CÁLCULO NUMÉRICO EM COMPUTADORES

Introdução

Profª. Juliana Eyng

PARTE 1

Representação Digital de Números

- Um número é representado, internamente, em um computador ou máquina de calcular através de uma sequência de impulsos elétricos que indicam dois estados: 0 ou 1 (base 2).

- Se utilizássemos um armazenamento em ponto fixo (vírgula fixa) seria necessário um número de posições (dígitos) no mínimo igual a variação dos limites dos expoentes.

Representação Digital de Números

- Para se obter a representação de uma calculadora científica comum com limites positivos entre $1{,}0.10^{-99}$ e $9{,}999999999.10^{+99}$ seria necessário:

i) Entre $1{,}0.10^{-99}$ e 1 seriam necessárias 99 posições:

$1{,}0.10^{-99}$ = 0,000.....00001

99 dígitos após a vírgula

Representação Digital de Números

ii) Entre 1 e $9,99999999.10^{+99}$ seriam necessárias 100 posições:

$9,99999999.10^{+99}$ = 999999999000.....0000,

100 dígitos inteiros

iii) Seria necessário mais uma posição para o sinal (s), para as representações de negativos, totalizando 200 posições em cada registro:

Representação Digital de Números

S		
	100 posições para a parte inteira	99 posições para a parte frac.

- Por outro lado, em uma representação em Ponto Flutuante, esta calculadora científica funciona com pouco mais de dez dígitos, incluindo posições reservadas ao expoente.

Representação Digital de Números

- Uma representação em Ponto Flutuante, onde a vírgula flutua segundo um certo padrão, temos a seguinte representação genérica na base β:

$$X = \pm [d_1/\beta + d_2/\beta^2 + d_3/\beta^3 + \ldots + d_t/\beta^t] . \beta^{exp}$$

ou

$$X = \pm (0,d_1 d_2 d_3 \ldots d_t)_\beta . \beta^{exp}$$

Representação Digital de Números

Onde:

- d_i = números inteiros contidos em $0 \leq d_i \leq (\beta - 1)$ $(i = 1, 2, \ldots, t)$ que constituem a mantissa;

Obs.: É necessário algum tipo de normalização para padronização da mantissa, no caso adota-se $d_1 \neq 0$.

- exp = expoente de β, assume valores limites I (Inferior) e S (Superior) onde $I \leq exp \leq S$.

- t = número de dígitos significativos do sistema de representação, é chamado de precisão da máquina.

Representação Digital de Números

Padrão 16 bits

- Representação em Ponto Flutuante da vaiável de 16 bits: base binária ($\beta = 2$), com t = 10 dígitos binários (bits) na mantissa e expoentes limitados entre I = -15 e S = + 15 ($15_{10} = 1111_2$)

- simbolicamente: $F(\beta, t, I, S) = F(2, 10, -15, 15)_{10}$. Esta é a representação clássica da *variável de 16 bits*

Representação Digital de Números

Representação Esquemática $F(2, 10, -15, 15)_{10}$

S1											S2				

t dígitos significativos   expoente

S1 sinal da mantissa   S2 sinal do expoente

- Convenciona-se que:

  Se $s_1 = 0 \rightarrow$ número positivo.

  Se $s_1 = 1 \rightarrow$ número negativo.

  s_2 idem.

Representação Digital de Números

- No registro total tem-se:

S1											S2				

- 1 *bit* para sinal da mantissa, *s1*.
- 10 *bits* para armazenar os dígitos significativos da mantissa (t=10), *f*.
- 1 *bit* para sinal do expoente, *s2*.
- 4 *bits* para o módulo do expoente, *exp.*

Totalizando 16 *bits* neste registro

Representação Digital de Números

- Exemplo:

$$x = (-1)^{s1}\,(0,f)_2\,2^{\pm exp}$$

Representar $-(101{,}011)_2$ na variável de 16 *bits* estabelecida anteriormente.

Normalizando $x = -(0{,}1010110000)_2\,2^3$

Convertendo-se o expoente: $(3)_{10} = (0011)_2$

1	1	0	1	0	1	1	0	0	0	0	0	0	0	1	1

Representação Digital de Números

- **Limites da Representação em ponto Flutuante**

a) Menor positivo representável (mp):

0	1	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Lembre-se de que toda representação na máquina de 16 bits usa normalização com padrão $d_1 \neq 0$.

$$mp = +(0{,}1)_2 \,.\, 2^{-15} = (2^{-1} \,.\, 2^{-15})_{10} = (2^{-16})_{10} =$$

$$(0{,}0000152587890625)_{10}$$

Representação Digital de Números

- **Limites da Representação em ponto Flutuante**

b) Maior positivo representável (MP):

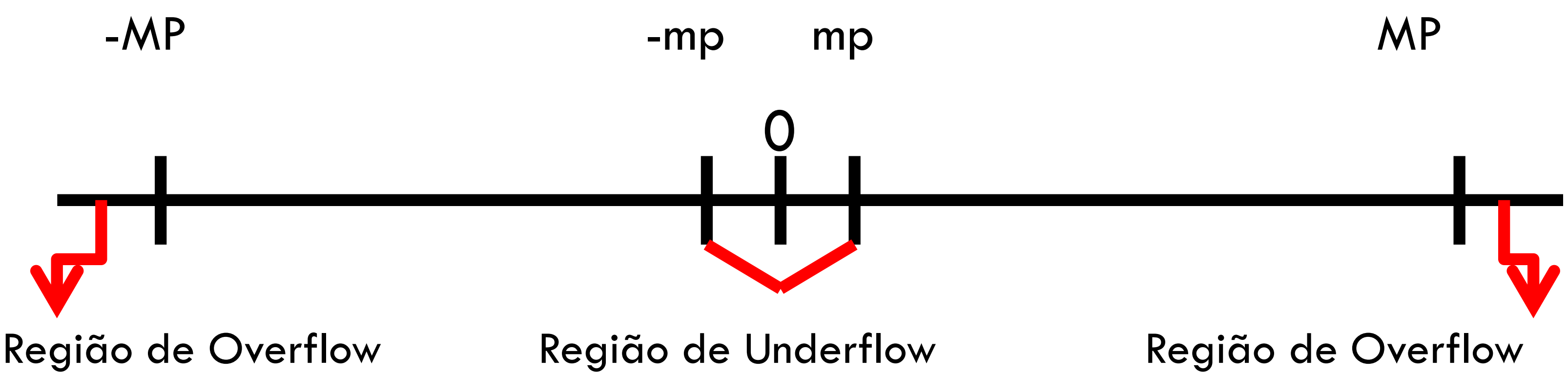
0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	1

$$MP = +(0{,}1111111111)_2 \,.\, 2^{15} = (2^{-1} + 2^{-2} + 2^{-3} + \ldots + 2^{-10})\,.\, 2^{15} = (32736)_{10} \cong (1\,.\,2^{15})$$

Representação Digital de Números

- Os limites de representação dos números negativos são simétricos aos limites positivos apresentados.

Na reta real temos a seguinte representação para F(2, 10, -15, +15):



Representação Digital de Números

- Região de Underflow: $\{x \in R \ / \ -mp < x < mp\}$ que compreende os números, em módulo, abaixo do mínimo representável. Caso aconteçam, são arredondados para o mais próximo entre –mp, zero, +mp
- Região de Overflow: $\{x \in R \ / \ x < -MP \text{ e } x > MP\}$ que compreende os números, em módulo, acima do máximo representável. Não são armazenados (mensagem de erro)

Representação Digital de Números

- **Limites da Representação em ponto Flutuante**

b) Representação do Zero

É obtida com mantissa nula e o menor expoente representável (I).

Representar o zero em F(2,10,-15,+15).

0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

Representação Digital de Números

- **Exercício**

1) Determinar todos os números de ponto flutuante normalizados em F(2,3,-2,+2).

2) Determinar todos os números de ponto flutuante normalizados em F(2,5,-4,+4).

Representação Digital de Números

- Pode-se notar que a distribuição de números representáveis de $F(\beta, t, I, S)$ não é uniforme em $\Re$, e que para cada potência da base β existe uma quantidade fixa de números representáveis dada por:

$$NC = (\beta - 1) \cdot \beta^{t-1}$$

Representação Digital de Números

- Exemplo:

  Em $F(2, 3, -1, +2)$ temos as seguintes representações possíveis:

mantissas possíveis	expoentes possíveis:
0,100	2^{-1}
0,101	2^{0}
0,110	2^{+1}
0,111	2^{+2}

Representação Digital de Números

- Em F(2, 3, -1, +2)
  - Quatro possibilidades de mantissas em cada potência da base

    $((\beta - 1).\beta^{t-1} = 4$ para $\beta = 2$ e $t = 3$
  - Quatro possibilidades de expoentes

    $(S - I + 1 = 4$ para $S = 2$ e $I = -1)$
  - Número total de positivos representáveis (NP = 16).

Representação Digital de Números

- Desta forma o número total de elementos representáveis em uma máquina genérica $F(\beta, t, I, S)$ é dado por:

$$NF(\beta, t, I, S) = 2.(S - I + 1).(\beta - 1).\beta^{t-1} + 1$$

- Incluindo os positivos, negativos e o zero.

Representação Digital de Números

- **Exemplo:**

  Em F(2, 10, -15, +15) (variável de 16 *bits*) temos:

  $NF = 2. (15 - (- 15) + 1) . (2 - 1) . 2^{10-1} + 1$

  $= 31745$ elementos

  incluindo os positivos, negativos e o zero.

Representação Digital de Números

- **Exemplo:**

  Em F(10, 10, -99, +99) (calculadora científica comum) temos:

  $NF = 2.\ (99 - (-\ 99) + 1)\ .\ (10 - 1)\ .\ 10^{10-1}\ + 1$

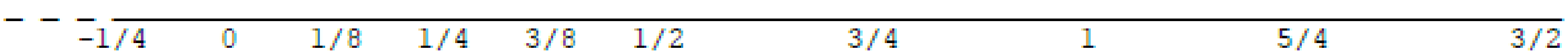
  $= 3582\ .\ 10^{9} + 1$ elementos

Representação Digital de Números

- **Exercício:**
- Liste todos os números que podem ser representados na forma F(2, 2, -1, +1) :

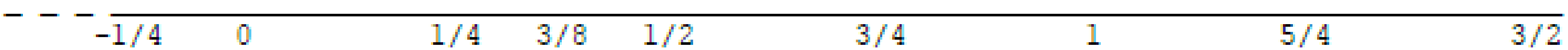
$0{,}00.2^{-1} = 0$	$0{,}01.2^{-1} = 1/8$	$0{,}10.2^{-1} = 1/4$	$0{,}11.2^{-1} = 3/8$
$0{,}00.2^{0} = 0$	$0{,}01.2^{0} = 1/4$	$0{,}10.2^{0} = 1/2$	$0{,}11.2^{0} = 3/4$
$0{,}00.2^{+1} = 0$	$0{,}01.2^{+1} = 1/2$	$0{,}10.2^{+1} = 1$	$0{,}11.2^{1} = 3/2$

Representação esquemática:



Representação Digital de Números

Admitindo apenas os números em ponto flutuante normalizado:



Há uma quantidade finita de números distribuídos de forma desigual

Qualquer número próximo ao zero, menor que ¼ underflow (-1/4 < x< 1/4)

Qualquer número fora da faixa de -3/2 a 3/2 overflow

(x<-1,5 ou x>1,5)

Representação Digital de Números

- **Exercício:**

  Na variável F(2, 3, -3, +3) com $d_1 \neq 0$ calcule:

  - O número de elementos representáveis;
  - Esquematize a representação de todos os elementos positivos na base 2;
  - Defina as regiões de underflow e overflow.

Representação Digital de Números

- **Exercício:**

  Considere o sistema F(2, 5, -3, 1) com $d_1 \neq 0$ calcule:
  - Quantos números podemos representar neste sistema?
  - Qual o maior número na base 10 que podemos representar neste sistema?
  - Defina as regiões de underflow e overflow?

Padrão IEEE 754

- Quem define o padrão de representação em ponto flutuante são os tipos de variáveis usadas em programação. Em cada padrão são definidos os limites (overflow e underflow), a **normalização** e a **polarização**
- O padrão mais utilizado é o IEEE 754

4 bytes = 32 bits	single (float)
8 bytes = 64 bits	double
10 bytes = 80 bits	extended (long double)

Padrão IEEE 754

- Polarização ou excesso é um número que é somado (em excesso) ao expoente para torná-lo sempre positivo, ampliando o valor do expoente superior.

- Tem como objetivo, ampliar a representação e reduzir as regiões de underflow e overflow.

Padrão IEEE 754

- **Variável de 16 bits**

Na variável de 16 bits F (2, 10, -15,+15) podemos usar uma polarização p=+15 =+ $(1111)_2$

Menor nº: -15+p = -15+15 = 0

Maior nº: +15+p = +15+15 = 30 $(11110)_2$

O que permite que o máximo expoente seja

$(11111)_2 = (31)_{10}$ logo, F(2, 10, 0, 31)

$$x = (-1)^s (0,f)_2\ 2^{exp-15}$$

Padrão IEEE 754

- Exemplo: Qual o decimal representado no registro binário a seguir?

0	1	1	0	1	0	0	0	0	0	0	1	0	0	1	0

- Expoente $(10010)_2 = (18)_{10}$
- $+(0{,}1101000000)_2 \,.\, 2^{18-15} = (110{,}1)_2 = (6{,}5)_{10}$
- $x = (-1)^0\,(0{,}1101000000)_2\; 2^{18-15}$

Padrão IEEE 754

- O não armazenamento do primeiro bit não nulo da mantissa é outra otimização pelo padrão IEEE 754
- $x = (-1)^s \, (1,f)_2 \, 2^{exp-15}$

0	1	1	0	1	0	0	0	0	0	0	1	0	0	1	0

- Expoente $(10010)_2 = (18)_{10}$
- $+(1,1101000000)_2 \, . \, 2^{18-15} = (1110,1)_2 = (14,5)_{10}$

Padrão IEEE 754

- Com a polarização e com o primeiro bit implícito, o valor de MP foi ampliado para:

0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0

- $+(1,1111111111)_2 \cdot 2^{30-15} = (2^0 + 2^{-1} + 2^{-2} + 2^{-3} + \ldots + 2^{-10}) \cdot 2^{15} = (65504)_{10} \cong (2^{16})_{10}$
- MP era $(32736)_{10}$

Padrão IEEE 754

- Flexibilidade na normalização da mantissa para ampliar a faixa de abrangência de números pequenos como para o primeiro número positivo mp, reduzindo a região de undeflow.
- Com $d_1 \neq 0$ implícito, mp seria:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- mp = $+(1{,}0000000000)_2 \cdot 2^{0-15} = (2^{-15})_{10}$ maior que o anterior $(2^{-16})_{10}$

Padrão IEEE 754

- A normalização com $d_1 \neq 0$ implícito foi eliminada, a mantissa pode assumir qualquer valor, mas o valor mínimo do expoente não polarizado deve ser -14 e mp se torna:

0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

- $mp = +(0{,}0000000001)_2 \;.\; 2^{0-14} = 2^{-10}\, 2^{-14} = (2^{-24})_{10}$
- mp era $(2^{-16})_{10}$

Padrão IEEE 754

- O novo padrão otimizado para a variável de 16 bits ficou:

s	-	e	x	p	-	-	-	-	-	f	-	-	-	-	-

- s = 1 bit sinal da mantissa
- exp polarizado $(15)_{10} = (01111)_2$ = 5 bits
- f = mantissa a partir de d_2 ($d_1 = 1$ e não armazenado) = 10 bits

Padrão IEEE 754

- Um número x armazenado nesse registro é interpretado conforme o valor do expoente polarizado da seguinte forma:

  Se $0 < exp < 31$ então $x = (-1)^s\, 2^{exp-15}(1,f)_2$

  Se $exp = 0$ e $f \neq 0$ então $x = (-1)^s\, 2^{-14}(0,f)_2$

  Se $exp = 0$ e $f = 0$ então $x = (-1)^s\, 2^{-14}(0,0)_2 = zero$

  Se $exp = 31$ então x está na região de overflow

Padrão IEEE 754

- **Variável de 32 bits**

Na variável de 32 bits F (2, 23, -127,+127) podemos usar uma polarização p=+127 =+ $(1111111)_2$

Menor nº: -127+p = -127+127 = 0

Maior nº: +127+p = +127+127 = 254 $(01111111)_2$

O que permite que o máximo expoente seja

$(1111111)_2 = (255)_{10}$ logo, F(2, 23, 0, 255)

Padrão IEEE 754

- **Variável de 32 bits**
- s = 1 bit sinal da mantissa
- exp polarizado $(127)_{10} = (01111111)_2$ = 8 bits
- f = mantissa a partir de d_2 (d_1 = 1 e não armazenado) = 23 bits

  Se $0 < exp < 255$ então $x = (-1)^s \, 2^{exp-127}(1,f)_2$

  Se $exp = 0$ e $f \neq 0$ então $x = (-1)^s \, 2^{-126}(0,f)_2$

  Se $exp = 0$ e $f = 0$ então $x = (-1)^s \, 2^{-126}(0,0)_2$ = zero

  Se $exp = 255$ então x está na região de overflow

Padrão IEEE 754

- **Variável de 64 bits**

Na variável de 64 bits F (2, 52, -1023,+1023) podemos usar uma polarização p=+1023 =+ $(1111111111)_2$

Menor nº: - 1023 +p = - 1023 + 1023 = 0

Maior nº: + 1023 +p = + 1023 + 1023 = 2046 $(01111111111)_2$

O que permite que o máximo expoente seja

$(11111111111)_2 = (2047)_{10}$ logo, F(2, 52, 0, 2047)

Padrão IEEE 754

- **Variável de 64 bits**
- s = 1 bit sinal da mantissa
- exp polarizado p=+1023 =+ $(1111111111)_2$ = 10 bits
- f = mantissa a partir de d_2 (d_1 = 1 e não armazenado) = 52 bits

Se $0 < exp < 2047$ então $x = (-1)^s\, 2^{exp-1023}(1,f)_2$

Se $exp = 0$ e $f \neq 0$ então $x = (-1)^s\, 2^{-1022}(0,f)_2$

Se $exp = 0$ e $f = 0$ então $x = (-1)^s\, 2^{-1022}(0,0)_2$ = zero

Se $exp = 2047$ então x está na região de overflow

Erros

-
- **Erro Absoluto:**

  Diferença entre o valor exato de um número *x* e seu valor aproximado $\bar{x}$ obtido a partir de um procedimento numérico.

$$EA_x = |x - \bar{x}|$$

- Em geral apenas *x* é conhecido, e o que se faz é assumir um limitante superior (K1 majorante) ou uma estimativa para o módulo do erro absoluto.

$$|EA_x| \leq k1$$

Erros

-
- **Exemplos (erro absoluto)**

Seja x *representado por* $\overline{x} = 2112{,}9$ *de forma que*

$$|EA_x| < 0{,}1$$

podemos dizer que $x \in (2112{,}8; 2113{,}0)$.

Seja y *representado por* $\overline{y} = 5{,}3$ *de forma*

que $|EA_x| < 0{,}1$

podemos dizer que $y \in (5{,}2;\ 5{,}4)$

Erros

- **Erro Relativo ou Taxa de Erro**

  Erro relativo de *x é o módulo do quociente entre o erro absoluto* E_{ax} *e o valor exato x ou o valor aproximado* $\overline{x}$.

$$E_{Rx} = \left| \frac{E_{ax}}{x} \right| = \left| \frac{x - \overline{x}}{x} \right| \quad ou \quad E_{Rx} = \left| \frac{E_{ax}}{\overline{x}} \right| = \left| \frac{x - \overline{x}}{\overline{x}} \right|$$

- Taxa de erro é $E_{Px} = E_{RX} .100$

Erros

- **Exemplo (erro relativo)**

- Seja *x representado por* $\bar{x}$ *= 2112,9 de forma que* $|E_{ax}|$ *< 0,1*

$$E_{Rx} = \left|\frac{E_{ax}}{\bar{x}}\right| = \frac{0,1}{2112,9} = 4,7.10^{-5}$$

$$E_{px} = 4,7.10^{-5}.100 = 0,0047\%$$

Erros

- **TIPOS DE ERROS EXISTENTES EM MÁQUINAS DIGITAIS**
  - **Erros Inerentes:** são aqueles existentes nos dados de entrada de um software numérico. Decorre, por exemplo, de medições experimentais, de outras simulações numéricas, ...
  - **Erros de truncamento:** ocorrem quando quebramos um processo matematicamente infinito, tornando-o finito, por incapacidade de execução ou armazenamento.

Erros

- **TIPOS DE ERROS EXISTENTES EM MÁQUINAS DIGITAIS**
  - **Erros de Arredondamento:** ocorrem quando são desprezados os últimos dígitos que, ou não são fisicamente significativos na representação numérica, ou estão além da capacidade de armazenamento na máquina digital.

Erros

-
- **EXEMPLOS:**
  - **Erro de truncamento:**
    - Aproximar π truncando na quarta casa decimal, sendo que $\pi =$ 3,1415926535...
    - Sabendo-se que e^x pode ser escrito como:

$$e^x = \sum_{i=0}^{\infty} \frac{x_i}{i!}$$

faça a aproximação de e^2 através de um truncamento após quatro termos da somatória.

Erros

- **EXEMPLOS:**
  - **Erro de Arredondamento:**
    - Arredondar π na quarta casa decimal, sendo que $\pi =$ 3,1415926535...

Erros

- **Erros de Arredondamento e Truncamento**

Considere a variável F(10, 3, -4, +4)

x	Repres. por arredondamento	Repres. por Truncamento
1,25	$0{,}125 \times 10$	$0{,}125 \times 10$
10,053	$0{,}101 \times 10^2$	$0{,}100 \times 10^2$
-238,15	$-0{,}238 \times 10^3$	$-0{,}238 \times 10^3$
2,71828	$0{,}272 \times 10$	$0{,}271 \times 10$
0,000007	Exp< -4 (underflow)	Exp < -4 (underflow)
718235,82	Exp > 4 (overflow)	Exp > 4 (overflow)

Erros

- Quando se utiliza o arredondamento os erros cometidos são menores que no truncamento, no entanto o arredondamento requer um maior tempo de execução e por esta razão o truncamento é mais utilizado.

Erros

- **Consequências dos erros de arredondamento**:

(a) Perda de significação: Esta é uma consequência de erros de arredondamento, que gera perda, total ou parcial, de dígitos significativos.

Exemplo:

Efetue a soma de a = 0,0135 e b = 10,51 em F(10,4,-10,+10)
Representação em ponto flutuante:

$a = 0{,}1350 \cdot 10^{-1}$ $\qquad$ $b = 0{,}1051 \cdot 10^{2}$

Erros

A adição em aritmética de ponto flutuante requer o alinhamento dos pontos decimais dos dois números. Para isto, a mantissa do número de menor expoente deve ser deslocado para a direita.

$a = 0{,}1350 \,.\, 10^{-1}$ $\qquad$ $b = 0{,}1051 \,.\, 10^{2}$

$a = 0{,}0001\underline{35} \,.\, 10^{2}$

$+$ $\qquad$ Soma de parcelas de grandezas muito diferentes

$b = 0{,}1051 \,.\, 10^{2}$

$a + b = 0{,}1052 \,.\, 10^{2}$

Erros

Representação do zero em F(10, 4, -10, +10)

$a = 0,1350.10^{-3}$ $b = 0,0 \, . \, 10^{0}$ $b = 0,0 \, . \, 10^{-10}$

Expoente diferente do inferior

$$
\begin{array}{r}
0,0001350.10^{0} \\
0,0000.\ 10^{0} \\
\hline
0,0001\underline{35} \, . \, 10^{0}
\end{array}
$$

Expoente da máquina

$$
\begin{array}{r}
0,1350.10^{-3} \\
0,0000.10^{-3} \\
\hline
0,1350 \, . \, 10^{-3}
\end{array}
$$

Em calculadoras dígitos internos de “guarda”

Erros

- **Consequências dos erros de arredondamento**:

(b) Instabilidade numérica: A acumulação sucessiva de erros de arredondamento pode conduzir um algoritmo de repetição a resultados absurdos.

Exemplo:

Avalie f(x) em x = 3 e em x = 3,00001 utilizando uma calculadora científica com representação de 10 dígitos.

$$f(x) = \frac{27985}{9{,}1 - x^2}$$

Erros

f(3) = 279850,00

f(3.00001) = 280018,0108

- Note que há uma variação no resultado final de f(x).
- Isto caracteriza uma instabilidade intrínseca do modelo matemático em relação aos seus dados de entrada.

Precisão x Exatidão

- **PRECISÃO**: estabelece a quantidade de algarismos significativos que representam um número.

- A precisão de uma máquina digital é definida como o número de dígitos t da mantissa na base β, e a precisão decimal “d ” equivalente pode ser definida baseada na equivalência entre as variações dos dígitos menos significativos em cada base.

Precisão x Exatidão

- **PRECISÃO**
- Existe uma equivalência entre a representação binária de t+1 bits totais e a decimal de d decimais:

  $10^{1-d} = \beta^{-t}$

  $\log(10^{1-d}) = \log(\beta^{-t})$

  $1 - d = (-t) \log \beta$

  $d = 1 + t \log \beta$

Precisão x Exatidão

- Exemplo: Calcule a precisão decimal equivalente da variável de 16 *bits*

  F(2, 10, -15, 15) $\beta = 2$ t = 10

  d = 1 + 10. (log 2)

  d = 4,0103

  temos uma precisão decimal equivalente d = 4 decimais significativos

Precisão x Exatidão

- Exemplo: Considere a variável de 32 bits

  F(2, 23, -127, 127)

  $2^{-23} = 10^{1-d} \Rightarrow \log 2^{-23} = \log 10^{1-d}$

  $1 - d = -23 \log 2 \Rightarrow d = 1 + 23 \log 2 \approx 7{,}9$

  7 e 8 decimais significativos

- Ou seja, uma calculadora científica de 8 decimais seria suficiente para armazenar os t+1 bits de forma exata.

Precisão x Exatidão

- A representação é discreta
- Nem todas as frações decimais têm representação binária finita
- Significa que todas as representações binárias da máquina estão “corretas” para 8 dígitos significativos na base 10, ou seja, apresentam decimais equivalentes com pelo menos 8 dígitos corretos.

Precisão x Exatidão

- **EXATIDÃO**: conceito relacionado com a forma que melhor representa uma grandeza numérica, ou seja, uma representação é mais exata quando tem o menor desvio (erro) em relação ao valor exato.

- Representar o π (3,1415926535) :

  (a) 3,14 $\rightarrow$ precisão de dois dígitos 0,00159

  (b) 3,141 $\rightarrow$ precisão de três dígitos 0,00059

  (c) 3,1416 $\rightarrow$ precisão de cinco dígitos 0,000007