# Padrão IEEE 754-2008

## Graduação/Pós-graduação em Engenharia Elétrica

Erivelton Geraldo Nepomuceno

Departamento de Engenharia Elétrica
Universidade Federal de São João del-Rei

21 de Setembro de 2018

# Introdução

- Computação numérica é uma parte vital da infraestrutura tecnológica e científica da atualidade.
- Praticamente toda computação numérica utiliza aritmética de ponto flutuante.
- Quase todos os computadores fazem uso do padrão IEEE para aritmética de ponto flutuante.
- Entretanto, percebe-se que aspectos importantes do padrão IEEE ainda não são compreendidos por vários estudantes e profissionais.
- Computação numérica significa computação com números.
- É uma área tão antiga quanto a própria civilização humana.
- Em torno de 1650, os egípcios já empregava técnicas de computação.
- Contar pedras e gravetos foi utilizado há anos para contar e armazenar números.
- O ábaco foi utilizado na Europa até a introdução da notação posicional decimal.

# Introdução

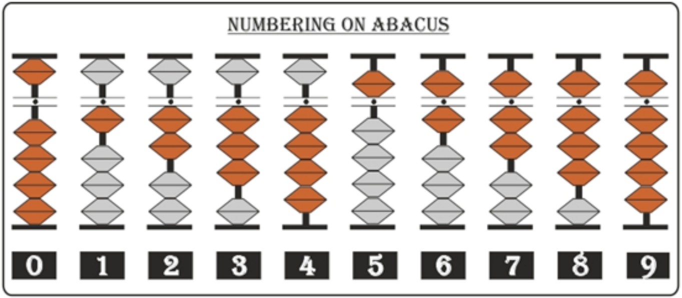


Figura 1: Primeira calculadora utilizada pelo homem: um ábaco representando o número 6302715408. Fonte: Wikipédia.

# Introdução

- A partir do séc. XVI, o sistema decimal se tornou base em toda a Europa.
- O próximo grande avanço foi a tabulação de logaritmos por John Napier no início do séc. XVII.
- Com logaritmos é possível substituir divisões e multiplicações por subtrações e adições.
- Isaac Newton e Leibniz desenvolveram o cálculo no séc. XVII e técnicas numéricas para a solução de vários problemas.
- Outros grandes matemáticos, tais como Euler, Lagrange, Gauss foram responsáveis por grandes desenvolvimentos na computação numérica.
- Um outro dispositivo utilizado foi a régua de deslizamento até a década de 70 do século passado.
- Dispositivos mecânicos para cálculo foram inventados por Schickard, Pascal e Leibniz.
- Charles Babbage iniciou o desenvolvimento de equipamentos sem intervenção humana.

# Introdução

- Durante Segunda Guerra Mundial houve um grande desenvolvimento de dispositivos para cálculos, e pode-se afirmar que mais ou menos nessa época começou-se a era da computação.
- Uma das primeiras máquinas consideradas como computador foi o Z3, construído pelo engenheiro Konrad Zuse na Alemanha entre os anos de 1939 e 1941. O Z3 usava dispositivos eletromecânicos e já empregava números binários de ponto flutuante.
- O governo britânico desenvolveu nessa mesma época um dispositivo eletrônico chamado Colossus usado para decodificar mensagens secretas.

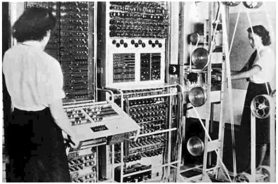

Figura 2: Colossus Mark 2 desenvolvido em 1944. Fonte: Wikipédia.

# Introdução

- Considera-se como primeiro computador eletrônico o ENIAC (*Electronic Numerical Integrator And Computer*). É um dispositivo de cerca de 18000 válvulas e foi construído na Universidade da Pensilvânia entre os anos de 1943 e 1945 por Eckert e Mauchly.

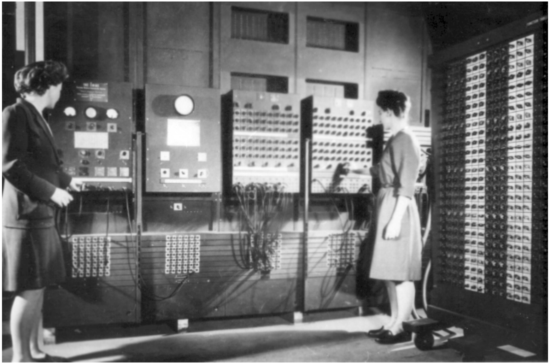

Figura 3: Eniac desenvolvido em 1946. Fonte: Wikipédia.

# Introdução

- Os dois principais cientistas que influenciaram os padrões de desenvolvimento dos dispositivos computacionais foram Alan Turing e John von Neumann.

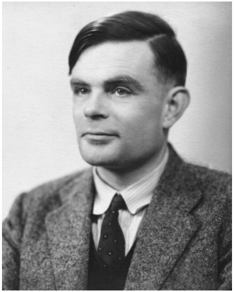

Figura 4: Alan Turing.
Fonte: Wikipédia.

Figura 5: John von Newmann.
Fonte: Wikipédia.

# Introdução

- Durante a década de 1950, o principal uso dos computadores foi para computação numérica.
- A partir de 1960, os computadores passaram a ser usados em empresas para processar informação, tais como, texto e imagem.
- Usuários frequentemente não estão cientes de que a manipulação de texto, som ou imagem envolve computação numérica.
- Os computadores são usados para resolver equações que modelam os mais diferentes sistemas: da expansão do universo à micro-estrutura do átomo; processamento de imagens e análise estatística de dados médicos; predição de clima; simulação de circuitos para projetos de computadores menores e mais rápidos; modelagem de aeronaves para testes e treinamento de pilotos; confiabilidade de sistemas elétricos.
- Os resultados numéricos são comparados com os resultados experimentais.
- Em síntese: todas áreas da ciência e engenharia utilizam fortemente a computação numérica.

# Os números reais

- Os números reais são representados por uma linha.

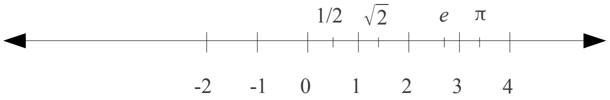


Figura 6: A linha que representa os números reais.

- A linha se estende infinitamente em direção a $-\infty$ e $+\infty$.
- Os símbolos $-\infty$ e $+\infty$ não são considerados como números.

**Definition 1**

O sistema de números real estendido consiste do campo real $R$ e dois símbolos: $+\infty$ e $-\infty$. Preserva-se a ordem original em $R$, e define-se

$$-\infty < x < +\infty$$

para qualquer $x \in R$. Utiliza-se o símbolo $\bar{R}$.

# Os números reais

- Há infinitos mas contáveis números inteiros $0{,}1,-1{,}2,-2,\ldots$.
- Os números racionais $(Q)$ são aqueles que consistem da razão de dois inteiros, tais como: $1/2, 2/3, 6/3$.
- Os números racionais são infinitos mas contáveis.

**Exercício 1**

Mostre que os números racionais são contáveis. Dica: utilize uma tabela e faça uso da diagonal.

- Os números irracionais são os números reais que não são racionais. Exemplos: $\sqrt{2}, \pi, e$.

**Exemplo 1**

O número $e$ é o limite de

$$\left(1+\frac{1}{n}\right)^n$$

quando $n \to \infty$.

# Os números reais

- As investigações para a definição de $e$ começaram no séc. XVII.
- Todo número irracional pode ser definido como o limite de uma sequência de números racionais.
- O conjunto de números irracionais é dito ser incontável.
- Número romano: MDCCCCLXXXV $= 1985$.
- O sistema posicional faz uso de um aspecto essencial: o zero é representado por um símbolo.
- Os babilônios em 300 a.C. usavam um símbolo para representar o zero.
- O sistema arábico foi desenvolvido na Índia por volta de 600 d.C.
- Após o ano de 1200 iniciou-se o uso dos números arábicos, notadamente devido a obra "Liber Abaci" (ou Livro do Cálculo) escrito pelo matemático italiano Leonardo Pisano Bigollo, mais conhecido como Fibonacci.

**Exercício 2**

Mostre que $\sqrt{2}$ é um número irracional.

# Os números reais

- O sistema decimal requer 10 símbolos (0 a 10). A escolha é pautada em função do número de dedos.
- Os babilônios tinham um outro sistema em base 60.
- O zero é necessário para distinguir 601 de 61.
- Sistema decimal foi utilizado inicialmente apenas para números inteiros.
- Embora o sistema decimal é conveniente para pessoas, o mesmo não acontece para computadores.
- O sistema binário é mais útil, no qual cada número é representado por uma palavra de bits.
- Cada bit corresponde a uma potência de 2.
- Um bit pode ser visto como um entidade física que está ligado ou desligado. Em eletrônica sabemos que o bit é representado por um nível de tensão baixo ou alto.
- Bits são organizados em grupos de 8, chamado de *byte*.
- Cada byte pode representar $2^8 = 256$ diferentes números.

# Os números reais

- O número $(71)_{10} = 7 \times 10 + 1$ tem sua representação binária como $(1000111)_2 = 1 \times 64 + 0 \times 32 + 0 \times 16 + 0 \times 8 + 1 \times 4 + 1 \times 2 + 1 \times 1.$
- O número fracionário pode ser representado, tal como
$$\frac{11}{2} = (5{,}5)_{10} = 5 \times 1 + 5 \times \frac{1}{10}$$
e
$$\frac{11}{2} = (101{,}1)_2 = 1 \times 4 + 0 \times 2 + 1 \times 1 + 1 \times \frac{1}{2}.$$
- Números irracionais possuem expansão decimal e binária infinita e sem repetição.

## Exercício 3

Faça a transformação de $\frac{1}{10}$ para o sistema binário.

## Exercício 4

Faça a expansão decimal e binária dos seguintes números: $\sqrt{2}, \pi, e$.

# Representação Computacional de Números

- Qual é a melhor maneira de representar números em um computador?
- Inteiros são representados por 32 bits. O inteiro 71 seria armazenado como

$$00000000000000000000000001000111.$$

- Faixa: 0 a $2^{32} - 1$ ou 0 a 4294967295.
- O número $2^{32}$ não é possível de ser representado.

**Exercício 5**

Qual é o número mínimo de bits necessários para representar o número 50000?

# Representação Computacional de Números

- É necessário representar números positivos e negativos.
- A ideia mais simples é representar o número com duas partes sinal e módulo.
- Neste caso, utiliza-se 1 bit para o sinal e 31 bits para para armazenar o módulo do número.
- Entretanto, o método mais comum é o complemento de 2.
- Seja $x$ tal que $0 \leq x \leq 2^{31} - 1$ é representado em sua forma binária.
- Já o o valor negativo $-y$ tal que $-1 \leq y \leq 2^{31}$ é armazenado na representação binária do inteiro positivo

$$2^{32} - y.$$

**Exercício 6**

Coloque o número 71 na forma binária de complemento de 2.

# Representação Computacional de Números

- Em um sistema de 32 bits, se dois números positivos forem adicionados e o resultado for maior que $2^{31} - 1$ ocorre o chamado integer overflow.
- Subtração de dois números inteiros representados na representação do complemento de 2 não necessita de hardware adicional.

# Representação Computacional de Números

- Números racionais podem ser representados por pares de inteiros: o numerador e denominador.
- Esta representação é precisa, mas é inconveniente do ponto de vista aritmético.
- Sistemas que representam os números racionais dessa forma tem sido chamados de simbólicos.
- Para a maioria dos casos, os números reais, entretanto, são armazenados usando representação binária.
- Há dois métodos: ponto fixo e ponto flutuante.
- Ponto fixo: 1 bit para o sinal, um grupo de bits para representar o número antes do ponto binário e um grupo de bits para representar o número após o ponto binário.

### Exemplo 2

Para uma precisão de 32 bits o número 11/2 pode ser representado como

$$|0|000000000000101|1000000000000000|.$$

# Representação Computacional de Números

## Exercício 7

Represente o número $1/10$ na representação binária de ponto fixo com 32 bits.

- Faixa: aproximadamente de $2^{-16}$ a $2^{15}$.
- O ponto fixo é bastante limitado quanto a faixa que se pode armazenar.
- É atualmente pouco usado para computação numérica.
- Entretanto microcontroladores com ponto fixo são mais econômicos, possuem circuitos internos mais simples e necessitam de menos memória [1].

[1] Anoop, C. V. and Betta, C. *Comparative Study of Fixed-Point and Floating Code for a Fixed-Point Micro*. dSPACE User Conference - India, 2012.

# Representação Computacional de Números

- Ponto flutuante é baseado na notação exponencial ou científica.
- Um número $x$ é representado por

$$x = \pm S \times 10^{E}, \quad \text{em que} \quad 1 \leq S < 10,$$

- em que $E$ é um inteiro. Os números $S$ e $E$ são chamados de significante ou mantissa e expoente, respectivamente.

**Exemplo 3**

A representação exponencial de 0,00036525 é $3{,}6525 \times 10^{-4}$.

- O ponto (vírgula) decimal flutua para a posição imediatamente posterior ao primeiro dígito não nulo. Está é a razão para o nome ponto flutuante.
- No computador, utiliza-se a base 2. Assim $x$ é escrito como

$$x = \pm S \times 2^{E}, \quad \text{em que} \quad 1 \leq S < 2. \tag{1}$$

# Representação Computacional de Números

- A expansão binária do significante é

$$S = (b_0 b_1 b_2 b_3 \ldots) \quad \text{com} \quad b_0 = 1. \tag{2}$$

**Exercício 8**

O número $11/2$ é representado como

$$\frac{11}{2} = (1{,}011)_2 \times 2^2.$$

- Os bits após o ponto binário são chamados de parte fracionária do significando.
- As Eq. (1) e (2) são representações normalizadas de $x$ e o processo de obtenção desta representação chama-se normalização.
- Para representar um número normalizado, a sua representação binária é dividida em três partes: sinal, expoente $E$ e o significante $S$, nesta ordem.

# Representação Computacional de Números

## Exemplo 4

Um sistema de 32 bits pode ser dividido nos seguintes campos: 1 bit para o sinal, 8 para para o expoente e 23 bits para o significante.

- Sinal: 0 para positivo e 1 para negativo.
- $E$ pode ter os valores entre -128 e 127 (usando complemento de 2).
- $S$ utiliza 23 bits para armazenar o valor após o ponto.

- Não é necessário armazenar $b_0$ (bit escondido).
- Se $x \in R$ pode ser armazenado exatamente em um computador então $x$ é chamado de número ponto flutuante. Senão, $x$ deve ser arredondado.

## Exemplo 5

O número 71 é representado por $(1{,}000111)_2 \times 2^6$ e é armazenado

$$|0|\ \mathrm{ebits}(6)\ |00011100000000000000000|.$$

# Representação Computacional de Números

- ebits(6) representa a conversão de 6 para o número binário no expoente.
- Se um número $x$ não tem uma expansão binária finita, é necessário terminar esta expansão. Esse processo é chamado de truncamento.

## Exemplo 6

Considere o número 1/10, cuja expansão é

$$\frac{1}{10} = (0{,}0001100110011\ldots)_2.$$

Primeiro devemos normalizar e em seguida truncar. Assim a representação de 1/10 é

$$|0|\ \mathrm{ebits}(-4)\ |10011001100110011001100|.$$

# Representação Computacional de Números

- A precisão ($p$) de um sistema de ponto flutuante é o número de bits do significante (incluindo o bit escondido).
- No sistema de 32 bits, $p = 24$, sendo 23 bits no significante e 1 bit escondido.
- Um ponto flutuante normalizado com precisão $p$ é expresso por
$$x = \pm(1{,}b_1 b_2 \dots b_{p-2} b_{p-1})_2 \times 2^E. \tag{3}$$
- O menor ponto flutuante $x$ que é maior que 1 é
$$(1{,}000\dots 1)_2 = 1 + 2^{-(p-1)}.$$
- Dá-se um especial nome machine epsilon (epsilon da máquina) a distância entre este número e o número 1:
$$\varepsilon = (0{,}000\dots 01)_2 = 2^{-(p-1)}. \tag{4}$$
- De modo mais geral, para um ponto flutuante $x$ dado por (2) nós definimos
$$\mathrm{ulp}(x) = (0{,}00\dots 01)_2 \times 2^E = 2^{-(p-1)} \times 2^E = \varepsilon \times 2^E. \tag{5}$$

# Representação Computacional de Números

- Ulp é abreviação de unit in the last place ou unidade da última posição.
- Se $x > 0$ então $\text{ulp}(x)$ é a distância entre $x$ e o próximo maior ponto flutuante maior.
- Se $x < 0$ então $\text{ulp}(x)$ é a distância entre $x$ e o próximo menor ponto flutuante.

### Exercício 9

Seja a precisão $p = 24$ tal que $\varepsilon = 2^{-23}$. Determine $\text{ulp}(x)$ para $x$ igual aos seguintes valores: a) 0,25; b) 2; c) 3; d) 4; e) 10; f) 100; g) 1030. Dê sua resposta em potência de 2.

- Até o momento, foi discutido apenas números não nulos.
- O zero não pode ser representado com o uso do bit escondido. $0000\ldots$ representa 1.
- Até 1975, resolvia esta questão não utilizando o bit escondido.
- O padrão IEEE reduz em 1 o expoente e utiliza um carácter especial para identificar o zero.

# Representação Computacional de Números

- Considere o seguinte sistema binário fictício em que todos os números podem ser representados da seguinte forma

$$\pm(b_0 b_1 b_2)_2 \times 2^E). \tag{6}$$

- $b_0$ é permitido ser 0 se $b_1$ e $b_2$ forem também zero. Neste caso, o número decimal representado é o zero.
- O número $E$ pode ser $-1$, 0 ou 1.
- O conjunto de número representados pode ser visto na Figura


Figura 7: Conjunto de números representados por (6). Fonte: Livro (Overton, 2001, p. 15).

# Representação Computacional de Números

- A precisão de (6) é $p = 3$.
- O maior número é $(1{,}11)_2 \times 2^1 = (3{,}5)_{10}$.
- O menor número positivo é $(1{,}00)_2 \times 2^{-1} = (0{,}5)_{10}$.
- O ponto flutuante seguinte ao número 1 é 1,25, assim $\varepsilon = 0{,}25$.
- A distância entre cada número é dada por

$$\mathrm{ulp}(x) = \varepsilon \times 2^E.$$

- A distância entre 0 e $\pm 0{,}5$ é maior do que a distância entre $\pm 0{,}5$ e $\pm 1$. Essa distância pode ser reduzida com a introdução dos números subnormalizados.
- Von Neumann foi inicialmente contra ao ponto flutuante.
- É possível escolher microprocessadores com ponto fixo ou ponto flutuante.

# Representação IEEE

- Ponto flutuante é usado desde meados da década de 1950.
- Durante as duas décadas seguintes, cada empresa adotava um padrão diferente.
- A IBM, líder do mercado durante as décadas de 1960 e 1970, adotou um padrão IBM 360/370, de 24 bits e hexadecimal.
- A mantissa era armazenada em 24 bits, para ser interpretado como 6 dígitos hexadecimais.
- 1 dígito era para o sinal, 7 para o expoente e 16 para a potência de 16.
- A normalização era obrigatória apenas para o primeiro dígito hexadecimal que devia ser zero.
- Como consequência, alguns números eram menos precisos que outros.

# Representação IEEE

- Em uma colaboração entre pesquisadores e projetistas de microprocessadores, um padrão para representação de ponto flutuante e aritmética foi desenvolvido nas décadas de 1970 e 1980.
- O grupo de cientistas e engenheiros foi liderado pelo *Institute for Electrical Engineering and Electronics Engineers - IEEE*. O grupo ficou conhecido como IEEE p754.
- A comunidade científica foi liderada por William Kahan.

Figura 8: William Morton Kahan (1933), matemático, professor da Universidade da Califórnia em Berkeley. Fonte: Wikipédia.

# Representação IEEE

- Os participantes industriais foram: Apple, Digital Equipment Corporation, Intel, Hewlett-Packard, Motorola e National Semiconductor.
- Kahan recebeu o prêmio Alan Turin em 1989 da ACM.
- O fundador do Matlab, Cleve Moler, chegou a mencionar que as duas grandes atrações turísticas dos Estados Unidos eram o Grand Canyon e os meetings of IEEE p754.

Figura 9: Cleve Moler (1939), matemático, criador do Matlab e fundador da empresa Mathworks. Fonte: Mathworks.

# Representação IEEE

- O padrão IEEE para ponto flutuante binário e aritmética foi publicada em 1985, sendo denominada oficialmente como ANSI/IEEE Std 754-1985.
- Em 1989, ela recebeu reconhecimento internacional como IEC 559.
- Em 2008, uma versão atualizada foi publicada, sendo denominada IEEE 754-2008.
- Em 2011, o padrão internacional ISO/IEC/IEEE 60559:2011 (com conteúdo idêntico ao IEEE 754) foi aprovado para uso.
- O padrão IEEE 754-2008 trata dos seguintes tópicos:
  - Formatos para aritmética;
  - Formatos para troca de dados;
  - Regras para arredondamento;
  - Operações aritméticas;
  - Tratamento de exceções.

# Representação IEEE

O padrão IEEE possui três aspectos essenciais:

1. Representação consistente em todas as máquinas que adotam o padrão;
2. Operação de arredondamento adequado para o ponto flutuante;
3. Tratamento consistente de exceções, tais como a divisão por zero.

- O primeiro dígito do padrão IEEE é escondido. Isso exige uma representação especial para o zero.
- Uma outra representação especial exigida é o número $\infty$[2].
- Há também a necessidade de representações especiais para $0, -0, -\infty$. Há também o símbolo *NaN*, que representa *Not a Number* (não é um número).

[2] Walter Rudin não considera $\infty$ como um número. Ele trata como um símbolo que só faz parte do conjunto dos reais estendidos.

# Representação IEEE

- O padrão IEEE especifica dois formatos básicos: single e double.
- O formato single usa 32 bits (Figura 32).

Table 4.1: IEEE Single Format

| | | |
|---|---|---|
| $\pm$ | $a_1a_2a_3\ldots a_8$ | $b_1b_2b_3\ldots b_{23}$ |

| If exponent bitstring $a_1\ldots a_8$ is | Then numerical value represented is |
|---|---|
| $(00000000)_2 = (0)_{10}$ | $\pm(0.b_1b_2b_3\ldots b_{23})_2 \times 2^{-126}$ |
| $(00000001)_2 = (1)_{10}$ | $\pm(1.b_1b_2b_3\ldots b_{23})_2 \times 2^{-126}$ |
| $(00000010)_2 = (2)_{10}$ | $\pm(1.b_1b_2b_3\ldots b_{23})_2 \times 2^{-125}$ |
| $(00000011)_2 = (3)_{10}$ | $\pm(1.b_1b_2b_3\ldots b_{23})_2 \times 2^{-124}$ |
| $\downarrow$ | $\downarrow$ |
| $(01111111)_2 = (127)_{10}$ | $\pm(1.b_1b_2b_3\ldots b_{23})_2 \times 2^{0}$ |
| $(10000000)_2 = (128)_{10}$ | $\pm(1.b_1b_2b_3\ldots b_{23})_2 \times 2^{1}$ |
| $\downarrow$ | $\downarrow$ |
| $(11111100)_2 = (252)_{10}$ | $\pm(1.b_1b_2b_3\ldots b_{23})_2 \times 2^{125}$ |
| $(11111101)_2 = (253)_{10}$ | $\pm(1.b_1b_2b_3\ldots b_{23})_2 \times 2^{126}$ |
| $(11111110)_2 = (254)_{10}$ | $\pm(1.b_1b_2b_3\ldots b_{23})_2 \times 2^{127}$ |
| $(11111111)_2 = (255)_{10}$ | $\pm\infty$ if $b_1 = \cdots = b_{23} = 0$, NaN otherwise |

Figura 10: Fonte: (Overton, 2001).

# Representação IEEE

- O formato single possui:
  - 1 bit: sinal;
  - 8 bits: expoente;
  - 24 bits:
- O sinal $\pm$ representa o sinal do número.
- A primeira linha representa o zero

$$| \quad \pm \quad | \quad 00000000 \quad | \quad 00000000000000000000000 \quad |.$$

- Todas as linhas, exceto zero, começam com um bit 1 escondido.
- O expoente utiliza uma representação deslocada (biased representation).O expoente é a representação binária de $E + 127$.

## Exemplo 7

O número $1_{10} = (1{,}000\ldots0)_2 \times 2^0$ é armazenado como

$$| \quad 0 \quad | \quad 01111111 \quad | \quad 00000000000000000000000 \quad |.$$

O expoente $E$ é a representação binária de $0 + 127$.

# Representação IEEE

- A faixa de $E$ vai do número binário de 1 a 254, o que corresponde a $E_{min} = -126$ e $E_{max} = 127$.
- O menor número positivo normalizado é

$$| \quad 0 \quad | \quad 00000001 \quad | \quad 00000000000000000000000 \quad |.$$

Esse valor é denotador por

$$N_{min} = (1{,}000\ldots 0)_2 \times 2^{-126} = 2^{-126} \approx 1{,}2 \times 10^{-38}. \tag{7}$$

- O maior número positivo normalizado é

$$| \quad 0 \quad | \quad 11111110 \quad | \quad 11111111111111111111111 \quad |.$$

Esse valor é denotador por

$$N_{max} = (1{,}111\ldots 1)_2 \times 2^{127} = (2 - 2^{-23}) \times 2^{127} \approx 3{,}4 \times 10^{38}. \tag{8}$$

# Representação IEEE

- Números subnormalizados usam a combinação de expoente zero e a parte fracionária não zero.
- Números subnormais não podem ser normalizados, pois a normalização iria resultar em um expoente fora da faixa de representação.
- Números subnormais tem sua precisão reduzida a medida que o número decresce.

## Exemplo 8

O número $2^{-127} = (0{,}1)_2 \times 2^{-126}$. é representado como

$$| \quad 0 \quad | \quad 00000000 \quad | \quad 10000000000000000000000 \quad |.$$

## Exercício 10

Calcule o menor número positivo que pode ser armazenado em um sistema de 32 bits.

# Representação IEEE

- O formato double utiliza 64 bits. As definições são semelhantes as do formato de 32 bits.
- A Tabela 4

| Formato | $E_{min}$ | $E_{max}$ | $N_{min}$ | $N_{max}$ |
|---|---|---|---|---|
| Single | $-126$ | 127 | $2^{-126} \approx 1{,}2 \times 10^{-38}$ | $\approx 2^{128} \approx 3{,}4 \times 10^{38}$ |
| Double | $-1022$ | 1023 | $2^{-1022} \approx 2{,}2 \times 10^{-308}$ | $\approx 2^{1023} \approx 1{,}8 \times 10^{308}$ |

Tabela 1: Faixa dos formatos de Ponto Flutuante do IEEE

- A Tabela 2 apresenta a precisão dos formatos IEEE.

| Formato | Precisão | Epsilon da Máquina |
|---|---|---|
| Single | $p = 24$ | $\varepsilon = 2^{-23} \approx 1{,}2 \times 10^{-7}$ |
| Double | $p = 53$ | $\varepsilon = 2^{-52} \approx 2{,}2 \times 10^{-16}$ |
| Extended | $p = 64$ | $\varepsilon = 2^{-63} \approx 1{,}1 \times 10^{-19}$ |

Tabela 2: Precisão dos formatos de Ponto Flutuante do IEEE

# Representação IEEE

- A precisão single $p = 24$ corresponde a aproximadamente 7 dígitos decimais, pois $2^{-24} \approx 10^{-7}$.
- Ou de modo equivalente

$$\log_{10}(2^{24}) \approx 7. \tag{9}$$

**Exemplo 9**

A representação no formato single IEEE para

$$\pi = 3{,}141592653\ldots$$

é quando convertida para decimal

$$\pi = 3{,}141592741\ldots.$$

# Arredondamento

- Os números no padrão IEEE podem ser expressos na forma
$$\pm(1{,}b_1 b_2 \ldots b_{p-2} b_{p-1})_2 \times 2^E,$$
em que $p$ é a precisão para números normalizados. $b_0 = 1$ e $E_{min} \leq E \leq E_{max}$. Para números subnormalizados e o zero, $b_0 = 0$ e $E = E_{min}$.
- $x \in \mathbb{R}$ está na faixa normalizada se $N_{min} \leq |x| \leq N_{max}$.
- Os números $\pm 0$ e $\pm\infty$ e os números subnormais não estão na faixa normalizada, embora sejam números válidos.
- Se o número real não é um número flutuante, então ao menos uma das situações seguintes ocorre:
    1. $x$ está fora dai faixa normalizada. Por exemplo, $x = 2^{130}$ está fora da faixa normalizada no formato *single*.
    2. A expansão binária de $x$ requer um número maior que $p$ bits para uma especificação exata. Por exemplo,
    $$1 + 2^{-25} = (1{,}000000000000000000000001)_2$$
    requer mais bits que o formato *single* permite.

# Arredondamento

- Em ambos casos, é necessário aproximar $x$ por uma representação diferente do número real original.

**Definition 2**

Seja o número $x_-$ o número flutuante mais próximo de $x$ que é menor ou igual a $x$. Seja $x_+$ o número flutuante mais próximo de $x$ que é maior ou igual a $x$.

- Se o número mais próximo é zero, o sinal do zero será o sinal de $x$.
- Seja $x$ representador por

$$x = (1,b_1 b_2 \dots b_{p-2} b_{p-1} b_p b_{p+1} \dots)_2 \times 2^E. \tag{10}$$

- O número flutuante mais próximo que é menor ou igual a $x$ em (10) é:

$$x_- = (1,b_1 b_2 \dots b_{p-2} b_{p-1} \dots)_2 \times 2^E. \tag{11}$$

- $x_-$ é obtido pelo truncamento da expansão binária da mantissa, em que se descartam $b_p, b_{p+1}$.

# Arredondamento

- Se $x$ não é um número flutuante então

$$x_+ = ((1{,}b_1 b_2 \dots b_{p-2} b_{p-1})_2 + (0{,}00\dots01)_2) \times 2^E. \qquad (12)$$

- O intervalo entre $x_-$ e $x_+$ é

$$2^{-(p-1)} \times 2^E. \qquad (13)$$

- O valor de (13) é igual a $ulp(x_-)$.
- Se $x > N_{max}$ então $x_- = N_{max}$ e $x_+ = +\infty$.
- Se $x$ é positivo mas menor que $N_{min}$ então $x_-$ é subnormal ou zero e $x_+$ é subnormal ou $N_{min}$. Se $x$ é negativo a situação é reversa.

# Arredondamento

- O padrão IEEE define o valor arredondado correto de $x$, denotado por $round(x)$ da seguinte forma.
- Se $x$ é um número flutuante então $round(x) = x$. Senão, o valor depende do modo de arredondamento:
  - Arredondamento para baixo: $round(x) = x_-$.
  - Arredondamento para cima: $round(x) = x^+$.
  - Arredondamento em direção a zero: $round(x) = x_-(x > 0)$ ou $round(x) = x_+(x < 0)$
  - Arredondamento para o mais próximo: $round(x)$ é tanto $x_-$ ou $x_+$, dependendo de qual for mais próximo de $x$. Se houver um empate, aquele com o último bit significativo igual a zero é escolhido.
- O modo padrão em praticamente todas as aplicações é o arredondamento para o mais próximo.

**Exemplo 10**

Usando o padrão IEEE single, elabore um exemplo em que $x_-$ e $x_+$ estão a um mesma distância de $x$.

# Arredondamento

## Definition 3

Seja $x$ um número real. Define-se como erro absoluto de arredondamento

$$\text{abserr}(x) = |\text{round}(x) - x|. \tag{14}$$

- O erro absoluto de arredondamento é menor que o intervalo entre $x_-$ e $x_+$. Então

$$\text{abserr}(x) = |\text{round}(x) - x| < 2^{-(p-1)} \times 2^E. \tag{15}$$

- Isso significa que *abserr* é menor que uma *ulp*.
- Com o arredondamento para o mais próximo, tem-se

$$\text{abserr}(x) = |\text{round}(x) - x| < 2^{-p} \times 2^E, \tag{16}$$

ou metade de uma ulp.

# Arredondamento

**Definition 4**

O erro de arredondamento relativo para um número $x$ é definido por

$$\mathrm{relerr}(x) = |\delta| = |\frac{\mathrm{round}(x) - x}{x}|. \tag{17}$$

- O erro relativo satisfaz o limite

$$\mathrm{relerr}(x) = |\delta| = |\frac{\mathrm{round}(x) - x}{x}| < \frac{2^{-(p-1)} \times 2^E}{2^E} = 2^{-(p-1)} = \varepsilon. \tag{18}$$

- Para o arredondamento para o mais próximo, tem-se:

$$|\delta| < \frac{1}{2}\varepsilon. \tag{19}$$

- O número de bits em acordo para o arredondamento para o mais próximo é

$$-\log_2(\mathrm{relerr}(x)) > p. \tag{20}$$

# Arredondamento

- O número de dígitos decimais em concordância é

$$-\log_{10}(\mathrm{relerr}(x)) > -\log_{10}(\varepsilon). \quad (21)$$

- Para o formato single, $\mathrm{round}(x)$ e $x$ concordam em pelo menos 7 casas digitais (Tabela 2).

**Theorem 1**

*Seja $x \in R$ normalizado em um sistema binário de ponto flutuante com precisão $p$. Então*

$$\mathrm{round}(x) = x(1+\delta)$$

*para um $\delta$ que satisfaz*

$$|\delta| < \varepsilon = 2^{-(p-1)}$$

*ou para modo de arredondamento para o mais próximo*

$$|\delta| < \frac{1}{2}\varepsilon = 2^{-p}.$$

# Operações no padrão IEEE

- Um ponto central do padrão IEEE é exigir que:
  - Operações aritmética arredondadas corretamente (adição, subtração, multiplicação e divisão).
  - Operação de resto e raiz quadrada arredondada corretamente.
  - Conversão de formato arredondada corretamente.
- Arredondado corretamente significa que o arredondamento se adequará o resultado de destino.
- É comum que o resultado de uma operação aritmética de dois pontos flutuantes não seja um ponto flutuante.

**Exemplo 11**

1 e $2^{-24}$ são números flutuantes (representação exata). Mas o resultado de $1 + 2^{-24}$ não é um número flutuante.

# Operações no padrão IEEE

- Seja $x$ e $y$ números flutuantes, então

$$\begin{aligned} x \oplus y &= \mathrm{round}(x+y) = (x+y)(1+\delta) \\ x \ominus y &= \mathrm{round}(x-y) = (x-y)(1+\delta) \\ x \otimes y &= \mathrm{round}(x\times y) = (x\times y)(1+\delta) \\ x \oslash y &= \mathrm{round}(x\div y) = (x\div y)(1+\delta) \end{aligned}$$

em que

$$|\delta| < \frac{1}{2}\varepsilon.$$

## Exercício 11

Seja $x = z = 1$ e $y = 2^{-25}$. Calcule

(a) $(x+y)-z$ no formato *single*.

(b) $(x+y)-z$ no formato *double*.

# Operações no padrão IEEE

### Exercício 12

Considerando os valores no Exercício 11 e formato *single*, calcule

(a) $x \oplus (y \ominus z)$.

(b) $(x \ominus z) \oplus y$.

- A disponibilidade dos modos de arredondamento para baixo e para cima permite que o programador faça qualquer computação duas vezes, em cada um dos modos. Os dois resultados definem um intervalo em que o resultado exato deve estar contido. Essa metodologia dá origem a computação por intervalos.

# Operações no padrão IEEE

- A entrada de número normalmente é feita por algum tipo de programação em alto nível. Em seguida é processada por um compilador ou interpretador. É possível de entrar com o número por meio de sua expansão decimal ou por meio de número fracionário.
- O padrão prevê vários tipos de conversão entre formatos. Segue alguns:
    - Conversão entre diferentes formatos de ponto flutuante (single e double, por exemplo).
    - Conversão entre ponto flutuante e formato inteiro.
    - Conversão de binário para decimal e decimal para binário.

# Exceções

- Um dos aspectos mais difíceis de programação é a necessidade de antecipar situações de exceção.
- O mais simples exemplo de exceção é a divisão por zero.
- Até os anos de 1950, a divisão de um número positivo por zero resultava no maior número flutuante disponível.
- A partir de 1960, essa divisão passou a gerar uma interrupção no programa, oferecendo ao usuário alguma mensagem do tipo: "fatal error - division by zero".

## Exemplo 12

O cálculo da resistência total de um circuito elétrico com dois resistores conectados em paralelo, $R_1$ e $R_2$, quando $R_1 = 0$ é dado por

$$T = \frac{1}{\frac{1}{R_1} + \frac{1}{R_2}} = \frac{1}{\frac{1}{0} + \frac{1}{R_2}} = \frac{1}{\infty + \frac{1}{R_2}} = \frac{1}{\infty} = 0.$$

# Exceções

- É verdade que $a \times 0$ é zero para qualquer valor finito de $a$. Também é verdade que $a/0 = \infty$, $\infty + \infty = \infty$ e $a \times \infty = \infty$.
- As expressões $0 \times \infty$, $\infty - \infty$ e $0/0$ não tem sentido matemático. São consideradas operações inválidas.
- O padrão IEEE responde a essas operações com o resultado NaN (Not a Number).

## Exemplo 13

Seja as sequências $x_k$ e $y_k$ divergentes to $\infty$, por exemplo, $x_k = 2^k$ e $y_k = 2k$, para $k = 1,2,3\ldots$. Claramente a sequência $x_k + y_k$ diverge para $\infty$. Mas não é possível calcular o resultado de $x_k - y_k$. Consequentemente $\infty - \infty$ é NaN.

# Exceções

- Qual a razão para $1/0$ ter o valor de $\infty$ e não $-\infty$?
- Para responder isso, tem-se diferentes números para 0 e $-0$, de tal forma que $a/0 = \infty$ e $a/-0 = -\infty$.
- O teste $0 == -0$ é verdadeiro e o teste $\infty == -\infty$ é falso.
- A raiz quadrada de um número negativo é uma operação inválida [3].
- Não há relação de ordem para variáveis que recebem NaN.
- Overflow ocorre quando o resultado da operação é finito mas com valor absoluto maior que o maior número flutuante disponível.
- No padrão IEEE, o resultado de um overflow é $\pm N_{max}$ ou $\pm\infty$ a depender do modo de arredondamento.
- Underflow ocorre quando o valor exato de uma operação não é zero mas possui valor absoluto menor que o menor número flutuante normalizado.
- No padrão IEEE, o resultado de um underflow é um número subnormal, $\pm N_{min}$ ou $\pm 0$ a depender do modo de arredondamento. Esse processo é conhecido como gradual underflow.

[3] No Scilab, o resultado de $\sqrt{-1} = i$

# Exceções

## Exemplo 14

Considere a seguinte operação usando o padrão single do IEEE. O segundo operando é $N_{min}$ e o primeiro é ligeiramente superior.

$$\begin{aligned}
& (\ 1{,}01000000000000000000000\ )_2 \times 2^{-126} \\
- \quad & (\ 1{,}00000000000000000000000\ )_2 \times 2^{-126} \\
= \quad & (\ 0{,}01000000000000000000000\ )_2 \times 2^{-126} \\
\text{Normalize :} \quad = \quad & (\ 1{,}00000000000000000000000\ )_2 \times 2^{-128}
\end{aligned} \tag{22}$$

- Sem gradual overflow, o resultado do Exemplo 14 seria zero. O valor correto pode ser armazenado como

$$|\ 0\ |\ 00000000\ |\ 01000000000000000000000\ |.$$

# Exceções

| Situação | Resultado |
|---|---|
| Operação Inválida | NaN |
| Divisão por zero | $\pm\infty$ |
| Overflow | $\pm\infty$ ou $\pm N_{max}$ |
| Underflow | Subnormal, $\pm 0$ ou $\pm N_{min}$ |
| Inexato | Valor arredondado corretamente |

Tabela 3: Resposta padrão a exceções de acordo com o padrão IEEE.

- A Tabela 3 sintetiza a resposta do padrão IEEE a cinco tipos de exceção.

## Exercício 13

Pense em alguma alternativa que possa tornar o cálculo de (23) mais eficiente.

$$\sqrt{x^2 + y^2} \tag{23}$$

# Microprocessadores Intel

- Os dois principais fabricantes de chips a adotarem o padrão IEEE foram Intel (nos computadores da IBM) e Motorola (nos computadores da Apple e Sun).
- O microprocessador Intel original era o chip 8086 em 1978, que incluía a unidade central de processamento (CPU) e a unidade de lógica e aritmética (ALU), mas sem operações de ponto flutuante.
- Em 1980, a Intel anunciou os chips 8087 e 8088, sendo primeiramente usados pelos computadores da IBM.
- O chip 8087 possuía um co-processador para ponto flutuante, oferecendo assim uma unidade de ponto flutuante (FPU).
- Os sucessores do 8087, os chips 80287, 80387 possuíam co-processadores separados.
- A partir dos chips 80486 DX, o Pentium, o Pentim Pro, o Pentium II passaram a incluir a FPU no chip principal.
- Embora as gerações de chips sejam cada vez mais rápidas, a arquitetura da família Pentium permanece essencialmente a mesma do chip 8087.

# Microprocessadores Intel

- Instruções de ponto flutuante operam principalmente em dados armazenados em registradores de 80 bits.
- Espera-se que os programas armazenem as variáveis usando o formato single ou double.
- A precisão extendida pode ser usada para aumentar a precisão de operações aritméticas. O resultado deve ser arredondado para o formato single ou double.

# Microprocessadores Intel

- Os registradores estão organizados em uma pilha numerada de 0 a 7.
- Em qualquer instante, o topo da pilha é denotado por $ST(0)$, o segundo por $ST(1)$, e assim sucessivamente.
- A pilha de registros é conveniente para calcular expressões aritméticas.
- Por exemplo, considere a seguinte expressão a ser computada

$$(a+b)\times c,$$

assumindo que os números flutuantes $a$,$b$, e $c$ estão disponíveis na memória na posição $A, B$ e $C$, respectivamente. O resultado será armazenado na posição $X$. A sequência de instruções em linguagem assembly é descrita por um conjunto mnemônico de palavras para descrever os passos adotados pelo computador.

# Microprocessadores Intel

- Instruções em Assembly para a expressão $(a + b) \times c$.

```
FLD A
FLD B
FADD
FLD C
FMUL
FSTP X
```

- `FLD` coloca o valor presente na memória na posição $A$ no topo da pilha.
- Quando há um valor no topo, `FLD` empurra esse valor para a segunda posição.
- `FADD` adiciona o valor de $ST(0)$ a $ST(1)$.
- `FMUL` multiplica os valores de $ST(0)$ e $ST(1)$.
- `FSTP` armazena o resultado final na posição $X$ da memória.

# Microprocessadores Intel

Tabela 4: Conteúdo dos registrados em tempos sucessivos.

| Registrador | Time 0 | Time 1 | Time 2 | Time 3 | Time 4 | Time 5 |
|---|---|---|---|---|---|---|
| $ST(0)$ | | $a$ | $b$ | $a \oplus b$ | $c$ | $(a \oplus b) \otimes c$ |
| $ST(1)$ | | | $a$ | | $a \oplus b$ | |
| $ST(2)$ | | | | | | |
| $ST(3)$ | | | | | | |
| $ST(4)$ | | | | | | |
| $ST(5)$ | | | | | | |
| $ST(6)$ | | | | | | |
| $ST(7)$ | | | | | | |

- Os valores nos registradores não números ponto flutuantes, não fórmulas. A expressão $a \oplus b$ é usada ao invés de $a + b$, porque demonstra o valor correto após arredondamento.

# Microprocessadores Intel

- Adicionalmente ao ponteiro que indica a posição no topo da pilha, o status word contém as cinco exceções que são requeridas pelo IEEE.
- Há também a control word, com registro de 16 bit exclusivo, é usada para definir o modo de arredondamento, o modo de precisão e as mascaras de exceção.
- O ponto flutuante tem suas condições iniciais definidas pela variável `FNINIT`. Esta variável define:
  - Modo de arredondamento: round to nearest.
  - Modo de precisão: extended.
  - Exceções: todas recebem uma máscara.
- Em 2000 a Intel anunciou a criação do IA 64 Itanium chip.
- ia 64 possui 128 registradores e é capaz de realizar a seguinte conta corretamente $(a \times b + c)$. Essa operação é chamada de `FMA`, do inglês fused multiply-add instruction.

# Exercícios em Sala

1. Calcule o resultado para $y$ nas seguintes operações.

   (a) $y = \lim_{n\to\infty} \frac{1}{2}\left(x_n + \frac{2}{x_n}\right)$ quando $x_0 \in [0{,}5; 1{,}5]$.

   (b) Calcule $y = \lim_{n\to\infty} x_{n+1}$, para $x_0 = 27$ e

   $$x_{n+1} = \begin{cases} x_n/2 & \text{if } x_n \text{ for par} \\ 3x_n + 1 & \text{if } x_n \text{ for ímpar} \end{cases}.$$

2. Seja a função recursiva definida por

   $$x_{n+1} = 4{,}1x_n(1 - x_n),$$

   que normalmente é chamada de equação logística. Considere a sequência de valores dada por $\{x_n\}$. Calcule $y = \{x_n\}$ quando $n \to \infty$ e a condição inicial é dada por

   $$x_0 = -\frac{\sqrt{20(41)^{5/2} + 1447341} - 1681}{3362}.$$

# Ponto Flutuante no C

- A linguagem de programação C tornou-se muito popular a partir dos anos 1980.
- No C, o format float refere-se ao formato single do IEEE.
- Cálculos feitos com formato float utiliza precisão de 32 bits.
- O Exemplo 15 refere-se a um programa que lê uma variável $x$ e imprime na tela o valor dessa variável.

### Exemplo 15

```
main ()  /* Program 1: Echo */
{
  float x;
  scanf("%f",&x);
  printf("x=%f",x);
}
```

# Ponto Flutuante no C

- O segundo argumento de `scanf` indica o endereço de $x$.
- O segundo argumento de `printf` é o valor de $x$.
- Os dois tipos de formato no C são o `%f` e `%e`, indicando úm número fixo de casas decimais e o formato exponencial.
- Considere a entrada de $x = 2/3$.

Tabela 5: Saída do Exemplo 15 para diferentes formatos.

| Código do Formato | Saída |
|---|---|
| `%f` | 0.666667 |
| `%e` | 6.666667e-01 |
| `%8.3f` | 0.667 |
| `%8.3e` | 6.667e-01 |
| `%21.15f` | 0.666666686534882 |
| `%21.15e` | 6.666666865348816e-01 |

# Ponto Flutuante no C

- O valor da entrada é arredondado corretamente para 24 bits de precisão na sua mantissa, o que corresponde a aproximadamente 7 dígitos decimais.
- O C utiliza o modo de arredondamento para o mais próximo.
- As duas primeiras linhas da Tabela 5 mostra o formato single com a o arredondamento para o mais próximo (o que explica o dígito 7).
- A terceira e quarta linha, solicitam apenas 3 casas decimais.
- As duas últimas linhas tentam mostrar um resultado com um número maior de casas decimais que o possível. O resultado é que cerca de metade dos dígitos não possuem significado.
- Para o formato double é necessário usar os comandos `%lf` e `%le`.

# Ponto Flutuante no C

- Considere o seguinte programa com um loop.

```
main ()  /* Program 2: First Loop Program */
{
  int n = 0;
  float x = 1;

/* Repatedly divide x by 2 until it underflows to 0 */

  while (x>0) {
  x = x/2;
  n++;
  printf("(2 raised to the power -%d) = %e \n",n,x);
  }
}
```

# Ponto Flutuante no C

- Considere um outro programa com um loop.

```
main ()  /* Program 3: Second Loop Program */
{
  int n = 0;
  float x = 1, y = 2;

/* Repatedly divide x by 2 until y = (1+x) rounds to 1 */

  while (y>1) {
  x = x/2;
  y=1+x;
  n++;
  printf("(1 added to (2 to the power -%d) = %e \n", n, y);
  }
}
```

# Ponto Flutuante no C

- Considere um outro programa com um loop.

```
main ()  /* Program 4: Parallel Resistance Formula */
{
  float r1, r2, total;

  printf("Enter the two resistances \n");
  scanf("%f %f, &r1,&r2);

  printf("r1=%e  r2=%e \n",r1, r2);

  total(1/(1/r1+1/r2);

  printf("Total resitance s %e \n", total);
}
```

# Ponto Flutuante no C

Tabela 6: Resultados do Cálculo da Resistência Paralela

| $R_1$ | $R_2$ | Resistência Total |
|---|---|---|
| 1 | 1 | $5.000000e-01$ |
| 1 | 10 | $9.090909e-01$ |
| 1 | 1000 | $9.990010e-01$ |
| 1 | $1.0e5$ | $9.999900e-01$ |
| 1 | $1.0e10$ | $1.000000e+00$ |
| 1 | 0.1 | $9.090909e-02$ |
| 1 | $1.0e-5$ | $9.999900e-06$ |
| 1 | $1.0e-10$ | $1.000000e-10$ |
| 1 | 0 | $0.000000e+00$ |

## Exercício 14

Se $R_1 = 1$, aproximadamente para qual faixa de valores de $R_2$ o Program 4 oferece um resultado igual a 1?

# Ponto Flutuante no C

- Para iniciar a biblioteca matemática no C, é necessário incluir o comando `#include <math.h>` no início do programa.

Tabela 7: Algumas funções matemáticas do C

| Rotina | Descrição |
|---|---|
| `fabs` | valor absoluto |
| `sqrt` | raiz quadrada |
| `exp` | exponencial (base $e$) |
| `log` | logaritmo (base $e$ |
| `log10` | logaritmo (base 10) |
| `sin` | seno (argumento em radianos) |
| `cos` | cosseno (argumento em radianos) |
| `atan` | arco-tangente (resultado em radianos) |
| `pow` | potência (`pow(x,y)` retorna $x^y$ |

- O padrão IEEE e o padrão C99 não definem qual a precisão para essas funções (exceto a raiz quadrada).

# Cancelamento

- Considere os os dois números

$$x = 3{,}141592653589793$$

e

$$y = 3{,}141592653585682$$

.

- O primeiro número é uma aproximação de 16 dígitos de $\pi$, enquanto o segundo concorda com $\pi$ em 12 dígitos. A diferença é

$$z = x - y = 0{,}000000000004111 = 4{,}111 \times 10^{-12}. \qquad (24)$$

- Se a diferença for calculada no C com precisão *single* o resultado seria

$$\mathtt{z} = 0{,}000000\mathtt{e} + 00 \qquad (25)$$

- A diferença entre (24) e (25) que primeiro o computador converte o número decimal para binário. Em seguida, trunca esse valor para poder armazenar. No formato single, esses dois valores são iguais.

# Cancelamento

- Se o formato double for usado, o resultado será a diferença:

$$z = 4{,}110933815582030 \times 10^{-12}. \tag{26}$$

- O valor em (26) concorda em 3 dígitos com (24). Qual é o significado dos outros dígitos?
- A resposta é que o valor em (26) é a diferença arredondada entre $x$ e $y$. Os demais dígitos são espúrios.
- Neste caso, dizemos que há uma perda de precisão na computação $z = x - y$.
- Independente se há uma perda de precisão completa ou parcial, este fenômeno é chamado de cancelamento.

# Cancelamento

- Um exemplo de cancelamento pode ser obtido pela aproximação da derivada.
- Seja $f : \mathbb{R} \to \mathbb{R}$ uma função contínua e diferenciável.
- Como pode ser feito uma estimativa de $f'(x)$?
- Por definição, tem-se:

$$\lim_{h \to 0} \frac{f(x+h) - f(x)}{h}. \tag{27}$$

- Uma idéia natural é avaliar (27) para um valor pequeno de $h$. Mas quão pequeno?
- Mostraremos um programa em que o valor de $h$ é definido para faixa de $10^{-1}$ a $10^{-20}$.
- Neste caso, nós sabemos precisamente qual é o valor da derivada de $\sin(x)$ em $x$, que é $\cos(x)$. E assim, pode-se comparar os resultados.
- O valor absoluto da diferença entre o valor calculado por (27) e o valor correto é denominado erro.

# Cancelamento

## Exemplo 16

```
%Programa 5 - Elaborado no Matlab

%Parâmetros iniciais
x=1;h=1;n=1;deriv=cos(x);
%Exibir na tela
sprintf('O valor da derivada é %13.6e',deriv)
disp('   h       diffquo       abs(deriv - diffquo)')

%Definir h na faixa de 10^{-1} a 10^{-20}

while (n <= 20)
  h(n+1)=h(n)/10;
  diffquo(n)=(sin(x+h(n+1))-sin(x))/h(n+1);
  error(n)=abs(deriv-diffquo(n));
  disp(sprintf('%5.1e %13.6e %13.6e',h(n+1),diffquo(n),error(n)))
  n=n+1;
end
```

# Cancelamento

```
O valor da derivada é  5.403023e-01

   h       diffquo       abs(deriv - diffquo)
1.0e-01  4.973638e-01  4.293855e-02
1.0e-02  5.360860e-01  4.216325e-03
1.0e-03  5.398815e-01  4.208255e-04
1.0e-04  5.402602e-01  4.207445e-05
1.0e-05  5.402981e-01  4.207362e-06
1.0e-06  5.403019e-01  4.207468e-07
1.0e-07  5.403023e-01  4.182769e-08
1.0e-08  5.403023e-01  2.969885e-09
1.0e-09  5.403024e-01  5.254127e-08
1.0e-10  5.403022e-01  5.848104e-08
1.0e-11  5.403011e-01  1.168704e-06
1.0e-12  5.403455e-01  4.324022e-05
1.0e-13  5.395684e-01  7.339159e-04
1.0e-14  5.440093e-01  3.706976e-03
1.0e-15  5.551115e-01  1.480921e-02
1.0e-16  0.000000e+00  5.403023e-01
1.0e-17  0.000000e+00  5.403023e-01
1.0e-18  0.000000e+00  5.403023e-01
1.0e-19  0.000000e+00  5.403023e-01
1.0e-20  0.000000e+00  5.403023e-01
```

# Cancelamento

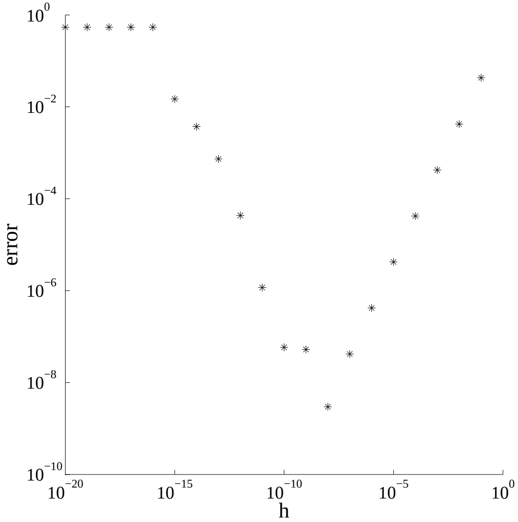


Figura 11: Erro em função do valor de $h$. Escala logarítmica. Conforme programa no Exemplo 16.

# Cancelamento

- O erro é apresentado na Figura 81 em função do valor de $h$, usando uma escala logaritmica.
- O resultado é bastante interessante. Percebe-se que o erro diminui a medida em que o valor de $h$ se torna menor, como é de se esperar.
- Mas esse fenômeno ocorre até um certo ponto. Quando $h$ se torna pequeno demais, a aproximação se torna pior. Por quê?
- Se $x = 1$ e $h$ é menor que metade da precisão da máquina ($\approx 10^{-16}$), então $x + h$, isto é, $1 + h$ é arredondado para 1, e o valor de $\sin(x + h)$ e $\sin(x)$ se cancelam completamente.
- Pode-se resumir essa questão da seguinte forma. Ao usar $h$ muito grande, enfrenta-se um alto erro de discretização. Enquanto ao usar um $h$ muito pequeno, aparece um alto erro de cancelamento.
- Para a função $f(x) = sin(x)$ com $x = 1$, a melhor escolha de $h$ é em torno de $10^{-8}$, aproximadamente o a raiz quadrada da precisão da máquina.

# Cancelamento

- O valor do erro é reduzido por 10 a medida que $h$ é reduzido em 10, até o ponto em que o cancelamento torna-se expressivo.
- Assume-se que $f$ contínua e duas vezes diferenciável, como é o caso da função $\sin(x)$.
- Existe um valor $z$ entre $x$ e $x+h$ tal que

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2} f''(z), \tag{28}$$

  em que $f''(z)$ é a segunda derivada de $f$ em $z$.
- A Equação (28) é chamada de série de Taylor truncada.
- De (28), tem-se:

$$\frac{f(x+h) - f(x)}{h} - f'(x) = \frac{h^2}{2} f''(z). \tag{29}$$

- A quantidade (29) é a diferença enter aquilo que é computado e o valor da derivada. O valor absoluto de (29) é o erro de discretização que é da ordem de $O(h)$.

# Cancelamento

- Ao reduzir $h$ em 10, o erro de discretização também reduz em 10 (aproximadamente, uma vez que o valor de $z$ também muda).
- Deve-se dessa forma, sempre que possível, evitar o cancelamento.
- Se $f$ for suficientemente suave, é possível calcular um valor mais aproximado da derivada.
- Para isso, computa-se a inclinação da reta passando por $(x+h, f(x+h))$ e $(x-h, f(x-h))$.
- A Equação (27) torna-se

$$\lim_{h \to 0} \frac{f(x+h) - f(x-h)}{2h}. \tag{30}$$

- A Equação (30) é chamada de quociente de diferença central.
- Para um valor suficientemente pequeno de $h$ (mas grande suficiente para que o cancelamento não seja expressivo), o quociente da diferença central é mais preciso que o quociente da diferença.

# Cancelamento

- Considere a série de Taylor truncada

$$f(x+h) = f(x) + hf'(x) + \frac{h^2}{2}f''(x) + \frac{h^3}{6}f'''(z_1) \qquad (31)$$

para $z_1$ entre $x$ e $x+h$.

- Considerando, $x-h$, tem-se

$$f(x-h) = f(x) - h'f'(x) + \frac{h^2}{2}f''(x) - \frac{h^3}{6}f'''(z_2) \qquad (32)$$

para $z_2$ entre $x$ e $x-h$.

- Subraindo (32) de (31) e divindo por $2h$,tem-se

$$\frac{f(x+h)-f(x-h)}{2h} - f'(x) = \frac{h^2}{12}(f'''(z_1) - f'''(z_2)). \qquad (33)$$

- O valor absoluto de (33) é o erro de discretização para o quociente da diferença central, que é da ordem de $O(h^2)$, ao invés de $O(h)$ para o quociente da diferença.

# Cancelamento

- O exercício 11.1, pág. 75, (Overton, 2001), pede para mudar o programa do Exemplo 16, e usar o quociente da diferença central.
- O programa foi implementado no Matlab conforme Exemplo 17.

## Exemplo 17

```
x=1;h=1;n=1;deriv=cos(x);
sprintf('O valor da derivada é %13.6e',deriv)
disp('   h       diffquo      abs(deriv - diffquo)')

%Definir h na faixa de 10^{-1} a 10^{-20}

while (n <= 20)
  h(n+1)=h(n)/10;
  diffquo(n)=(sin(x+h(n+1))-sin(x-h(n+1)))/(2*h(n+1));
  error(n)=abs(deriv-diffquo(n));
  disp(sprintf('%5.1e %13.6e %13.6e',h(n+1),diffquo(n),error(n)))
  n=n+1;
end
```

# Cancelamento

```
O valor da derivada é  5.403023e-01

   h       diffquo       abs(deriv - diffquo)
1.0e-01  5.394023e-01  9.000537e-04
1.0e-02  5.402933e-01  9.004993e-06
1.0e-03  5.403022e-01  9.005045e-08
1.0e-04  5.403023e-01  9.004295e-10
1.0e-05  5.403023e-01  1.114087e-11
1.0e-06  5.403023e-01  2.771683e-11
1.0e-07  5.403023e-01  1.943278e-10
1.0e-08  5.403023e-01  2.581230e-09
1.0e-09  5.403023e-01  2.969885e-09
1.0e-10  5.403022e-01  5.848104e-08
1.0e-11  5.403011e-01  1.168704e-06
1.0e-12  5.402900e-01  1.227093e-05
1.0e-13  5.401235e-01  1.788044e-04
1.0e-14  5.440093e-01  3.706976e-03
1.0e-15  5.551115e-01  1.480921e-02
1.0e-16  5.551115e-01  1.480921e-02
1.0e-17  0.000000e+00  5.403023e-01
1.0e-18  0.000000e+00  5.403023e-01
1.0e-19  0.000000e+00  5.403023e-01
1.0e-20  0.000000e+00  5.403023e-01
```

# Cancelamento

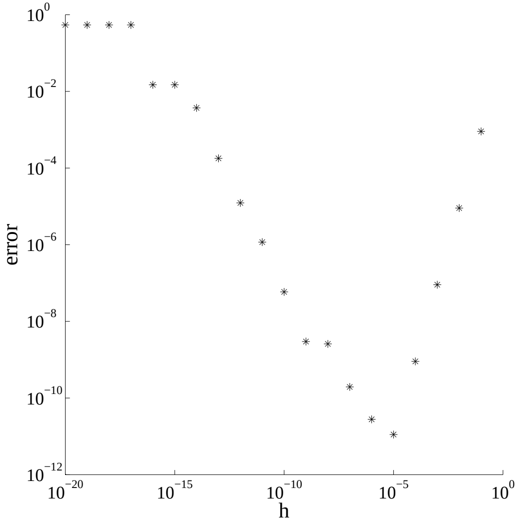


Figura 12: Erro em função do valor de $h$. Escala logarítmica. Conforme programa no Exemplo 17.

# Condicionamento de Problemas

- O condicionamento de um problema mede a precisão que se pode esperar ao se utilizar um sistema de ponto flutuante, independentemente do algoritmo utilizado.
- Seja $y = f(x)$ uma função duas vezes diferenciável e que $x$ e $f(x)$ estejam na faixa normalizada. Considere

$$\hat{x} = \mathrm{round}(x).$$

- Utilizando um computador com ponto flutuante, o melhor que se pode esperar do resultado é

$$\hat{y} = f(\hat{x}).$$

- Observação: $f$ não pode ser calculada sempre exatamente.
- O erro relativo satisfaz o seguinte limite

$$\frac{|\hat{x} - x|}{|x|} < \varepsilon,$$

em que $\varepsilon$ é a precisão de máquina (há um fator de 1/2 quando se arredonda para o mais próximo).

# Condicionamento de Problemas

- Segue então que

$$-\log_{10}\left(\frac{|\hat{x}-x|}{|x|}\right) > -\log_{10}(\varepsilon). \tag{34}$$

- O lado esquerdo da inequação (34) estima o número de dígitos em que $\hat{x}$ concorda com $x$.
- Em quantos dígitos, pode-se esperar que $\hat{y}$ concorde com $y$?
- Para encontrar a resposta, deve-se analisar

$$-\log_{10}\left(\frac{|\hat{y}-y|}{|y|}\right).$$

- Seja

$$\frac{\hat{y}-y}{y} = \frac{f(\hat{x})-f(x)}{\hat{x}-x} \times \frac{x}{f(x)} \times \frac{\hat{x}-x}{x}. \tag{35}$$

# Condicionamento de Problemas

- O fator (36) aproxima $f'(x)$.

$$\frac{f(\hat{x}) - f(x)}{\hat{x} - x}, \tag{36}$$

- Assim, tem-se:

$$\frac{\hat{y} - y}{y} \approx \kappa_f \times \frac{\hat{x} - x}{x} \tag{37}$$

em que

$$\kappa_f = \frac{|x| \times |f'(x)|}{|f(x)|}. \tag{38}$$

- $\kappa_f$ é chamado de número condicional de $f$ em $x$. Isto mede aproximadamente quanto o erro de arredondamento relativo é amplificado pelo cálculo de $f$ em $x$.
- Tomando o logaritmo na base 10 de ambos os lados de (37), tem-se

$$-\log_{10}\left(\frac{\hat{y} - y}{y}\right) \approx -\log_{10}\left(\frac{\hat{x} - x}{x}\right) - \log_{10}(\kappa_f). \tag{39}$$

# Condicionamento de Problemas

- O lado esquerdo de (39) é aproximadamente o número de dígitos em que $\hat{y}$ concorda com $y$.
- O primeiro termo do lado direito de (39) é aproximadamente o número de dígitos que $\hat{x}$ concorda com $x$ (que no formato single é de 7 dígitos).
- Regra prática: para estimar o número de dígitos em que $\hat{y} = f(\hat{x})$ concorda com $y = f(x)$, diminua

$$\log_{10}(\kappa_f)$$

do número de dígitos que $\hat{x} = \mathrm{round}(x)$ concorda com $x$ (7 no single e 16 no double).

# Condicionamento de Problemas

## Exercício 15

Calcule o númro de dígitos perdido para as seguintes funções e condições iniciais:

(1) $\exp(x)$
   (a) 1,000001
   (b) 0,000001
   (c) $-1{,}000001$
(2) $\log(x)$
   (a) $e$ (double)
   (b) 1,001
   (c) 1,000001
   (d) $1/e$ (double)
(3) $\sin(x)$
   (a) $\pi$ (double)
   (b) $\pi/2$ (double)
   (c) 0,000001

### Exercício 16

Seja a equação do mapa logístico

$$x_{n+1} = rx_n(1 - x_n), \tag{40}$$

com $r = 327/100$ e $r = 39/10$. Considere o conjunto de condições iniciais $x_0 = [0{,}1; 100/327; 10/39]$. Calcule o valor de $\kappa_f$ para $x_1$ e para $x_n$ quando $n \to \infty$.

# Estabilidade de Algoritmos

- Um algoritmo é um método computacional bem definido para resolver uma determinada classe de problemas.
- Em ciência da computação, o estudo de algoritmos se concentra na eficiência, ou seja, na habilidade em obter a resposta correta.
- Algoritmos numéricos objetiva encontrar uma solução aproximadamente correta.
- A estabilidade de um algoritmo mede a qualidade do algoritmo em obter uma resposta para o problema.
- Algoritmos que obtém respostas imprecisas são chamadas instáveis.

# Estabilidade de Algoritmos

- Seja a função
$$y = f(x),$$
duas vezes diferenciável com $x$ e $f(x)$ na faixa normalizada.
- Deseja-se calcular o valor de $y$, mas o melhor que se pode conseguir é
$$\hat{y} = f(\hat{x}). \tag{41}$$
- Um algoritmo para calcular $f(x)$ é estável se o resultado $\tilde{y}$ satisfaz
$$\frac{|\tilde{y} - y|}{|y|} \approx \kappa_f \frac{|\hat{x} - x|}{|x|}, \tag{42}$$
- $\tilde{y}$ é uma aproximação de $\hat{y}$. Assim, não se afirma que $\tilde{y} = f(\hat{x})$.
- Considere o problema de juros compostos.
- Seja um investimento de $a_0$ em um banco que paga 5% de juros ao ano, calculado a cada 4 meses.

# Estabilidade de Algoritmos

- Isso significa que ao fim do primeiro quadrimestre do ano, o valor do investimento é

$$a_1 = a_0 \times (1 + 0{,}05/4). \tag{43}$$

- Ao fim do segundo período tem-se

$$a_2 = a_0 \times (1 + 0{,}05/4)^2. \tag{44}$$

- E assim ao fim de $n$ períodos

$$a_n = a_0 \times (1 + 0{,}05/4)^n. \tag{45}$$

- Em termos gerais, a equação para juros compostos é

$$a_n = a_0 \times C_n(x) \tag{46}$$

em que

$$C_n(x) = \left(1 + \frac{x}{n}\right)^n. \tag{47}$$

# Estabilidade de Algoritmos

- Para um valor fixo de $x$, tem-se que $C_n(x)$ tem um valor limite, para $n \to \infty$, de $\exp(x)$.
- Vamos investigar o número condicional de $C_n(x)$. Usando a regra da cadeia, o valor da derivada é

$$C_n'(x) = n\left(1 + \frac{x}{n}\right)^{n-1}\frac{1}{n} = \frac{C_n(x)}{1 + \frac{x}{n}}. \tag{48}$$

e o valor do número condicional

$$\kappa_C = \frac{|x|}{|C_n(x)|} \times |C_n'(x)| = \frac{|x|}{|C_n(x)|} \times \frac{|C_n(x)|}{|1 + \frac{x}{n}|} = \frac{|x|}{|1 + x/n|}. \tag{49}$$

- Eq. (49) é bem definida para $n$ grande, considerando que $|x|$ não seja grande.

# Estabilidade de Algoritmos

## Exemplo 18

```
x=input('Entre com o valor de x: ')
n=input('Entre com o valor do número de meses: ')
%Precisão single
x=single(x);n=single(n);
z=1+x/n;w=1;
for i=1:n
    w=w*z;
end
v=log(z);
disp(sprintf('Algoritmo 1: a resposta é %13.6e',w));
disp(sprintf('Algoritmo 2: a resposta é %13.6e',power(z,n)));
disp(sprintf('Algoritmo 3: a resposta é %13.6e',exp(n*v)));
XS%Melhor método - Algoritmo 4:
%log1p(s) = log(1+s)
u=x/n;
v=log1p(u);
disp(sprintf('Algoritmo 4: a resposta é %13.6e',exp(n*v)));
```

# Estabilidade de Algoritmos

- Resultado do Exemplo 18

```
Algoritmo 1: a resposta é  1.000000e+00
Algoritmo 2: a resposta é  1.000000e+00
Algoritmo 3: a resposta é  1.000000e+00
Algoritmo 4: a resposta é  1.051271e+00
```

# Estabilidade de Algoritmos

- Instabilidade devido a Cancelamento.
- O fenômeno de cancelamento pode ser pode ser explicado pelo número de condicionamento. Seja a função

$$f(x) = x - 1 \tag{50}$$

e

$$\kappa_f = \frac{|x|}{|x-1|} \tag{51}$$

- A Eq. (51) mostra valores muito altos para $x \approx 1$.
- Assim, um algoritmo que introduz cancelamento, também introduz mal condicionamento.

# Estabilidade de Algoritmos

## Exemplo 19

- Leia, discuta em grupo, e implemente o artigo:

- Nepomuceno, E. G. (2014) Convergence of recursive functions on computers. *The Journal of Engineering, Institution of Engineering and Technology*, 1-3. doi: 10.1049/joe.2014.0228

# Análise por Intervalos

- Na era da computação com aritmética finita, há uma necessidade por compreender a análise por intervalos ou matemática por intervalos.
- Qualquer pessoa que use um computador, pode questionar: Qual é o efeito do erro no meu resultado?
- O intervalo é visto como um novo tipo de número, representado por um par de números reais, que são os seus extremos.
- O intervalo passa a ter uma dupla natureza: tanto como número, quanto como conjunto.
- No caso dos pontos extremos terem o mesmo valor, esse novo número é um número real.
- Se for possível calcular um intervalo $[a,b]$ em que esteja a solução $x$ para algum problema, então o ponto médio, $m = (a+b)/2$, deste intervalo é uma aproximação de $x$.
- Tem-se também que $|x - m| \leq w/2$, em que $w = b - a$ é a largura do intervalo $[a,b]$.

# Representações Finitas

## Definition 5

Por um intervalo, entende-se um conjunto fechado e limitado de números reais

$$[a,b] = \{x : a \leq x \leq b\}. \quad (52)$$

- Pode-se considerar também o intervalo como um par ordenado de pontos limites $a$ e $b$.
- Um intervalo será denotado por uma letra maiúscula.
- Seja $X$ um intervalo. Os pontos limites serão denotados por $\underline{X}$ e $\bar{X}$, tal que $X = [\underline{X},\bar{X}]$.

## Exercício 17

Represente em um plano um intervalor vetorial, tal que $X = [X_1,X_2]$.

# Representações Finitas

- Dois intervalos são iguais se os pontos limites são iguais. Assim, $X = Y$ se $\underline{X} = \underline{Y}$ e $\bar{X} = \bar{Y}$.
- A interseção de dois intervalos $X$ e $Y$ é vazia, $X \cap Y = \emptyset$ se $\underline{X} > \bar{Y}$ ou $\underline{Y} > \bar{X}$.
- Quando a interseção não for um conjunto vazio, o resultado é também um intervalo e é definido por

$$X \cap Y = [\max(\underline{X}, \underline{Y}), \min(\bar{X}, \bar{Y})]. \tag{53}$$

- Se $X \cap Y \neq \emptyset$, a união é definida por

$$X \cup Y = [\min(\underline{X}, \underline{Y}), \max(\bar{X}, \bar{Y})]. \tag{54}$$

- A ordem pode ser estabelecida assim:

$$X < Y \text{ se e somente se } \bar{X} < \underline{Y}. \tag{55}$$

# Representações Finitas

- A inclusão de conjuntos é definida como

$$X \subseteq Y \text{ se e somente se } \underline{Y} \leq \underline{X} \text{ e } \bar{X} \leq \bar{Y}. \tag{56}$$

- A largura do intervalo é definida como

$$w(X) = \bar{X} - \underline{X}. \tag{57}$$

- O valor absoluto de um intervalo $X$ é dado por

$$|X| = \max(\underline{X}, \bar{X}) \tag{58}$$

- O ponto intermediário é dado por

$$m(X) = \frac{\bar{X} - \underline{X}}{2} \tag{59}$$

## Exercício 18

Escreva um exemplo para as seguintes definições: intercesão, união, ordem, inclusão de conjuntos, largura de intervalos, valor absoluto e ponto intermediário.

# Representações Finitas

- A adição de $X + Y = Z$ tal que

$$\underline{Z} = \underline{X} + \underline{Y} \quad (60)$$
$$\bar{Z} = \bar{X} + \bar{Y}. \quad (61)$$

- O negativo de um intervalo é dado por

$$-X = -[\underline{X}, \bar{X}] = [-\bar{X}, -\underline{X}] \quad (62)$$

- A diferença de dois intervalos é definida como:

$$Y - X = Y + (-X) == [\underline{Y} - \bar{X}, \bar{Y} - \underline{X}] \quad (63)$$

- O inverso é definido por

$$1/X = \{1/x : x \in X\}. \quad (64)$$

- Se $0 \notin X$, então

$$1/X = [1/\underline{X}, 1/\bar{X}]. \quad (65)$$

# Representações Finitas

- A multiplicação é definida como $X \cdot Y = [\underline{X \cdot Y}, \overline{X \cdot Y}]$ tal que

$$\underline{X \cdot Y} = \min(\underline{X}\,\underline{Y}, \underline{X}\,\bar{Y}, \bar{X}\,\underline{Y}, \bar{X}\,\bar{Y}) \quad (66)$$

$$\overline{X \cdot Y} = \max(\underline{X}\,\underline{Y}, \underline{X}\,\bar{Y}, \bar{X}\,\underline{Y}, \bar{X}\,\bar{Y}) \quad (67)$$

- Para a divisão de dois intervalos, tem-se:

$$X/Y = X \cdot (1/Y). \quad (68)$$

- A adição e multiplicação são associativas e comutativas.
- Entretanto, a propriedade distributiva não é geral.

**Exercício 19**

Elabore dois exemplos para cada operação aritmética: adição, subtração, multiplicação e divisão.

**Exercício 20**

Seja $X = [1,2]$ e um escalar $a = 1$. Mostre que $X \cdot (a - a) \neq X \cdot a - X \cdot a$.

# Cálculo Finito

## Definition 6

Seja $M_1$ e $M_2$ conjuntos arbitrários e $g : M_1 \to M_2$ um mapeamento arbitrário (função) de $M_1$ para $M_2$. Considere $S(M_1)$ e $S(M_2)$ como as famílias de subconjuntos de $M_1$ e $M_2$ , respectivamente. Chama-se mapeamento do valor-conjunto, $\bar{g} : S(M_1) \to S(M_2)$,

$$\bar{g}(X) = \{g(x) : x \in X, X \in S(M_1)\} \tag{69}$$

a extensão unida de $g$. Pode-se também escrever

$$\bar{g}(X) = \cup_{x \in X}\{g(x)\}. \tag{70}$$

- Em geral, não há uma representação finita para $\bar{g}$. Não se pode calcular, em um número finito de operações aritméticas, o intervalo $\bar{g}(X)$.
- Funções de intervalo racional são casos em que os valores de intervalo são definidos por uma sequência finita de operações aritméticas de intervalos.

# Cálculo Finito

## Exemplo 20

Considere o mapa cujos valores de $F$ são definidos por

$$\begin{aligned} T_1 &= [1,2]X_1, \\ T_2 &= T_1 + [0,1], \qquad (71) \\ F(X_1,X_2) &= T_2X_2. \end{aligned}$$

(72)

Para esta função em particular, tem-se:

$$F(X_1,X_2) = \bar{g}([1,2],[0,1],X_1,X_2) \qquad (73)$$

em que

$$g(c_1,c_2,x_1,x_2) = (c_1x_1 + c_2)x_2. \qquad (74)$$

# Cálculo Finito

## Definition 7

A extensão unida implica na propriedade de subconjunto. Seja $X, Y \in S(M_1)$ com $X \subseteq Y$ implica em $\bar{g}(X) \subseteq \bar{g}(Y)$.

## Definition 8

Um função de intervalo $F$ das variáveis $X_1, X_2, \ldots, X_n$ é monotônica inclusiva se

$$Y_i \subseteq X_i, \quad i = 1,2,\ldots,n,$$

então

$$F(Y_1, Y_2, \ldots, Y_n) \subseteq F(X_1, X_2, \ldots, X_n). \tag{75}$$

# Cálculo Finito

- Todas as operações aritméticas para intervalos são monotônicas inclusivas. Assim, se $Y_1 \subseteq X_1$ e $Y_2 \subseteq X_2$ então

$$\begin{aligned} Y_1 + Y_2 &\subseteq X_1 + X_2, \\ Y_1 - Y_2 &\subseteq X_1 - X_2, \\ Y_1 Y_2 &\subseteq X_1 X_2, \\ Y_1/Y_2 &\subseteq X_1/X_2, \end{aligned} \tag{76}$$

(77)

- Nem todas as funções são monotônicas inclusivas.

**Exercício 21**

Mostre que $F(X) = m(X) + \frac{1}{2}(X - m(X))$ não é um função monotônica inclusiva. Dica: compare os resultados de $X = [0,2]$ e $X = [0,1]$.

# Cálculo Finito

## Definition 9

Seja $f$ uma função no domínio dos reais de $n$ variáveis $x_1, x_2, \ldots, x_n$. Por uma extensão de intervalo de $f$, entende-se uma função de intervalo $F$ com $n$ variáveis $X_1, X_2, \ldots, X_n$ com a propriedade

$$F(x_1, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_n), \quad \text{para argumentos reais.} \tag{78}$$

- Não há uma única extensão de intervalo para $f$. Seja $F(x) = f(x)$, então $F_1(X) = F(X) + X - X$.

## Theorem 2

*Se $F$ é uma extensão de intervalo inclusiva monotônica de $f$, então $\bar{f}(X_1, \ldots, X_n) \subseteq F(X_1, \ldots, X_n)$.*

- Duas expressões racionais podem ser equivalentes na aritmética real e diferentes na aritmética de intervalos.

# Cálculo Finito

## Exercício 22

Seja $f(x) = x(1-x) = x - x \cdot x$. Pode se definir duas aritméticas de intervalo $F_1(X) = X(1-X)$ e $F_2(X) = X - X \cdot X$. Compare $F_1([0,1])$ e $F_2([0,1])$.

## Colorário 1

*Se $F$ é função de intervalo racional e extensão de intervalo de $f$, então $\bar{f}(X_1, \cdots, X_n) \subseteq F(X_1, \cdots, X_n)$.*

## Exercício 23

Considere o polinômio

$$p(x) = 1 - 5x + \frac{1}{3}x^3. \tag{79}$$

Determine a faixa de valores de $p(x)$ quando $x$ pode ser qualquer valor do intervalo $[2,3]$.

# Cálculo Finito

Exemplo p(x)=1-5x+1/3(x^3)
(Moore, 1979) - p. 22

(%i1)

```
f(x):=1-5*x+1/3*(x^3);
```

$$\mathrm{f}\left(x\right) := 1 - 5\,x + \frac{1}{3}\,x^3$$