para simplificar.

$(13)_{10} \Rightarrow$

0	0	0	0	1	1	0	1
bits 7	6	5	4	3	2	1	0

Bit de sinal:

- 0 $\Rightarrow$ positivo
- 1 $\Rightarrow$ negativo

Representação de números inteiros no computador

Complemento a dois

- Existe uma forma de representação mais esperta para números inteiros com sinal chamada de **complemento a dois**.

  O bit mais significativo ainda é usado para o sinal.

  Um número x positivo, tal que $0 \leq x \leq 2^{31} - 1$ é representado normalmente pela representação binária deste número, com bit de sinal 0.

  Entretanto, um número negativo $-y$, onde $1 \leq y \leq 2^{31}$ é armazenado como a representação binária do inteiro positivo:

$$2^{32} - y$$

Representação de números inteiros no computador

Complemento a dois

Para ilustrar considere uma palavra de 4 bits. Podemos representar os seguintes números de 0 a 7 e de -8 a -1. Veja a tabela.

número	bits	número	$2^B - y$	bits
0	0000	-8	1000	$2^4 - 8 = 8$
1	0001	-7	1001	$2^4 - 7 = 9$
2	0010	-6	1010	$2^4 - 6 = 10$
3	0011	-5	1011	$2^4 - 5 = 11$
4	0100	-4	1100	$2^4 - 4 = 12$
5	0101	-3	1101	$2^4 - 3 = 13$
6	0110	-2	1110	$2^4 - 2 = 14$
7	0111	-1	1111	$2^4 - 1 = 15$

Para negar um número em complemento a dois, use o seguinte algoritmo:

1. Inverta todos os bits do númeoro ($0 \to 1$ e $1 \to 0$).
2. Some 1 ao resultado invertido.

Exemplo

Para simplificar, considere uma palavra de 4 bits. Seja $x = 2$ e $y = 2$. Qual a representação em complemento a dois de $-y$?

Exemplo

Para simplificar, considere uma palavra de 4 bits. Seja $x = 2$ e $y = 2$. Qual a representação em complemento a dois de $-y$?

Solução do exemplo

Temos que $(2)_{10} = (0010)_2$. Pelos passos do algoritmo temos:

1. Invertendo os bits $0010 \rightarrow 1101$
2. Somando 1

$$\begin{array}{r} 1101 \\ +0001 \\ \hline 1110 \end{array}$$

Exemplo

Para simplificar, considere uma palavra de 4 bits. Seja $x = 2$ e $y = 2$. Qual a representação em complemento a dois de $-y$?

Solução do exemplo

Temos que $(2)_{10} = (0010)_2$. Pelos passos do algoritmo temos:

1. Invertendo os bits $0010 \rightarrow 1101$
2. Somando 1

$$\begin{array}{r} 1101 \\ +0001 \\ \hline 1110 \end{array}$$

Ou seja, $-y = -2$ é representado por $2^4 - 2 = 16 - 2 = 14$, cuja representação binária é $(14)_{10} = (1110)_2$.

Representação de números inteiros no computador

Complemento a dois

- A grande motivação para esse sistema é que não precisamos de hardware específico para a operação de subtração. O hardware de adição pode ser usado uma vez que $-y$ tenha sido representado com complemento a dois.
- Este sistema é usado em muitos dos computadores atuais.

Representação de números inteiros no computador

Complemento a dois

- A grande motivação para esse sistema é que não precisamos de hardware específico para a operação de subtração. O hardware de adição pode ser usado uma vez que $-y$ tenha sido representado com complemento a dois.
- Este sistema é usado em muitos dos computadores atuais.

Subtração: $x - y = 2 + (-2)$

$$\begin{array}{r} 0010 \\ +\ 1110 \\ \hline 10000 \end{array}$$

Como o bit mais à esquerda não pode ser representado neste sistema fictício cuja palavra tem 4 bits, ele é descartado e o resultado é 0, como esperado.

Representação de números fracionários

Se x é um número real positivo, sua parte integral x_i é o maior inteiro menor ou igual a x, enquanto

$$x_f = x - x_i$$

é a sua parte fracionária.

Representação de números fracionários

Se x é um número real positivo, sua parte integral x_i é o maior inteiro menor ou igual a x, enquanto

$$x_f = x - x_i$$

é a sua parte fracionária. A parte fracionária sempre pode ser escrita como uma fração decimal

$$x_f = \sum_{k=1}^{\infty} b_k 10^{-k} \tag{2}$$

onde cada b_k é um inteiro não-negativo menor que 10.

Representação de números fracionários

Se x é um número real positivo, sua parte integral x_i é o maior inteiro menor ou igual a x, enquanto

$$x_f = x - x_i$$

é a sua parte fracionária. A parte fracionária sempre pode ser escrita como uma fração decimal

$$x_f = \sum_{k=1}^{\infty} b_k 10^{-k} \tag{2}$$

onde cada b_k é um inteiro não-negativo menor que 10.

Se $b_k = 0$ para todo k maior do que algum inteiro, dizemos então que a fração **termina**. Caso contrário, dizemos que a fração **não termina**.

Representação de números fracionários

Exemplos

$$\frac{1}{4} = 0.25 = 2 \cdot 10^{-1} + 5 \cdot 10^{-2} \qquad \text{(termina)}$$

$$\frac{1}{3} = 0.333\ldots = 3 \cdot 10^{-1} + 3 \cdot 10^{-2} + 3 \cdot 10^{-3} \qquad \text{(não termina)}$$

□

Representação de números fracionários

Exemplos

$$\frac{1}{4} = 0.25 = 2 \cdot 10^{-1} + 5 \cdot 10^{-2} \qquad \text{(termina)}$$

$$\frac{1}{3} = 0.333\ldots = 3 \cdot 10^{-1} + 3 \cdot 10^{-2} + 3 \cdot 10^{-3} \qquad \text{(não termina)}$$

□

Se a parte integral de x é um inteiro no sistema decimal da forma

$$x_i = (a_n a_{n-1} \ldots a_0)_{10}$$

enquanto a parte fracionária é dada por (2), é comum escrever

$$x = (a_n a_{n-1} \ldots a_0 \cdot b_1 b_2 b_3 \ldots)_{10}$$

Representação de números fracionários

Algoritmo de conversão para binário

De forma análoga para o sistema binário podemos escrever

$$x_f = \sum_{k=1}^{\infty} b_k 2^{-k}, \qquad x_i = (a_n a_{n-1} \ldots a_0)_2$$

então $x = (a_n a_{n-1} \ldots a_0 \cdot b_1 b_2 b_2 \ldots)_2$.

A fração binária $(\cdot b_1 b_2 b_2 \ldots)_2$ para um dado x_f entre 0 e 1 pode ser calculada da seguinte forma: dado x entre 0 e 1, gere b_1, b_2, b_3 fazendo:

$$\begin{aligned}
c_0 &= x \\
b_1 &= (2 \cdot c_0)_i, \quad c_1 = (2 \cdot c_0)_f \\
b_2 &= (2 \cdot c_1)_i, \quad c_2 = (2 \cdot c_1)_f \\
b_3 &= (2 \cdot c_2)_i, \quad c_3 = (2 \cdot c_2)_f, \quad \ldots
\end{aligned}$$

onde $(.)_i$ representa a parte integral e $(.)_f$ a parte fracionária do número.

Representação de números fracionários

Algoritmo de conversão para binário

Exemplo 1

Qual a representação binária de $(0.625)_{10}$?

Exemplo 1

Qual a representação binária de $(0.625)_{10}$?

Solução do Exemplo

$$\begin{aligned} 2 \cdot 0.625 = 1.25 &\quad \Rightarrow \quad b_1 = 1 \\ 2 \cdot 0.25 = 0.50 &\quad \Rightarrow \quad b_2 = 0 \\ 2 \cdot 0.5 = 1.00 &\quad \Rightarrow \quad b_3 = 1 \\ 2 \cdot 0.0 = 0.00 &\quad \Rightarrow \quad b_4 = b_5 = \ldots = 0 \end{aligned}$$

Portanto $(0.625)_{10} = (0.101)_2$.

□

Representação de números fracionários

Algoritmo de conversão para binário

Exemplo 2

Qual a representação binária de $(0.1)_{10}$?

Exemplo 2

Qual a representação binária de $(0.1)_{10}$?

Solução do Exemplo 2

Temos $x = 0.1$.

$$
\begin{aligned}
2 \cdot 0.1 = 0.2 \quad &\Rightarrow \quad b_1 = 0 \\
2 \cdot 0.2 = 0.4 \quad &\Rightarrow \quad b_2 = 0 \\
2 \cdot 0.4 = 0.8 \quad &\Rightarrow \quad b_3 = 0 \\
2 \cdot 0.8 = 1.6 \quad &\Rightarrow \quad b_4 = 1 \\
2 \cdot 0.6 = 1.2 \quad &\Rightarrow \quad b_5 = 1 \\
2 \cdot 0.2 = 0.4 \quad &\Rightarrow \quad b_6 = 0 \\
2 \cdot 0.4 = 0.8 \quad &\ldots
\end{aligned}
$$

Portanto $(0.1)_{10} = (0.000110011\ldots)_2 = (0.0\overline{0011})_2$. □

Representação de números fracionários

Algoritmo de conversão para binário

entrada: x menor do que 1
saída: número binário b
$k = 1$;
faça
  se $2x >= 1$ **então**
    $b_k = 1$;
  senão
    $b_k = 0$;
  fim-se
  $x = 2x - b_k$;
  $k = k + 1$;
enquanto $(x \neq 0)$;
retorne $(b_1 b_2 \ldots b_k)$;

Mudança de base

Para transformar um número real que está representado na base 10 para a base 2, o procedimento é transformar a parte inteira e a parte fracionária usando os respectivos algoritmos já vistos.

Exemplos

Converter da base 10 para a base 2.

$$(5.75)_{10} =$$
$$(3.8)_{10} =$$
$$(33.023)_{10} =$$

Exemplo

Converter da base 2 para a base 10.

$$(11.0101)_2 =$$

Mudança de base

Para transformar um número real que está representado na base 10 para a base 2, o procedimento é transformar a parte inteira e a parte fracionária usando os respectivos algoritmos já vistos.

Exemplos

Converter da base 10 para a base 2.

$$(5.75)_{10} = (101.110)_2$$
$$(3.8)_{10} = (11.11001100\ldots)_2$$
$$(33.023)_{10} = (100001.00000101\ldots)_2$$

Exemplo

Converter da base 2 para a base 10.

$$(11.0101)_2 = (3.3125)_{10}$$

Representação de números reais

No computador números reais são armazenados usando o sistema binário e a representação destes números é finita.

Por exemplo, os números

$$\pi = 3.1415\ldots$$
$$e = 2.71828\ldots$$

não podem ser representados perfeitamente no computador. O que é armazenado então é uma versão aproximada destes números.

De forma geral, existem duas possibilidades para essa representação:

- Representação em Ponto Fixo
- **Representação em Ponto Flutuante (*)**

Representação em ponto fixo

Neste esquema de representação, a palavra do computador de usualmente 32 bits é dividida em 3 campos:

- 1 bit para o sinal
- campo de bits para a parte integral
- campo de bits para a parte fracionária

Exemplo

Sistema com 1 bit para sinal, 15 bits para parte integral e 16 bits para a parte fracionária. Neste sistema $(11/2)_{10} = (5.5)_{10}$ é representado da seguinte forma:

0	000000000000101	1000000000000000

Representação em ponto fixo

O sistema de ponto fixo é severamente limitado pelo tamanho dos números que este pode armazenar. No exemplo anterior, apenas números de magnitude entre 2^{-16} (exatamente) e 2^{15} (um pouco menos) podem ser armazenados no sistema.

Representação em ponto fixo

O sistema de ponto fixo é severamente limitado pelo tamanho dos números que este pode armazenar. No exemplo anterior, apenas números de magnitude entre 2^{-16} (exatamente) e 2^{15} (um pouco menos) podem ser armazenados no sistema.

Em algumas aplicações essa limitação pode não ser aceitável, e portanto este formato de representação é raramente utilizado.

Representação em ponto fixo

O sistema de ponto fixo é severamente limitado pelo tamanho dos números que este pode armazenar. No exemplo anterior, apenas números de magnitude entre 2^{-16} (exatamente) e 2^{15} (um pouco menos) podem ser armazenados no sistema.

Em algumas aplicações essa limitação pode não ser aceitável, e portanto este formato de representação é raramente utilizado.

O formato de representação mais usado para números reais é a representação de ponto flutuante, que descrevemos a seguir.

Representação em ponto flutuante

A representação de ponto flutuante é baseada na notação científica. Nessa notação um número real não-zero é expresso por

$$x = \pm d \times \beta^e$$

onde β é a base do sistema de numeração, d é a mantissa e e é o expoente.

Representação em ponto flutuante

A representação de ponto flutuante é baseada na notação científica. Nessa notação um número real não-zero é expresso por

$$x = \pm d \times \beta^e$$

onde β é a base do sistema de numeração, d é a mantissa e e é o expoente.

A mantissa é um número da forma

$$(0 \cdot d_1 d_2 d_3 \ldots d_t)_\beta$$

representada por t dígitos, onde $0 \leq d_i \leq (\beta - 1)$, para $i = 1, \ldots, t$ com $d_1 \neq 0$. O expoente e está no intervalo $[L, U]$.

Ao exigir que $d_1 \neq 0$, dizemos que o número está **normalizado**.

Representação em ponto flutuante

Iremos denotar um sistema de ponto flutuante por

$$F(\beta, t, L, U)$$

onde

- β é a base do sistema
- t número de dígitos da mantissa
- L menor valor para o expoente
- U maior valor para o expoente

Em qualquer máquina apenas um subconjunto dos números reais é representado exatamente, e portanto nesse processo a representação de um número real será feita com **arredondamento** ou **truncamento**.

Representação em ponto flutuante

Arredondamento em ponto flutuante

Imagine que só dispomos de quatro dígitos para representar os números em uma máquina. Como seria a melhor forma de representar $\frac{15}{7} = 2.142857 = x$? Usando 2.142 ou talvez 2.143?

Representação em ponto flutuante

Arredondamento em ponto flutuante

Imagine que só dispomos de quatro dígitos para representar os números em uma máquina. Como seria a melhor forma de representar $\frac{15}{7} = 2.142857 = x$? Usando 2.142 ou talvez 2.143? Se calcularmos o erro vemos que

$$|2.142 - x| = 0.000857$$
$$|2.143 - x| = 0.000143$$

Como o erro é menor para 2.143 concluimos que essa é a melhor forma de representar esse número. Este número foi arredondado.

O que significa arredondar um número?

Representação em ponto flutuante

Arredondamento em ponto flutuante

Imagine que só dispomos de quatro dígitos para representar os números em uma máquina. Como seria a melhor forma de representar $\frac{15}{7} = 2.142857 = x$? Usando 2.142 ou talvez 2.143? Se calcularmos o erro vemos que

$$|2.142 - x| = 0.000857$$
$$|2.143 - x| = 0.000143$$

Como o erro é menor para 2.143 concluimos que essa é a melhor forma de representar esse número. Este número foi arredondado.

O que significa arredondar um número?

Para arredondar um número na base 10, devemos apenas observar o primeiro dígito a ser descartado. Se este dígito é menor que 5 deixamos os dígitos inalterados; e se é maior ou igual a 5 devemos somar 1 ao último dígito remanescente.

Representação em ponto flutuante

Exemplo

Considere o seguinte sistema: $F(10, 3, -3, 3)$.

Neste sistema o número 12.5 é representado por

$$+(0.125)_{10} \times 10^2$$

O número de Euler

$$e = 2.718281\ldots$$

é representado por

Representação em ponto flutuante

Exemplo

Considere o seguinte sistema: $F(10, 3, -3, 3)$.

Neste sistema o número 12.5 é representado por

$$+(0.125)_{10} \times 10^2$$

O número de Euler

$$e = 2.718281\ldots$$

é representado por

$$\begin{aligned} &+ (0.271)_{10} \times 10^1 \quad \text{(com truncamento)} \\ &+ (0.272)_{10} \times 10^1 \quad \text{(com arredondamento)} \end{aligned}$$

□

Representação em ponto flutuante

Exemplo

Considere o seguinte sistema: $F(10, 3, -5, 5)$. Os números representados neste sistema terão a forma

$$\pm(0.d_1d_2d_3) \times 10^e, \quad 0 \leq d_i \leq 9, \quad d_1 \neq 0, \quad e \in [-5, 5]$$

Representação em ponto flutuante

Exemplo

Considere o seguinte sistema: $F(10, 3, -5, 5)$. Os números representados neste sistema terão a forma

$$\pm(0.d_1d_2d_3) \times 10^e, \quad 0 \leq d_i \leq 9, \quad d_1 \neq 0, \quad e \in [-5, 5]$$

O menor número em valor absoluto representado nessa máquina é

$$m = 0.100 \times 10^{-5} = 10^{-1} \times 10^{-5} = 10^{-6}$$

Representação em ponto flutuante

Exemplo

Considere o seguinte sistema: $F(10, 3, -5, 5)$. Os números representados neste sistema terão a forma

$$\pm(0.d_1d_2d_3) \times 10^e, \quad 0 \leq d_i \leq 9, \quad d_1 \neq 0, \quad e \in [-5, 5]$$

O menor número em valor absoluto representado nessa máquina é

$$m = 0.100 \times 10^{-5} = 10^{-1} \times 10^{-5} = 10^{-6}$$

O maior número em valor absoluto é

$$M = 0.999 \times 10^5 = 99900$$

□

Representação em ponto flutuante

Exemplo

Considere o seguinte sistema: $F(10, 3, -5, 5)$. Os números representados neste sistema terão a forma

$$\pm(0.d_1d_2d_3) \times 10^e, \quad 0 \leq d_i \leq 9, \quad d_1 \neq 0, \quad e \in [-5, 5]$$

O menor número em valor absoluto representado nessa máquina é

$$m = 0.100 \times 10^{-5} = 10^{-1} \times 10^{-5} = 10^{-6}$$

O maior número em valor absoluto é

$$M = 0.999 \times 10^5 = 99900$$

□

De forma geral, o menor e o maior número são dados por

$$m = \beta^{L-1}, \qquad M = \beta^U(1 - \beta^{-t})$$

Representação em ponto flutuante

Seja um sistema $F(\beta, t, L, U)$ onde o menor e maior número são denotados por m e M, respectivamente. Dado um número real x, então temos as seguintes situações:

Representação em ponto flutuante

Seja um sistema $F(\beta, t, L, U)$ onde o menor e maior número são denotados por m e M, respectivamente. Dado um número real x, então temos as seguintes situações:

1. $m \leq |x| \leq M$
   O número **pode** ser representado no sistema.
   Exemplo: $235.89 = 0.23589 \times 10^3$. No sistema $F(10, 3, -3, 3)$ temos

$$(0.235)_{10} \times 10^3 \text{ com truncamento}$$

$$(0.236)_{10} \times 10^3 \text{ com arredondamento}$$

Representação em ponto flutuante

Seja um sistema $F(\beta, t, L, U)$ onde o menor e maior número são denotados por m e M, respectivamente. Dado um número real x, então temos as seguintes situações:

1. $m \leq |x| \leq M$
   O número **pode** ser representado no sistema.
   Exemplo: $235.89 = 0.23589 \times 10^3$. No sistema $F(10, 3, -3, 3)$ temos

   $$(0.235)_{10} \times 10^3 \text{ com truncamento}$$
   $$(0.236)_{10} \times 10^3 \text{ com arredondamento}$$

2. $|x| \leq m$
   O número **não pode** ser representado no sistema. Neste caso dizemos que ocorreu **underflow**. Ex: 0.517×10^{-8}.

Representação em ponto flutuante

Seja um sistema $F(\beta, t, L, U)$ onde o menor e maior número são denotados por m e M, respectivamente. Dado um número real x, então temos as seguintes situações:

1. $m \leq |x| \leq M$
   O número **pode** ser representado no sistema.
   Exemplo: $235.89 = 0.23589 \times 10^3$. No sistema $F(10, 3, -3, 3)$ temos

   $$(0.235)_{10} \times 10^3 \text{ com truncamento}$$
   $$(0.236)_{10} \times 10^3 \text{ com arredondamento}$$

2. $|x| \leq m$
   O número **não pode** ser representado no sistema. Neste caso dizemos que ocorreu **underflow**. Ex: 0.517×10^{-8}.
3. $|x| \geq M$
   O número **não pode** ser representado no sistema. Neste caso dizemos que ocorreu **overflow**. Ex: 0.725×10^9.

Representação em ponto flutuante

Exemplo

Considere o sistema $F(2, 3, -3, 3)$. Para simplificar, considere uma palavra com 7 bits, onde temos 1 bit para o sinal do número, 3 bits para o expoente (incluindo seu sinal) e 3 bits para a mantissa. Vamos representar $x_{min} = (0.100)_2 \times 2^{-3}$.

Representação em ponto flutuante

Exemplo

Considere o sistema $F(2, 3, -3, 3)$. Para simplificar, considere uma palavra com 7 bits, onde temos 1 bit para o sinal do número, 3 bits para o expoente (incluindo seu sinal) e 3 bits para a mantissa. Vamos representar $x_{min} = (0.100)_2 \times 2^{-3}$.

sinal do número	0
sinal do expoente	1
expoente	$(3)_{10} = (11)_2$
mantissa	$(0.100)_2$

Sendo assim temos a seguinte representação para $(0.0625)_{10}$ neste sistema

0	111	100

□

Representação em ponto flutuante

Exemplo

Considere o sistema $F(2, 3, -1, 2)$. Quantos e quais números podem ser representados neste sistema?

Representação em ponto flutuante

Exemplo

Considere o sistema $F(2, 3, -1, 2)$. Quantos e quais números podem ser representados neste sistema?

Solução do Exemplo

Neste sistema os números são da forma

$$\pm 0.d_1 d_2 d_3 \times 2^e$$

Representação em ponto flutuante

Exemplo

Considere o sistema $F(2, 3, -1, 2)$. Quantos e quais números podem ser representados neste sistema?

Solução do Exemplo

Neste sistema os números são da forma

$$\pm 0.d_1 d_2 d_3 \times 2^e$$

temos

- 2 possibilidades para o sinal

Representação em ponto flutuante

Exemplo

Considere o sistema $F(2, 3, -1, 2)$. Quantos e quais números podem ser representados neste sistema?

Solução do Exemplo

Neste sistema os números são da forma

$$\pm 0.d_1 d_2 d_3 \times 2^e$$

temos

- 2 possibilidades para o sinal
- $(1 \cdot 2 \cdot 2) = 4$ possibilidades para a mantissa

Representação em ponto flutuante

Exemplo

Considere o sistema $F(2, 3, -1, 2)$. Quantos e quais números podem ser representados neste sistema?

Solução do Exemplo

Neste sistema os números são da forma

$$\pm 0.d_1 d_2 d_3 \times 2^e$$

temos

- 2 possibilidades para o sinal
- $(1 \cdot 2 \cdot 2) = 4$ possibilidades para a mantissa
- 4 possibilidades para o expoente $(-1, 0, 1, 2)$

Representação em ponto flutuante

Exemplo

Considere o sistema $F(2, 3, -1, 2)$. Quantos e quais números podem ser representados neste sistema?

Solução do Exemplo

Neste sistema os números são da forma

$$\pm 0.d_1 d_2 d_3 \times 2^e$$

temos

- 2 possibilidades para o sinal
- $(1 \cdot 2 \cdot 2) = 4$ possibilidades para a mantissa
- 4 possibilidades para o expoente $(-1, 0, 1, 2)$

portanto temos $2 \cdot 4 \cdot 4 = 32$, e considerando que o zero também faz parte do sistema, concluimos que podemos representar 33 números distintos.

Solução Exemplo

Para responder quais são os números, notemos que as possíveis formas da mantissa são 0.100, 0.101, 0.110 e 0.111 e as formas do expoente são 2^{-1}, 2^{0}, 2^{1} e 2^{2}.

Solução Exemplo

Para responder quais são os números, notemos que as possíveis formas da mantissa são 0.100, 0.101, 0.110 e 0.111 e as formas do expoente são 2^{-1}, 2^0, 2^1 e 2^2. Assim obtemos:

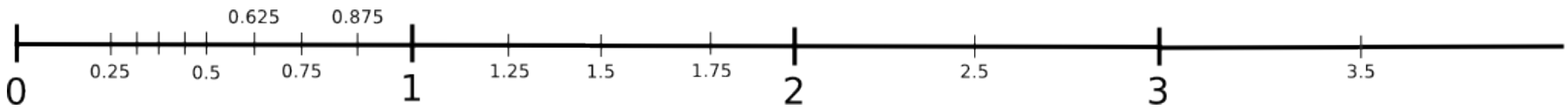
$(0.100)_2 \times 2^{-1}$	$= (0.25)_{10}$	$(0.101)_2 \times 2^{-1}$	$= (0.3125)_{10}$
$(0.100)_2 \times 2^0$	$= (0.5)_{10}$	$(0.101)_2 \times 2^0$	$= (0.625)_{10}$
$(0.100)_2 \times 2^1$	$= (1.0)_{10}$	$(0.101)_2 \times 2^1$	$= (1.25)_{10}$
$(0.100)_2 \times 2^2$	$= (2.0)_{10}$	$(0.101)_2 \times 2^2$	$= (2.5)_{10}$
$(0.110)_2 \times 2^{-1}$	$= (0.375)_{10}$	$(0.111)_2 \times 2^{-1}$	$= (0.4375)_{10}$
$(0.110)_2 \times 2^0$	$= (0.75)_{10}$	$(0.111)_2 \times 2^0$	$= (0.875)_{10}$
$(0.110)_2 \times 2^1$	$= (1.5)_{10}$	$(0.111)_2 \times 2^1$	$= (1.75)_{10}$
$(0.110)_2 \times 2^2$	$= (3.0)_{10}$	$(0.111)_2 \times 2^2$	$= (3.5)_{10}$

O zero é representado de uma forma especial: todos os dígitos d_i da mantissa e do expoente são nulos.

Representação em ponto flutuante

Solução Exemplo

É extremamente importante observar com relação aos números de ponto flutante, que eles são **discretos** e não contínuos como um número real $x \in \mathbb{R}$, como definido usualmente na matemática.
A figura abaixo ilustra essa característica para o sistema do exemplo (por simplicidade, omitimos a exibição dos números negativos).



□

Representação em ponto flutuante

Exemplo

Considerando o mesmo sistema do exemplo anterior, $F(2, 3, -1, 2)$, represente os números: $x_1 = 0.38$, $x_2 = 5.3$ e $x_3 = 0.15$ dados na base 10.

Representação em ponto flutuante

Exemplo

Considerando o mesmo sistema do exemplo anterior, $F(2, 3, -1, 2)$, represente os números: $x_1 = 0.38$, $x_2 = 5.3$ e $x_3 = 0.15$ dados na base 10.

Solução Exemplo 4

Convertendo os números para binário temos

$$(0.38)_{10} =$$
$$(5.3)_{10} =$$
$$(0.15)_{10} =$$

Representação em ponto flutuante

Exemplo

Considerando o mesmo sistema do exemplo anterior, $F(2, 3, -1, 2)$, represente os números: $x_1 = 0.38$, $x_2 = 5.3$ e $x_3 = 0.15$ dados na base 10.

Solução Exemplo 4

Convertendo os números para binário temos

$$(0.38)_{10} = 0.110 \cdot 2^{-1}$$
$$(5.3)_{10} = 0.101 \cdot 2^{3}$$
$$(0.15)_{10} = 0.100 \cdot 2^{-2}$$

Representação em ponto flutuante

Exemplo

Considerando o mesmo sistema do exemplo anterior, $F(2, 3, -1, 2)$, represente os números: $x_1 = 0.38$, $x_2 = 5.3$ e $x_3 = 0.15$ dados na base 10.

Solução Exemplo 4

Convertendo os números para binário temos

$$
\begin{aligned}
(0.38)_{10} &= 0.110 \cdot 2^{-1} \quad \Rightarrow \quad \text{OK} \\
(5.3)_{10} &= 0.101 \cdot 2^{3} \quad \Rightarrow \quad \text{overflow} \\
(0.15)_{10} &= 0.100 \cdot 2^{-2} \quad \Rightarrow \quad \text{underflow}
\end{aligned}
$$

□

Formato ponto flutuante IEEE 754

Por volta dos anos 50 computação usando ponto flutuante estava em alta. Cada fabricante de computador desenvolvia o seu sistema de ponto flutuante, o que levava a muita inconsistência a como um programa funcionava em diferentes máquinas.

Apesar da maioria das máquinas usarem o sistema binário, algumas como as da série IBM 360/370 usavam o sistema hexadecimal.

Formato ponto flutuante IEEE 754

Por volta dos anos 50 computação usando ponto flutuante estava em alta. Cada fabricante de computador desenvolvia o seu sistema de ponto flutuante, o que levava a muita inconsistência a como um programa funcionava em diferentes máquinas.

Apesar da maioria das máquinas usarem o sistema binário, algumas como as da série IBM 360/370 usavam o sistema hexadecimal.

Em resumo, era muito difícil escrever um software **portável** que pudesse funcionar corretamente em todas as máquinas.

Em uma cooperação extraordinária entre cientistas e desenvolvedores de microprocessadores, um padrão para representação de números binários usando ponto flutuante foi desenvolvida por volta no final dos anos 70, início dos anos 80.

Formato ponto flutuante IEEE 754

O padrão usa o sistema binário e dois formatos para representação de números podem ser adotados: precisão simples e precisão dupla.

Neste formato um número é representado de forma **normalizada** por

$$\pm 1.d_1 d_2 \ldots d_n \times 2^e$$

Formato ponto flutuante IEEE 754

O padrão usa o sistema binário e dois formatos para representação de números podem ser adotados: precisão simples e precisão dupla.

Neste formato um número é representado de forma **normalizada** por

$$\pm 1.d_1 d_2 \ldots d_n \times 2^e$$

Em precisão simples um número real é representado por 32 bits, sendo que:

- 1 bit para o sinal
- 8 bits para o expoente
- 23 bits para a mantissa

Formato ponto flutuante IEEE 754

O padrão usa o sistema binário e dois formatos para representação de números podem ser adotados: precisão simples e precisão dupla.

Neste formato um número é representado de forma **normalizada** por

$$\pm 1.d_1 d_2 \ldots d_n \times 2^e$$

Em precisão simples um número real é representado por 32 bits, sendo que:

- 1 bit para o sinal
- 8 bits para o expoente
- 23 bits para a mantissa

ou seja, tem o seguinte formato binário

$\pm$	$e_1 e_2 \ldots e_8$	$d_1 d_2 \ldots d_{23}$

Formato ponto flutuante IEEE 754

O padrão usa o sistema binário e dois formatos para representação de números podem ser adotados: precisão simples e precisão dupla.

Neste formato um número é representado de forma **normalizada** por

$$\pm 1.d_1 d_2 \ldots d_n \times 2^e$$

Em precisão simples um número real é representado por 32 bits, sendo que:

- 1 bit para o sinal
- 8 bits para o expoente
- 23 bits para a mantissa

ou seja, tem o seguinte formato binário

$\pm$	$e_1 e_2 \ldots e_8$	$d_1 d_2 \ldots d_{23}$

O primeiro bit à esquerda do ponto binário, isto é $d_0 = 1$, é chamado de **bit escondido** (*hidden bit*).

Formato ponto flutuante IEEE 754

Expoente

Nesse formato, o expoente não é representado como um inteiro via complemento a dois. Os oito bits do expoente armazenam o número $s = e + 127$.

Formato ponto flutuante IEEE 754

Expoente

Nesse formato, o expoente não é representado como um inteiro via complemento a dois. Os oito bits do expoente armazenam o número $s = e + 127$.

Exemplos:

$$
\begin{aligned}
e = 1 \quad &\Rightarrow \quad s = 1 + 127 = (128)_{10} = (10000000)_2 \\
e = -3 \quad &\Rightarrow \quad s = -3 + 127 = (124)_{10} = (01111100)_2 \\
e = 52 \quad &\Rightarrow \quad s = 52 + 127 = (179)_{10} = (10110011)_2
\end{aligned}
$$

Formato ponto flutuante IEEE 754

Expoente

Nesse formato, o expoente não é representado como um inteiro via complemento a dois. Os oito bits do expoente armazenam o número $s = e + 127$.

Exemplos:

$$
\begin{aligned}
e = 1 \quad &\Rightarrow \quad s = 1 + 127 = (128)_{10} = (10000000)_2 \\
e = -3 \quad &\Rightarrow \quad s = -3 + 127 = (124)_{10} = (01111100)_2 \\
e = 52 \quad &\Rightarrow \quad s = 52 + 127 = (179)_{10} = (10110011)_2
\end{aligned}
$$

Em particular as sequências de bits (00000000) e (11111111) para o expoente, são usadas para representar, respectivamente, o zero e infinito ou ocorrência de erro, que é denotado por NaN (Not a Number).

Formato ponto flutuante IEEE 754

Mantissa

Vamos considerar agora a representação da mantissa. Como o sistema é normalizado temos $d_0 \neq 0$. Dado que a base é dois, a única possibilidade para o primeiro dígito será sempre igual a 1, e portanto este bit não precisa ser armazenado, por isso é chamado de **bit escondido**.

Formato ponto flutuante IEEE 754

Mantissa

Vamos considerar agora a representação da mantissa. Como o sistema é normalizado temos $d_0 \neq 0$. Dado que a base é dois, a única possibilidade para o primeiro dígito será sempre igual a 1, e portanto este bit não precisa ser armazenado, por isso é chamado de **bit escondido**.

Com o uso desta normalização temos um ganho na precisão, pois a mantissa passa a ser representada com 24 bits (23 + 1 bit escondido).

Formato ponto flutuante IEEE 754

Mantissa

Vamos considerar agora a representação da mantissa. Como o sistema é normalizado temos $d_0 \neq 0$. Dado que a base é dois, a única possibilidade para o primeiro dígito será sempre igual a 1, e portanto este bit não precisa ser armazenado, por isso é chamado de **bit escondido**.

Com o uso desta normalização temos um ganho na precisão, pois a mantissa passa a ser representada com 24 bits (23 + 1 bit escondido).

Exemplo: $(0.125)_{10} = (0.001)_2 = 1.0 \times 2^{-3}$ é armazenado como:

0	01111100	0000000 00000000 00000000

A mantissa só possui um dígito significativo, que é justamente o bit escondido, e portanto os demais 23 bits são representados com 0.

Formato ponto flutuante IEEE 754

Table 1: IEEE Single Precision

$\pm$	$a_1 a_2 a_3 \ldots a_8$	$b_1 b_2 b_3 \ldots b_{23}$

If exponent bitstring $a_1 \ldots a_8$ is	Then numerical value represented is
$(00000000)_2 = (0)_{10}$	$\pm(0.b_1 b_2 b_3 \ldots b_{23})_2 \times 2^{-126}$
$(00000001)_2 = (1)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{23})_2 \times 2^{-126}$
$(00000010)_2 = (2)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{23})_2 \times 2^{-125}$
$(00000011)_2 = (3)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{23})_2 \times 2^{-124}$
$\downarrow$	$\downarrow$
$(01111111)_2 = (127)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{23})_2 \times 2^{0}$
$(10000000)_2 = (128)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{23})_2 \times 2^{1}$
$\downarrow$	$\downarrow$
$(11111100)_2 = (252)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{23})_2 \times 2^{125}$
$(11111101)_2 = (253)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{23})_2 \times 2^{126}$
$(11111110)_2 = (254)_{10}$	$\pm(1.b_1 b_2 b_3 \ldots b_{23})_2 \times 2^{127}$
$(11111111)_2 = (255)_{10}$	$\pm\infty$ if $b_1 = \ldots = b_{23} = 0$, NaN otherwise

Formato ponto flutuante IEEE 754

O **menor** número normalizado positivo representável nesse padrão é

0	00000001	0000000 00000000 00000000

nesse caso, como $s = (00000001)_2 = (1)_{10}$, o expoente é

$$\begin{aligned} s &= e + 127 \\ 1 &= e + 127 \quad \Rightarrow \quad e = -126 \end{aligned}$$

assim, o número é

$$(1.0000000\ldots)_2 \times 2^{-126} = 2^{-126} \approx 1.2 \times 10^{-38}$$

Formato ponto flutuante IEEE 754

O **maior** número normalizado positivo, terá como mantissa um número com 24 bits iguais a 1 e como expoente $(11111110)_2$, isto é

0	11111110	1111111 11111111 11111111

nesse caso, como $s = (11111110)_2 = (254)_{10}$, temos

$$s = e + 127 \quad \Rightarrow \quad 254 = e + 127 \quad \Rightarrow \quad e = 127$$

e ainda

$$(1.1111\ldots1)_2 = 1 + \frac{1}{2} + \ldots + \frac{1}{2^{23}} = \frac{2^{23} + 2^{22} + \ldots + 1}{2^{23}}$$
$$= \frac{\sum_{i=0}^{23} 2^i}{2^{23}} = \frac{2^{24} - 1}{2^{23}} = 2 - \frac{1}{2^{23}}$$

assim

$$(1.111111\ldots)_2 \times 2^{127} = (2 - 2^{-23}) \times 2^{127} \approx 3.4 \times 10^{38}$$

Formato ponto flutuante IEEE 754

O zero é representado com zeros para expoente e mantissa

0	00000000	0000000 00000000 00000000

Formato ponto flutuante IEEE 754

O zero é representado com zeros para expoente e mantissa

0	00000000	0000000 00000000 00000000

Os valores $+\infty$ e $-\infty$ são representados por

0	11111111	0000000 00000000 00000000
1	11111111	0000000 00000000 00000000

Formato ponto flutuante IEEE 754

O zero é representado com zeros para expoente e mantissa

0	00000000	0000000 00000000 00000000

Os valores $+\infty$ e $-\infty$ são representados por

0	11111111	0000000 00000000 00000000
1	11111111	0000000 00000000 00000000

Se a sequência de bits para o expoente for composta por todos dígitos iguais a um e a da mantissa for não nula, isto é:

1	11111111	$xxxxxxx\ xxxxxxxx\ xxxxxxxx$

temos a ocorrência de *NaN: not a number*, que representam expressões inválidas como:

$$0 * \infty, \quad 0/0, \quad \infty/\infty, \quad \infty - \infty$$

Formato ponto flutuante IEEE 754

Por outro lado certas expressões que envolvam ∞ e zero possuem um resultado plausível:

$$z * 0 = 0$$
$$z/0 = +\infty \text{ se } z > 0$$
$$z/0 = -\infty \text{ se } z < 0$$
$$z * \infty = \infty$$
$$\infty + \infty = \infty$$

Formato ponto flutuante IEEE 754

```
#include <stdio.h>
#include <math.h>
int main() {
    float x, y;

    x = 23.0/0.0;
    printf("x = %f\n", x);

    y = sqrt(-5.0);
    printf("y = %f\n", y);

    printf("a = %f\n",       x*0);
    printf("0/0 = %f\n",     0./0.);
    printf("inf/inf = %f\n", x/x);
    printf("inf-inf = %f\n", x-x);
}
```

Saída

```
x = inf
y = -nan
a = -nan
0/0 = -nan
inf/inf = -nan
inf-inf = -nan
```

Formato ponto flutuante IEEE 754

$\pm$	expoente	mantissa	valor
0	00000000	00000000000000000000000	0
1	00000000	00000000000000000000000	-0
0	11111111	00010000000000000000000	NaN
1	11111111	00010100100100100100000	NaN
0	11111111	00000000000000000000000	∞
1	11111111	00000000000000000000000	$-\infty$
0	10000001	10100000000000000000000	$1.101 \cdot 2^{129-127} = 6.5$
1	10000001	10100000000000000000000	$-1.101 \cdot 2^{129-127} = -6.5$
0	10000000	00000000000000000000000	$1.0 \cdot 2^{128-127} = 2$
0	10000010	10100000000000000000000	$1.0 \cdot 2^{130-127} = 8$

Formato ponto flutuante IEEE 754

A precisão p de um sistema de ponto flutuante é definida como o número de bits da mantissa (incluindo bits escondidos). Sendo assim a precisão simples do formato IEEE 754 é $p = 24$, o que corresponde aproximadamente a 7 dígitos decimais significantes, já que

$$2^{-24} \approx 10^{-7}$$

Formato ponto flutuante IEEE 754

A precisão p de um sistema de ponto flutuante é definida como o número de bits da mantissa (incluindo bits escondidos). Sendo assim a precisão simples do formato IEEE 754 é $p = 24$, o que corresponde aproximadamente a 7 dígitos decimais significantes, já que

$$2^{-24} \approx 10^{-7}$$

De forma análoga, na precisão dupla com $p = 53$ temos aproximadamente 16 dígitos significantes.

Por exemplo, a representação em precisão simples de

$$\pi = 3.141592653\ldots$$

Formato ponto flutuante IEEE 754

A precisão p de um sistema de ponto flutuante é definida como o número de bits da mantissa (incluindo bits escondidos). Sendo assim a precisão simples do formato IEEE 754 é $p = 24$, o que corresponde aproximadamente a 7 dígitos decimais significantes, já que

$$2^{-24} \approx 10^{-7}$$

De forma análoga, na precisão dupla com $p = 53$ temos aproximadamente 16 dígitos significantes.

Por exemplo, a representação em precisão simples de

$$\pi = 3.141592653\ldots$$

é

$$\underline{3.141592}741\ldots$$

Formato ponto flutuante IEEE 754

Propriedade	Simples	Dupla	Estendida
comprimento total	32	64	80
bits na mantissa	23	52	64
bits no expoente	8	11	15
base	2	2	2
expoente máximo	127	1023	16383
expoente mínimo	-126	-1022	-16382
maior número	$\approx 3.4 \times 10^{38}$	$\approx 1.8 \times 10^{308}$	$\approx 1.2 \times 10^{4932}$
menor número	$\approx 1.2 \times 10^{-38}$	$\approx 2.2 \times 10^{-308}$	$\approx 3.4 \times 10^{-4932}$
dígitos decimais	7	16	19

Formato ponto flutuante IEEE 754

Propriedade	Simples	Dupla	Estendida
comprimento total	32	64	80
bits na mantissa	23	52	64
bits no expoente	8	11	15
base	2	2	2
expoente máximo	127	1023	16383
expoente mínimo	-126	-1022	-16382
maior número	$\approx 3.4 \times 10^{38}$	$\approx 1.8 \times 10^{308}$	$\approx 1.2 \times 10^{4932}$
menor número	$\approx 1.2 \times 10^{-38}$	$\approx 2.2 \times 10^{-308}$	$\approx 3.4 \times 10^{-4932}$
dígitos decimais	7	16	19

Na linguagem C

- `float`: precisão simples
- `double`: precisão dupla
- `long double`: precisão estendida

Formato ponto flutuante IEEE 754

Exercícios

Considere os seguintes números representados no sistema decimal:

a) 2

b) 30

c) 5.75

d) 0.1

Escreva a representação de cada um dos números no formato de ponto flutuante IEEE 754 usando precisão simples.

Conteúdo

- Última aula
  - Representação de Números Inteiros
  - Representação de Números Reais
    - Ponto Fixo
    - Ponto Flutuante
  - Padrão de Ponto Flutuante IEEE 754
- Aula de hoje
  - Operações aritméticas com ponto flutuante
  - Noções básicas de erro
  - Erro no arredondamento e truncamento
  - Efeitos numéricos

Operações aritméticas em ponto flutuante

- **Adição/subtração**: quando dois números em ponto flutuante são somados (ou subtraídos), é preciso alinhar as casas decimais do número de menor expoente para a direita até que os expoentes fiquem iguais.
- **Multiplicação/divisão**: nessa operação realizamos o produto (ou divisão) das mantissas e o expoente final da base é obtido, somando (subtraindo) os expoentes de cada parcela.
- Os resultados devem ser truncados ou arredondados.
- Truncamento ou arredondamento depende da máquina.

Operações aritméticas em ponto flutuante

Exemplo

Adicionar 4.32 e 0.064 em uma máquina com mantissa $t = 2$ e base 10.

Operações aritméticas em ponto flutuante

Exemplo

Adicionar 4.32 e 0.064 em uma máquina com mantissa $t = 2$ e base 10.

Solução do Exemplo

$$\begin{aligned} 4.32 + 0.064 = 0.43 \times 10^1 + 0.64 \times 10^{-1} = & \quad 0.4300 \times 10^1 \\ & + 0.0064 \times 10^1 \\ & = 0.4364 \times 10^1 \end{aligned}$$

Operações aritméticas em ponto flutuante

Exemplo

Adicionar 4.32 e 0.064 em uma máquina com mantissa $t = 2$ e base 10.

Solução do Exemplo

$$4.32 + 0.064 = 0.43 \times 10^1 + 0.64 \times 10^{-1} = \begin{aligned} & 0.4300 \times 10^1 \\ & + 0.0064 \times 10^1 \\ & = 0.4364 \times 10^1 \end{aligned}$$

Truncamento $\rightarrow 0.43 \times 10^1$
Arredondamento $\rightarrow 0.44 \times 10^1$

□

Operações aritméticas em ponto flutuante

Exemplo

Subtrair 371 de 372 em uma máquina com mantissa $t = 2$ e base 10.

Operações aritméticas em ponto flutuante

Exemplo

Subtrair 371 de 372 em uma máquina com mantissa $t = 2$ e base 10.

Solução do Exemplo

$$\begin{aligned} 372 - 371 = 0.37 \times 10^3 - 0.37 \times 10^3 = & \quad 0.37 \times 10^3 \\ & - 0.37 \times 10^3 \\ & = 0.00 \times 10^3 \end{aligned}$$

Operações aritméticas em ponto flutuante

Exemplo

Subtrair 371 de 372 em uma máquina com mantissa $t = 2$ e base 10.

Solução do Exemplo

$$372 - 371 = 0.37 \times 10^3 - 0.37 \times 10^3 = \begin{array}{r} 0.37 \times 10^3 \\ - \ 0.37 \times 10^3 \\ = 0.00 \times 10^3 \end{array}$$

A subtração deu 0 em vez de 1. Problema na subtração de dois números aproximadamente iguais.

□

Operações aritméticas em ponto flutuante

Exemplo

Multiplicar 1234 por 0.016 em uma máquina com mantissa $t = 2$ e base 10.

Operações aritméticas em ponto flutuante

Exemplo

Multiplicar 1234 por 0.016 em uma máquina com mantissa $t = 2$ e base 10.

Solução do Exemplo

$$
\begin{aligned}
1234 * 0.016 = 0.12 \times 10^4 * 0.16 \times 10^{-1} = \quad & 0.12 \times 10^4 \\
& * 0.16 \times 10^{-1} \\
& = 0.0192 \times 10^3 \\
& = 0.19 \times 10^2
\end{aligned}
$$

Operações aritméticas em ponto flutuante

Exemplo

Multiplicar 1234 por 0.016 em uma máquina com mantissa $t = 2$ e base 10.

Solução do Exemplo

$$
\begin{aligned}
1234 * 0.016 = 0.12 \times 10^4 * 0.16 \times 10^{-1} = \quad & 0.12 \times 10^4 \\
& * 0.16 \times 10^{-1} \\
& = 0.0192 \times 10^3 \\
& = 0.19 \times 10^2
\end{aligned}
$$

Neste caso usando arredondamento ou truncamento, o resultado é 19, em vez de 19.744 que é o resultado exato.

□

Operações aritméticas em ponto flutuante

Exemplo

Dividir 0.00183 por 492 em uma máquina com mantissa $t = 2$ e base 10.

Operações aritméticas em ponto flutuante

Exemplo

Dividir 0.00183 por 492 em uma máquina com mantissa $t = 2$ e base 10.

Solução do Exemplo

$$0.00183 \div 492 = 0.18 \times 10^{-2} \div 0.49 \times 10^{3} = \begin{aligned} & 0.18 \times 10^{-2} \\ & \div\ 0.49 \times 10^{3} \\ & = 0.3673 \times 10^{-5} \end{aligned}$$

Operações aritméticas em ponto flutuante

Exemplo

Dividir 0.00183 por 492 em uma máquina com mantissa $t = 2$ e base 10.

Solução do Exemplo

$$0.00183 \div 492 = 0.18 \times 10^{-2} \div 0.49 \times 10^{3} = \begin{array}{r} 0.18 \times 10^{-2} \\ \div\ 0.49 \times 10^{3} \\ = 0.3673 \times 10^{-5} \end{array}$$

Arredondamento $\rightarrow 0.37 \times 10^{-5}$

Truncamento $\rightarrow 0.36 \times 10^{-5}$

□

Operações aritméticas em ponto flutuante

É importante observar que algumas propriedades aritméticas como

$$\text{associatividade: } (a+b)+c = a+(b+c)$$
$$\text{distributividade: } a(b+c) = ab+ac$$

não são válidas em sistemas de ponto flutuante.

Para os exemplos a seguir, considere um sistema com base β e três dígitos na mantissa, ou seja, $F(10,3,L,U)$. Considere ainda que o sistema trabalha com arredondamento após cada uma das operações efetuadas.

Exemplo

a) $(11.4+3.18)+5.05$ e $11.4+(3.18+5.05)$
b) $5.55(4.45-4.35)$ e $5.55*4.45-5.55*4.35$

Operações aritméticas em ponto flutuante

Solução do Exemplo

a) $(11.4 + 3.18) + 5.05$

$$
\begin{aligned}
& 0.1140 \times 10^2 \\
+\ & 0.0318 \times 10^2 \\
= \ & 0.1458 \times 10^2 \\
= \ & 0.146 \times 10^2 \text{ (arr.)}
\end{aligned}
$$

$$
\begin{aligned}
& 0.1460 \times 10^2 \\
+\ & 0.0505 \times 10^2 \\
= \ & 0.1965 \times 10^2 \\
= \ & 0.197 \times 10^2 \text{ (arr.)}
\end{aligned}
$$

a) $11.4 + (3.18 + 5.05)$

$$
\begin{aligned}
& 0.318 \times 10^1 \\
+\ & 0.505 \times 10^1 \\
= \ & 0.823 \times 10^1
\end{aligned}
$$

$$
\begin{aligned}
& 0.0823 \times 10^2 \\
+\ & 0.1140 \times 10^2 \\
= \ & 0.1963 \times 10^2 \\
= \ & 0.196 \times 10^2 \text{ (arr.)}
\end{aligned}
$$

Operações aritméticas em ponto flutuante

Solução do Exemplo (cont.)

b) $5.55(4.45 - 4.35)$

$$
\begin{aligned}
&0.445 \times 10^1 \\
-\ &0.435 \times 10^1 \\
=\ &0.010 \times 10^1
\end{aligned}
$$

$$
\begin{aligned}
&0.555 \times 10^1 \\
*\ &0.100 \times 10^0 \\
=\ &0.055500 \times 10^1 \\
=\ &0.555
\end{aligned}
$$

b) $5.55 * 4.45 - 5.55 * 4.35$

$$
\begin{aligned}
&0.555 \times 10^1 \\
*\ &0.445 \times 10^1 \\
=\ &0.246975 \times 10^2 \\
=\ &0.247 \times 10^2
\end{aligned}
$$

$$
\begin{aligned}
&0.555 \times 10^1 \\
*\ &0.435 \times 10^1 \\
=\ &0.241425 \times 10^2 \\
=\ &0.241 \times 10^2
\end{aligned}
$$

$$
\begin{aligned}
&0.247 \times 10^2 \\
-\ &0.241 \times 10^2 \\
=\ &0.006 \times 10^2 \\
=\ &0.6
\end{aligned}
$$

Noções básicas sobre erros

Já vimos que introduzimos erros ao representar um número no computador.

Como vamos representar *aproximadamente* um número real x por sua versão ponto flutuante no computador, precisamos definir medidas apropriadas para calcular o erro cometido nessa aproximação.

Vamos usar as seguintes medidas de erro:

- erro absoluto
- erro relativo

Noções básicas sobre erros

Erro absoluto

Se $\tilde{x}$ é uma aproximação de x, o erro absoluto é definido por

$$EA(\tilde{x}) = x - \tilde{x}$$

Noções básicas sobre erros

Erro absoluto

Se $\tilde{x}$ é uma aproximação de x, o erro absoluto é definido por

$$EA(\tilde{x}) = x - \tilde{x}$$

Exemplo

Seja $x = 1428.756$. Em uma máquina com mantissa $t = 4$, usando arredondamento e truncamento, respectivamente, temos

$$\tilde{x_t} = 0.1428 \times 10^4 \quad \Rightarrow \quad EA(\tilde{x_t}) = 0.756 \times 10^0$$
$$\tilde{x_a} = 0.1429 \times 10^4 \quad \Rightarrow \quad EA(\tilde{x_a}) = 0.244 \times 10^0$$

Erro relativo

O erro relativo é definido por

$$ER(\tilde{x}) = \frac{x - \tilde{x}}{\tilde{x}} = \frac{EA(\tilde{x})}{\tilde{x}}$$

dado que $\tilde{x} \neq 0$.

Erro relativo

O erro relativo é definido por

$$ER(\tilde{x}) = \frac{x - \tilde{x}}{\tilde{x}} = \frac{EA(\tilde{x})}{\tilde{x}}$$

dado que $\tilde{x} \neq 0$.

Exemplo

$$\begin{aligned} x_1 = 1000.5, \quad \tilde{x_1} = 1000.6 \\ x_2 = 10.5, \quad \tilde{x_2} = 10.6 \end{aligned} \qquad \Rightarrow \quad EA(\tilde{x_i}) = 0.1, i = 1, 2$$

Noções básicas sobre erros

Erro relativo

O erro relativo é definido por

$$ER(\tilde{x}) = \frac{x - \tilde{x}}{\tilde{x}} = \frac{EA(\tilde{x})}{\tilde{x}}$$

dado que $\tilde{x} \neq 0$.

Exemplo

$$\begin{aligned} x_1 = 1000.5, \quad \tilde{x_1} = 1000.6 \\ x_2 = 10.5, \quad \tilde{x_2} = 10.6 \end{aligned} \qquad \Rightarrow \quad EA(\tilde{x_i}) = 0.1, i = 1, 2$$

$$ER(\tilde{x_1}) = \frac{0.1}{1000.6} \approx 0.00009994 = 0.9994 \times 10^{-4}$$

$$ER(\tilde{x_2}) = \frac{0.1}{10.6} \approx 0.009433 = 0.9433 \times 10^{-2}$$

Noções básicas sobre erros

Ao invés de erro relativo, as vezes o conceito de **dígitos corretos** pode ser usado. Dizemos que $\tilde{x}$ possui m dígitos (decimais) corretos com relação a x, se o erro $|x - \tilde{x}|$ possui magnitude menor ou igual a 5 no dígito de posição $(m+1)$, contando a partir do primeiro dígito não-zero de x.

$$
\begin{array}{llll}
x = 1/3, & \tilde{x} = 0.333, & EA(\tilde{x}) = 0.00033 & \Rightarrow \quad m = 3 \\
x = 23.496, & \tilde{x} = 23.494, & EA(\tilde{x}) = 0.002 & \Rightarrow \quad m = 4 \\
x = 0.02138, & \tilde{x} = 0.02144, & EA(\tilde{x}) = 0.00006 & \Rightarrow \quad m = 2
\end{array}
$$

Noções básicas sobre erros

Apesar de termos definido erro absoluto e erro relativo para valores reais x e $\tilde{x}$, a mesma definição serve para vetores ou matrizes.

Nesse caso, se denotarmos a norma de um vetor por $||\cdot||$, então teríamos:

- erro absoluto
$$||x - \tilde{x}||$$
- erro relativo
$$\frac{||x - \tilde{x}||}{||\tilde{x}||}$$

E de forma similar para matrizes. Iremos usar estes conceitos mais adiante no curso.

Noções básicas sobre erros

É claro que em geral não conhecemos o valor real que estamos interessados em aproximar, se soubessemos, não seria necessário se preocupar em aproxima-lo. O que se faz na prática é usar uma medida de erro aproximado, dada por

$$ER = \frac{x_{new} - x_{old}}{x_{new}}$$

A seguir iremos obter estimativas para o erro quando um número é aproximado em ponto flutuante em uma máquina que opera com truncamento ou arredondamento.

Erros no arredondamento e truncamento

Vamos analisar agora os erros cometidos quando a máquina opera com arredondamento ou truncamento.

Erros no arredondamento e truncamento

Vamos analisar agora os erros cometidos quando a máquina opera com arredondamento ou truncamento.

Para isso, vamos considerar um sistema que trabalha em aritmética de ponto flutuante com t dígitos na mantissa e base 10. Seja $x \in \mathbb{R}$. Vamos escrever x na forma

$$x = f_x \times 10^e + g_x \times 10^{e-t}$$

onde

$$\begin{aligned} e &\rightarrow \text{ expoente} \\ t &\rightarrow \text{ num. dígitos da mantissa} \\ 0.1 \leq f_x &< 1 \\ 0 \leq g_x &< 1 \end{aligned}$$

Erros no arredondamento e truncamento

Exemplo

Seja $t = 4$ e $x = 234.57$, então

$$x = \underbrace{0.2345 \times 10^{3}}_{f_x \times 10^{e}} + \underbrace{0.7 \times 10^{-1}}_{g_x \times 10^{e-t}}$$

Na representação de x neste sistema, a parcela $g_x \times 10^{e-t}$ não pode ser incorporada à mantissa.

□

Erros no arredondamento e truncamento

Exemplo

Seja $t = 4$ e $x = 234.57$, então

$$x = \underbrace{0.2345 \times 10^3}_{f_x \times 10^e} + \underbrace{0.7 \times 10^{-1}}_{g_x \times 10^{e-t}}$$

Na representação de x neste sistema, a parcela $g_x \times 10^{e-t}$ não pode ser incorporada à mantissa.

□

Estamos interessados em obter estimativas para o erro absoluto (ou relativo) máximo cometido quando usamos arredondamento ou truncamento.

Erro no truncamento

Em sistemas que usam truncamento a parcela $g_x \times 10^{e-t}$ é desprezada e portanto x é representado aproximadamente por $\tilde{x} = f_x \times 10^e$.

$$|EA(\tilde{x})| = |x - \tilde{x}|$$

Erro no truncamento

Em sistemas que usam truncamento a parcela $g_x \times 10^{e-t}$ é desprezada e portanto x é representado aproximadamente por $\tilde{x} = f_x \times 10^e$.

$$|EA(\tilde{x})| = |x - \tilde{x}| = |f_x \times 10^e + g_x \times 10^{e-t} - f_x \times 10^e|$$

Erro no truncamento

Em sistemas que usam truncamento a parcela $g_x \times 10^{e-t}$ é desprezada e portanto x é representado aproximadamente por $\tilde{x} = f_x \times 10^e$.

$$\begin{aligned} |EA(\tilde{x})| &= |x - \tilde{x}| = |f_x \times 10^e + g_x \times 10^{e-t} - f_x \times 10^e| \\ &= |g_x \times 10^{e-t}| \end{aligned}$$

Erro no truncamento

Em sistemas que usam truncamento a parcela $g_x \times 10^{e-t}$ é desprezada e portanto x é representado aproximadamente por $\tilde{x} = f_x \times 10^e$.

$$\begin{aligned} |EA(\tilde{x})| &= |x - \tilde{x}| = |f_x \times 10^e + g_x \times 10^{e-t} - f_x \times 10^e| \\ &= |g_x \times 10^{e-t}| \\ &< 10^{e-t}, \quad \text{pois } 0 \leq g_x < 1 \end{aligned}$$

Erro no truncamento

Em sistemas que usam truncamento a parcela $g_x \times 10^{e-t}$ é desprezada e portanto x é representado aproximadamente por $\tilde{x} = f_x \times 10^e$.

$$\begin{aligned} |EA(\tilde{x})| &= |x - \tilde{x}| = |f_x \times 10^e + g_x \times 10^{e-t} - f_x \times 10^e| \\ &= |g_x \times 10^{e-t}| \\ &< 10^{e-t}, \quad \text{pois } 0 \leq g_x < 1 \end{aligned}$$

$$|ER(\tilde{x})| = \frac{|x - \tilde{x}|}{|\tilde{x}|}$$

Erro no truncamento

Em sistemas que usam truncamento a parcela $g_x \times 10^{e-t}$ é desprezada e portanto x é representado aproximadamente por $\tilde{x} = f_x \times 10^e$.

$$\begin{aligned} |EA(\tilde{x})| &= |x - \tilde{x}| = |f_x \times 10^e + g_x \times 10^{e-t} - f_x \times 10^e| \\ &= |g_x \times 10^{e-t}| \\ &< 10^{e-t}, \quad \text{pois } 0 \leq g_x < 1 \end{aligned}$$

$$|ER(\tilde{x})| = \frac{|x - \tilde{x}|}{|\tilde{x}|} = \frac{|g_x \times 10^{e-t}|}{|f_x \times 10^e|}$$

Erro no truncamento

Em sistemas que usam truncamento a parcela $g_x \times 10^{e-t}$ é desprezada e portanto x é representado aproximadamente por $\tilde{x} = f_x \times 10^e$.

$$\begin{aligned} |EA(\tilde{x})| &= |x - \tilde{x}| = |f_x \times 10^e + g_x \times 10^{e-t} - f_x \times 10^e| \\ &= |g_x \times 10^{e-t}| \\ &< 10^{e-t}, \quad \text{pois } 0 \leq g_x < 1 \end{aligned}$$

$$\begin{aligned} |ER(\tilde{x})| &= \frac{|x - \tilde{x}|}{|\tilde{x}|} = \frac{|g_x \times 10^{e-t}|}{|f_x \times 10^e|} \\ &< \frac{10^{e-t}}{0.1 \times 10^e} \end{aligned}$$

Erro no truncamento

Em sistemas que usam truncamento a parcela $g_x \times 10^{e-t}$ é desprezada e portanto x é representado aproximadamente por $\tilde{x} = f_x \times 10^e$.

$$\begin{aligned} |EA(\tilde{x})| &= |x - \tilde{x}| = |f_x \times 10^e + g_x \times 10^{e-t} - f_x \times 10^e| \\ &= |g_x \times 10^{e-t}| \\ &< 10^{e-t}, \quad \text{pois } 0 \leq g_x < 1 \end{aligned}$$

$$\begin{aligned} |ER(\tilde{x})| &= \frac{|x - \tilde{x}|}{|\tilde{x}|} = \frac{|g_x \times 10^{e-t}|}{|f_x \times 10^e|} \\ &< \frac{10^{e-t}}{0.1 \times 10^e} = \frac{10^{e-t}}{10^{e-1}} = 10^{e-t}\ 10^{-(e-1)} = 10^{1-t} \end{aligned}$$

Erro no truncamento

Em sistemas que usam truncamento a parcela $g_x \times 10^{e-t}$ é desprezada e portanto x é representado aproximadamente por $\tilde{x} = f_x \times 10^e$.

$$\begin{aligned} |EA(\tilde{x})| &= |x - \tilde{x}| = |f_x \times 10^e + g_x \times 10^{e-t} - f_x \times 10^e| \\ &= |g_x \times 10^{e-t}| \\ &< 10^{e-t}, \quad \text{pois } 0 \leq g_x < 1 \end{aligned}$$

$$\begin{aligned} |ER(\tilde{x})| &= \frac{|x - \tilde{x}|}{|\tilde{x}|} = \frac{|g_x \times 10^{e-t}|}{|f_x \times 10^e|} \\ &< \frac{10^{e-t}}{0.1 \times 10^e} = \frac{10^{e-t}}{10^{e-1}} = 10^{e-t}\ 10^{-(e-1)} = 10^{1-t} \end{aligned}$$

Em resumo

$$\boxed{|EA(\tilde{x})| < 10^{e-t}} \qquad \boxed{|ER(\tilde{x})| < 10^{1-t}}$$

Erro no arredondamento

Em sistemas com arredondamento, f_x é modificado de forma a levar em consideração g_x. O arredondamento é feito da seguinte forma:

$$\tilde{x} = \begin{cases} f_x, & \text{se } |g_x| < 1/2 \\ f_x \times 10^e + 1 \times 10^{e-t}, & \text{se } |g_x| \geq 1/2 \end{cases}$$

ou seja, se $|g_x| < 1/2$, a parcela g_x é desprezada, caso contrário somamos 1 ao último dígito de f_x.

Erro no arredondamento

Em sistemas com arredondamento, f_x é modificado de forma a levar em consideração g_x. O arredondamento é feito da seguinte forma:

$$\tilde{x} = \begin{cases} f_x, & \text{se } |g_x| < 1/2 \\ f_x \times 10^e + 1 \times 10^{e-t}, & \text{se } |g_x| \geq 1/2 \end{cases}$$

ou seja, se $|g_x| < 1/2$, a parcela g_x é desprezada, caso contrário somamos 1 ao último dígito de f_x.

Para analisar o erro máximo cometido quando representamos um número usando ponto flutuante com arredondamento, vamos considerar os dois casos

1. $|g_x| < 1/2$
2. $|g_x| \geq 1/2$

Erro no arredondamento

Caso 1: $|g_x| < 1/2$

$$|EA(\tilde{x})| = |x - \tilde{x}|$$

Erro no arredondamento

Caso 1: $|g_x| < 1/2$

$$|EA(\tilde{x})| = |x - \tilde{x}| = |g_x \times 10^{e-t}|$$

Erro no arredondamento

Caso 1: $|g_x| < 1/2$

$$|EA(\tilde{x})| = |x - \tilde{x}| = |g_x \times 10^{e-t}| < \tfrac{1}{2} \times 10^{e-t}$$

Erro no arredondamento

Caso 1: $|g_x| < 1/2$

$$|EA(\tilde{x})| = |x - \tilde{x}| = |g_x \times 10^{e-t}| < \tfrac{1}{2} \times 10^{e-t}$$

$$|ER(\tilde{x})| = \frac{|x - \tilde{x}|}{|\tilde{x}|}$$

Erro no arredondamento

Caso 1: $|g_x| < 1/2$

$$|EA(\tilde{x})| = |x - \tilde{x}| = |g_x \times 10^{e-t}| < \tfrac{1}{2} \times 10^{e-t}$$

$$|ER(\tilde{x})| = \frac{|x - \tilde{x}|}{|\tilde{x}|}$$

$$= \frac{|g_x \times 10^{e-t}|}{|f_x \times 10^{e}|}$$

Erro no arredondamento

Caso 1: $|g_x| < 1/2$

$$|EA(\tilde{x})| = |x - \tilde{x}| = |g_x \times 10^{e-t}| < \tfrac{1}{2} \times 10^{e-t}$$

$$\begin{aligned} |ER(\tilde{x})| &= \frac{|x - \tilde{x}|}{|\tilde{x}|} \\ &= \frac{|g_x \times 10^{e-t}|}{|f_x \times 10^{e}|} \\ &< \frac{\frac{1}{2} \times 10^{e-t}}{0.1 \times 10^{e}} \end{aligned}$$

Erro no arredondamento

Caso 1: $|g_x| < 1/2$

$$|EA(\tilde{x})| = |x - \tilde{x}| = |g_x \times 10^{e-t}| < \tfrac{1}{2} \times 10^{e-t}$$

$$\begin{aligned} |ER(\tilde{x})| &= \frac{|x - \tilde{x}|}{|\tilde{x}|} \\ &= \frac{|g_x \times 10^{e-t}|}{|f_x \times 10^{e}|} \\ &< \frac{\frac{1}{2} \times 10^{e-t}}{0.1 \times 10^{e}} = \tfrac{1}{2} \times 10^{1-t} \end{aligned}$$

Erro no arredondamento

Caso 2: $|g_x| \geq 1/2$

$$|EA(\tilde{x})| = |x - \tilde{x}|$$

Erro no arredondamento

Caso 2: $|g_x| \geq 1/2$

$$|EA(\tilde{x})| = |x - \tilde{x}| = |f_x \times 10^e + g_x \times 10^{e-t} - (f_x \times 10^e + 1 \times 10^{e-t})|$$

Erro no arredondamento

Caso 2: $|g_x| \geq 1/2$

$$\begin{aligned} |EA(\tilde{x})| &= |x - \tilde{x}| = |f_x \times 10^e + g_x \times 10^{e-t} - (f_x \times 10^e + 1 \times 10^{e-t})| \\ &= |g_x \times 10^{e-t} - 1 \times 10^{e-t}| \end{aligned}$$

Erro no arredondamento

Caso 2: $|g_x| \geq 1/2$

$$
\begin{aligned}
|EA(\tilde{x})| &= |x - \tilde{x}| = |f_x \times 10^e + g_x \times 10^{e-t} - (f_x \times 10^e + 1 \times 10^{e-t})| \\
&= |g_x \times 10^{e-t} - 1 \times 10^{e-t}| \\
&= |(g_x - 1) \times 10^{e-t}|
\end{aligned}
$$

Erro no arredondamento

Caso 2: $|g_x| \geq 1/2$

$$
\begin{aligned}
|EA(\tilde{x})| &= |x - \tilde{x}| = |f_x \times 10^e + g_x \times 10^{e-t} - (f_x \times 10^e + 1 \times 10^{e-t})| \\
&= |g_x \times 10^{e-t} - 1 \times 10^{e-t}| \\
&= |(g_x - 1) \times 10^{e-t}| \\
&< \tfrac{1}{2} \times 10^{e-t}
\end{aligned}
$$

Erro no arredondamento

Caso 2: $|g_x| \geq 1/2$

$$\begin{aligned}
|EA(\tilde{x})| &= |x - \tilde{x}| = |f_x \times 10^e + g_x \times 10^{e-t} - (f_x \times 10^e + 1 \times 10^{e-t})| \\
&= |g_x \times 10^{e-t} - 1 \times 10^{e-t}| \\
&= |(g_x - 1) \times 10^{e-t}| \\
&< \tfrac{1}{2} \times 10^{e-t}
\end{aligned}$$

$$|ER(\tilde{x})| = \frac{|x - \tilde{x}|}{|\tilde{x}|} \leq \frac{\frac{1}{2} \times 10^{e-t}}{|f_x \times 10^e + 1 \times 10^{e-t}|}$$

Erro no arredondamento

Caso 2: $|g_x| \geq 1/2$

$$\begin{aligned}
|EA(\tilde{x})| &= |x - \tilde{x}| = |f_x \times 10^e + g_x \times 10^{e-t} - (f_x \times 10^e + 1 \times 10^{e-t})| \\
&= |g_x \times 10^{e-t} - 1 \times 10^{e-t}| \\
&= |(g_x - 1) \times 10^{e-t}| \\
&< \tfrac{1}{2} \times 10^{e-t}
\end{aligned}$$

$$\begin{aligned}
|ER(\tilde{x})| &= \frac{|x - \tilde{x}|}{|\tilde{x}|} \leq \frac{\frac{1}{2} \times 10^{e-t}}{|f_x \times 10^e + 1 \times 10^{e-t}|} \\
&< \frac{\frac{1}{2} \times 10^{e-t}}{0.1 \times 10^e}
\end{aligned}$$

Erro no arredondamento

Caso 2: $|g_x| \geq 1/2$

$$\begin{aligned}
|EA(\tilde{x})| &= |x - \tilde{x}| = |f_x \times 10^e + g_x \times 10^{e-t} - (f_x \times 10^e + 1 \times 10^{e-t})| \\
&= |g_x \times 10^{e-t} - 1 \times 10^{e-t}| \\
&= |(g_x - 1) \times 10^{e-t}| \\
&< \tfrac{1}{2} \times 10^{e-t}
\end{aligned}$$

$$\begin{aligned}
|ER(\tilde{x})| = \frac{|x - \tilde{x}|}{|\tilde{x}|} &\leq \frac{\frac{1}{2} \times 10^{e-t}}{|f_x \times 10^e + 1 \times 10^{e-t}|} \\
&< \frac{\frac{1}{2} \times 10^{e-t}}{0.1 \times 10^e} = \tfrac{1}{2} \times 10^{e-t} \times 10^{-(e-1)} = \tfrac{1}{2} \times 10^{1-t}
\end{aligned}$$

Erro no arredondamento

Ou seja, em ambos os casos temos

$$\boxed{|EA(\tilde{x})| < \frac{1}{2} \times 10^{e-t}} \qquad \boxed{|ER(\tilde{x})| < \frac{1}{2} \times 10^{1-t}}$$

Erro no arredondamento

Ou seja, em ambos os casos temos

$$\boxed{|EA(\tilde{x})| < \tfrac{1}{2} \times 10^{e-t}} \qquad \boxed{|ER(\tilde{x})| < \tfrac{1}{2} \times 10^{1-t}}$$

De forma geral para uma base β qualquer, seguindo o mesmo raciocínio, podemos enunciar o seguinte teorema:

Teorema

Suponha uma máquina com base β e mantissa com t dígitos. Então, qualquer número real pode ser representado no intervalo de ponto flutuante da máquina com um erro relativo que não excede o epsilon de máquina ϵ_{mach} (machine epsilon ou round-off unit), o qual é definido por

$$\epsilon_{mach} = \begin{cases} \frac{1}{2}\beta^{1-t}, & \text{se arredondamento for usado} \\ \beta^{1-t}, & \text{se truncamento for usado} \end{cases}$$

Efeitos numéricos

Além dos erros causados pela representação no computador e pelas operações aritméticas, existem certos efeitos numéricos que contribuem para aumentar os erros introduzidos.

A seguir iremos estudar alguns casos importantes como a adição ou subtração entre um número grande e um pequeno, subtração de dois números quase iguais, propagação do erro, etc.

Efeitos numéricos

Somar ou subtrair um número pequeno e um grande

Para exemplificar considere um sistema $F(10, 4, L, U)$.

Exemplo

Somar 0.1 e 5000.

$$\begin{aligned} 0.1 + 5000 = & \ 0.1000 \times 10^0 + 0.5000 \times 10^4 \\ = & \ 0.00001 \times 10^4 \\ + & \ 0.50000 \times 10^4 \\ = & \ 0.50001 \times 10^4 \end{aligned}$$

Usando arredondamento (ou truncamento), obtemos 0.5×10^4.

□

Efeitos numéricos

Exemplo

Calcular

$$S = 5000 + \underbrace{0.1 + \ldots 0.1}_{10\times} = 5000$$

$$S = \underbrace{0.1 + \ldots 0.1}_{10\times} + 5000 = 5001$$

Observe que embora analiticamente o resultado das somas seja o mesmo, quando usamos uma máquina de ponto flutuante $F(10, 4, L, U)$, o resultado é diferente.

No primeiro caso, ao somar um número grande 5000 e um pequeno 0.1 cometemos um erro que se propaga em toda a soma de S. No segundo caso, evitamos este problema.

□

Efeitos numéricos

Cancelamento

O **cancelamento** ocorre quando subtraimos dois números quase iguais, ou quando somamos números de sinais opostos mas de magnitudes semelhantes.

Como vimos, quando calculamos a diferença $x - y$, o resultado terá o mesmo expoente e. Para normalizar o resultado obtido, devemos mover os dígitos para a esquerda de tal forma que o primeiro seja diferente de zero.

Desta forma, uma quantidade de dígitos iguais a zero aparece no final da mantissa do número normalizado. Estes zeros não possuem significado algum, e dizemos que ocorreu a perda de dígitos significativos.

Efeitos numéricos

Cancelamento

Exercício 1 - (Em sala)

Calcular $\sqrt{37} - \sqrt{36}$ em uma máquina $F(10, 4, L, U)$ com **arredondamento**.

- Calcule o valor exato e compare o que acontece.
- Em seguida use $\frac{x-y}{\sqrt{x}-\sqrt{y}}$ para calcular o valor de $\sqrt{37} - \sqrt{36}$.

Exercício 2 - (Em sala)

Resolva: $x^2 - 1634x + 2 = 0$ usando um sistema $F(10, 10, -10, 10)$.

- Compare as raízes obtidas com os valores exatos.
- Em seguida utilize $x_2 = \frac{2c}{-b+\sqrt{b^2-4ac}}$ para calcular a raiz.

Solução do Exercício 1

Para efeitos de comparação, apresentamos a resposta exata aqui:

$$\sqrt{37} - \sqrt{36} = 6.08276253 - 6 = 0.08276253 = 0.8277 \times 10^{-1}$$

Nessa máquina temos

$$\begin{aligned} \sqrt{37} = 6.08276253 \quad &\rightarrow \quad 0.6083 \times 10^{1} \\ \sqrt{36} = 6.0 \quad &\rightarrow \quad 0.6000 \times 10^{1} \end{aligned}$$

Efetuando a subtração tem-se

$$\begin{aligned} & 0.6083 \times 10^{1} \\ - \; & 0.6000 \times 10^{1} \\ = \; & 0.0083 \times 10^{1} = 0.8\mathbf{300} \times 10^{-1} \end{aligned}$$

$\rightarrow$ Perda de dígitos significativos!

Solução do Exercício 1 - (cont.)

É possível obter um resultado mais preciso? Sim, basta considerar que

$$\sqrt{x}-\sqrt{y}=\sqrt{x}-\sqrt{y}\frac{(\sqrt{x}+\sqrt{y})}{\sqrt{x}+\sqrt{y}}=\frac{x-y}{\sqrt{x}+\sqrt{y}}$$

Portanto para

$$\sqrt{37}-\sqrt{36}$$

temos

$$\begin{aligned}\frac{x-y}{\sqrt{x}+\sqrt{y}}=\frac{37-36}{\sqrt{37}+\sqrt{36}}&=\frac{1}{0.6083\times 10^1+0.6000\times 10^1}\\&=\frac{1}{0.1208\times 10^2}\\&=0.08278145=0.8278\times 10^{-1}\end{aligned}$$

que é um resultado mais preciso que o anterior. □

Solução do Exercício 2

Para $b^2 >> 4ac$ temos problema ao usar a fórmula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Usando a fórmula temos

$$x = \frac{1634 \pm \sqrt{(1634)^2 - 4(2)}}{2} = 817 \pm \sqrt{667487}$$

Assim obtemos as raízes:

$$x_1 = 817 + 816.9987760 = 0.1633998776 \times 10^4$$
$$x_2 = 817 - 816.9987760 = 0.1224000000 \times 10^{-2}$$

O valor exato da segunda raiz é $x_2 = 0.001223991124 9$. Podemos obter um resultado melhor? Sim, veremos algumas opções.

Solução do Exercício 2 - (cont.)

$$x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \frac{(-b + \sqrt{b^2 - 4ac})}{(-b + \sqrt{b^2 - 4ac})}$$
$$= \frac{b^2 - (b^2 - 4ac)}{2a(-b + \sqrt{b^2 - 4ac})} = \frac{4ac}{2a(-b + \sqrt{b^2 - 4ac})}$$
$$= \frac{2c}{-b + \sqrt{b^2 - 4ac}}$$

E assim temos

$$x_2 = \frac{2(2)}{1634 + \sqrt{(1634)^2 - 4(2)}}$$
$$= \frac{4}{1634 + \sqrt{2669948}}$$
$$= 0.001223991125 = 0.1223991125 \times 10^{-2}$$

Solução do Exercício 2 - (cont.)

Veja o que acontece se usarmos essa fórmula para x_1

$$x_1 = \frac{2c}{-b - \sqrt{b^2 - 4ac}} \quad \Rightarrow \quad x_1 = 1634.000278$$

Uma boa estratégia consiste em analisar o sinal de b para calcular uma daz raízes (neste caso x_1) sem problemas com cancelamento e depois usar a seguinte relação para obter a outra raiz: $x_1 x_2 = \frac{c}{a}$.

De forma geral podemos usar

$$\boxed{x_1 = \frac{-\left(b + \text{sign}(b)\sqrt{b^2 - 4ac}\right)}{2a}} \qquad \boxed{x_1 x_2 = \frac{c}{a}}$$

Solução do Exercício 2 - (cont.)

Neste caso teríamos

$$x_1 x_2 = \frac{c}{a}$$
$$x_1 x_2 = 2$$
$$x_2 = \frac{2}{1633.998776} = 0.001223991125$$

□

Exemplo

Calcular $f(x) = 1 - \cos(x)$ para x próximo de zero. Considere uma máquina com $t = 10$ dígitos na mantissa e base $\beta = 10$.

Exemplo

Calcular $f(x) = 1 - \cos(x)$ para x próximo de zero. Considere uma máquina com $t = 10$ dígitos na mantissa e base $\beta = 10$.

Solução do Exemplo

Como $\cos(x) \approx 1$ para x próximo de zero, haverá perda de dígitos significativos se primeiro calcularmos $\cos(x)$ e depois realizarmos a subtração, devido a subtração de dois números quase iguais. Veja.

Exemplo

Calcular $f(x) = 1 - \cos(x)$ para x próximo de zero. Considere uma máquina com $t = 10$ dígitos na mantissa e base $\beta = 10$.

Solução do Exemplo

Como $\cos(x) \approx 1$ para x próximo de zero, haverá perda de dígitos significativos se primeiro calcularmos $\cos(x)$ e depois realizarmos a subtração, devido a subtração de dois números quase iguais. Veja.

Seja $x = 0.01$, então

$$y = \cos(x) = 0.99995000041 66\ldots$$

Nesta máquina y é representado por $\tilde{y}$

$$\tilde{y} = 0.9999500004 \times 10^0$$

Solução do Exemplo - (cont.)

O resultado aproximado da avaliação da função é

$$\begin{aligned}\tilde{f}(0.01) &= 1 - 0.9999500004 \times 10^0 \\ &= \ 0.1000000000 \times 10^1 \\ &- \ 0.0999950000 \times 10^1 \\ &= \ 0.0000050000 \times 10^1 = 0.5000000000 \times 10^{-4}\end{aligned}$$

enquanto o resultado exato é

$$f(0.01) = 0.4999958333 \times 10^{-4}$$

o que confirma a perda de dígitos significativos.

Solução do Exemplo - (cont.)

Podemos evitar esta perda, usando neste caso uma fórmula alternativa baseada em relações trigonométricas:

$$1 - \cos x \, \frac{1 + \cos x}{1 + \cos x} = \frac{1 - \cos^2 x}{1 + \cos x} = \frac{\sin^2 x}{1 + \cos x}$$

Solução do Exemplo - (cont.)

Podemos evitar esta perda, usando neste caso uma fórmula alternativa baseada em relações trigonométricas:

$$1 - \cos x \, \frac{1 + \cos x}{1 + \cos x} = \frac{1 - \cos^2 x}{1 + \cos x} = \frac{\sin^2 x}{1 + \cos x}$$

ou ainda, como visto anteriormente, podemos usar aproximação por série de Taylor em torno do ponto $a = 0$ para $\cos x$, isto é

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + R(x)$$

Solução do Exemplo - (cont.)

Podemos evitar esta perda, usando neste caso uma fórmula alternativa baseada em relações trigonométricas:

$$1 - \cos x \, \frac{1 + \cos x}{1 + \cos x} = \frac{1 - \cos^2 x}{1 + \cos x} = \frac{\sin^2 x}{1 + \cos x}$$

ou ainda, como visto anteriormente, podemos usar aproximação por série de Taylor em torno do ponto $a = 0$ para $\cos x$, isto é

$$\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + R(x)$$

resultando na seguinte função

$$f(x) = 1 - \cos(x) = \frac{x^2}{2!} - \frac{x^4}{4!} + \frac{x^6}{6!}$$

cuja avaliação em $x = 0.01$ fornece uma resposta mais precisa. □

Efeitos numéricos

Propagação do erro

Também ocorrem problemas quando uma soma parcial é muito grande se comparada com o resultado final.

Considere que:

$$s = \sum_{k=1}^{n} a_k$$

seja a soma a ser computada e que os a_k podem ser positivos ou negativos e de diferentes magnitudes.

O cálculo pode ser feito da seguinte forma:

$$s_1 = a_k, \quad s_k = s_{k-1} + a_k, \quad k = 2, 3, \ldots, n$$

tal que $s = s_n$.

Exemplo

Considere uma máquina $F(10, 4, L, U)$ com truncamento. Vamos efetuar a seguinte operação:

$$S = \sum_{i=1}^{4}(x_i + y_i) \quad \text{com} \quad x_i = 0.46709, \quad \text{e} \quad y_i = 3.5678$$

Exemplo

Considere uma máquina $F(10, 4, L, U)$ com truncamento. Vamos efetuar a seguinte operação:

$$S = \sum_{i=1}^{4}(x_i + y_i) \quad \text{com} \quad x_i = 0.46709, \quad \text{e} \quad y_i = 3.5678$$

Para $i = 1$, temos

$$(x_1 + y_1) = 0.4034 \times 10^1$$

E o erro absoluto é

$$EA(\bar{S}) = |4.03569 - 4.034| = 0.00169$$

Exemplo (cont.)

Para $i = 2$, temos

$$(x_1 + y_1) + (x_2 + y_2) = 0.8068 \times 10^1$$
$$EA(\bar{S}) = |8.07138 - 8.068| = 0.00338$$

Para $i = 3$, temos

$$(x_1 + y_1) + (x_2 + y_2) + (x_3 + y_3) = 0.1210 \times 10^2$$
$$EA(\bar{S}) = |12.10707 - 12.10| = 0.00707$$

Para $i = 4$, temos

$$(x_1 + y_1) + (x_2 + y_2) + (x_3 + y_3) + (x_4 + y_4) = 0.1613 \times 10^2$$
$$EA(\bar{S}) = |16.14267 - 16.13| = 0.01276$$

De onde pode-se observar que o erro absoluto aumenta à medida em que as operações aritméticas são realizadas. □

Exemplo

Ao aproximar a função e^x por uma série de Taylor em torno do ponto $a = 0$, temos

$$p(x) = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \ldots$$

Se tentarmos avaliar $p(x)$ para valores negativos de x como por exemplo -20, -25, -30, com muitos termos para obter boa precisão o resultado obtido estará comprometido devido a propagação do erro que acontece.

Para ilustrar o problema apresentamos uma pequena função implementada na linguagem Python para calcular $p(x)$. Comparamos o valor de $p(x)$ e da função $\exp(x)$ da linguagem Python para diferentes valores de x.

Efeitos numéricos

Propagação do erro

```
def exp_taylor(n,x):
  fat = 1.0
  term = 1.0
  soma = term
  i = 1
  while(i<=n):
     fat = fat * i
     term = term * x
     soma = soma + (term/fat)
     i = i + 1
  return soma
```

x	e^x	exp_taylor(n,x)
10	2.202647e+04	2.202646e+04
-5	6.737947e-02	6.737947e-02
-10	4.539993e-05	9.703415e-04
-20	2.061153e-09	1.599694e+06
-30	9.357622e-14	3.848426e+11

Efeitos numéricos

Exemplo

O problema ocorre pois, por exemplo para $x = -25$ os termos $\frac{25^{24}}{24!}$ e $-\frac{25^{25}}{25!}$ são muito próximos e portanto sofrem **cancelamento**.

Efeitos numéricos

Exemplo

O problema ocorre pois, por exemplo para $x = -25$ os termos $\frac{25^{24}}{24!}$ e $-\frac{25^{25}}{25!}$ são muito próximos e portanto sofrem **cancelamento**.

```
...
23  termo  5.057240e+09  soma -2.394348e+09
24  termo -5.497000e+09  soma  2.662893e+09
25  termo  5.726042e+09  soma -2.834108e+09
26  termo -5.726042e+09  soma  2.891934e+09
27  termo  5.505810e+09  soma -2.834108e+09
28  termo -5.097972e+09  soma  2.671702e+09
29  termo  4.551761e+09  soma -2.426270e+09
30  termo -3.923932e+09  soma  2.125491e+09
...
```

Efeitos numéricos

Exemplo

Nesse caso, uma estratégia simples pode ser adotada para evitar o cancelamento. Para valores negativos de x, basta calcular normalmente e^x e depois retornar $\frac{1}{e^x}$ como resultado.

```
def exp_taylor(n,x):
  fat, term = 1.0, 1.0
  soma, i = term, 1

  if x<0:
    x = - x
    neg = True

  while(i<=n):
     fat = fat * i
     term = term * x
     soma = soma + (term/fat)
     i = i + 1

  if neg: return 1.0/soma
  else:   return soma
```

□

Desastres

A implementação incorreta ou o uso incorreto de algoritmos e/ou softwares científicos já foi responsável por alguns desastres reais.

Patriot missile failure - Guerra do Golfo (1991)

- Uma bateria de mísseis Patriot ("Phased Array TRacking Intercept Of Target") americano, falhou ao rastrear e interceptar um míssil Scud do Iraque.
- O míssil Scud acertou o acampamento americano, matou 28 soldados e feriu centenas.
- Relatório técnico apontou uma falha no software.
- Palavra do computador 24 bits.
- Tempo era medido em décimos de segundo $(1/10)$.
- O valor $(1/10)$ ao ser representado em binário não termina.
- Acúmulo do erro no software após longo tempo do sistema rodando levou a falha.

Ariane 5

- Foguete da European Space Agency explode 40s após o lançamento.
- Milhões de dolares foram investidos no seu desenvolvimento e equipamento.
- Relatório acusou um erro do software no sistema de referência inercial.
- Problema: número de 64 bits de ponto flutuante era convertido em um inteiro de 16 bits com sinal. Falha na conversão para números maiores que 32767, que é o maior inteiro representável com 16 bits.

Sleipnir offshore

- Plataforma de petróleo Sleipnir A afunda.
- Após o acidente a empresa da plataforma, Statoil, uma empresa norueguesa solicita a empresa SINTEF um relatório técnico.
- Falha em uma parede, resultando em uma rachadura e vazamento. A falha aconteceu como resultado de uma combinação de erros no programa de análise de elementos finitos, que subestimou a tensão na parede.