INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SANTA CATARINA
CAMPUS FLORIANÓPOLIS
DEPARTAMENTO ACADÊMICO DE ELETRÔNICA

COMPUTAÇÃO CIENTÍFICA

Prof. Marco Valério Miorim Villaça
Versão 2019/1

Ementa da disciplina (1)

- Erros em representações numéricas e aritmética em ponto flutuante.
- Zeros de funções reais: método de bisseção, *Newton-Raphson* e secante.
- Resolução de sistemas de equações lineares por métodos diretos e iterativos.
- Resolução de sistemas não lineares.
- Resolução numérica de equações diferenciais ordinárias, método de *Runge-Kutta* de 4ª ordem.

Ementa da disciplina (2)

- Ajuste de curvas: método dos mínimos quadrados, método polinomial e linearização.
- Interpolação polinomial (forma de Lagrange e forma de Newton), interpolação inversa, escolha do polinômio interpolador, fenômeno de Runge e funções Spline.
- Integração numérica: regra do trapézio, regra de Simpson, estudo sobre erros e integração dupla.

O que se espera do estudante?

- COMPETÊNCIAS:

  Conhecer e aplicar os métodos e algoritmos utilizados para resolução numérica de problemas científicos.

- HABILIDADES:

  Implementar e utilizar algoritmos necessários para a resolução computacional de problemas de cálculo diferencial e integral da área de Engenharia, trabalhosos ou sem solução teórica.

Sugestão de bibliografia básica

[1] CHAPRA, S. C. **Métodos numéricos aplicados com Matlab para engenheiros e cientistas.** Porto Alegre: AMGH, 2013.

[2] RUGGIERO, M. A. G. **Cálculo numérico: aspectos teóricos e computacionais.** São Paulo: Pearson, 1996.

[3] MARROSO, L. C. **Cálculo numérico (com aplicações).** São Paulo: Harbra, 1987.

[4] ARENALES, S. **Cálculo numérico: aprendizagem com apoio de software.** São Paulo: Cengage, 2010.

Sugestão de bibliografia complementar

[4] FRANCO, N. B. **Cálculo numérico.** São Paulo: Pearson, 2006.

[5] GUSTAFSSON, B. **Fundamentals of scientific computing.** 1. ed. Berlin: Springer, 2011.

[6] QUARTERONI, A.; SALERI, F. GERVÁSIO, P. **Scientific computing with Matlab and Octave.** 3. ed. Berlin: Springer, 2010.

Avaliação

- **Avaliações teóricas:** duas, P1 e P2.
- **Avaliação prática (computador):** P3
- **Média das Tarefas: T –** mínimo 2, máximo 4.
- **Recuperações: Teórica** (R1), Prática (R2)
- **Cálculo da Média (X):**

  $X = 0{,}25\ P1 + 0{,}25\ P2 + 0{,}3\ P3 + 0{,}2\ T$
- Calculo da média para alunos em recuperação.

  $X_{REC} = 0{,}5\ R1 + 0{,}5\ R2$
- **Condições de aprovação:**

  X >= 6 **“e”** Faltas <= 0,25* (número de aulas dadas)

Capítulo I

Erros em representações numéricas
e
aritmética em ponto flutuante

Capítulo I

Parte I
Erros e erros de arredondamento

Exatidão e Precisão
Instrumentação

- Exatidão (acurácia): Refere-se a quão próximo um valor medido está do valor real.
- Precisão (repetibilidade): Refere-se a quão próximo um valor medido está de outro valor medido.
- Inexatidão: Sistemático desvio do real.
- Imprecisão ou incerteza (dispersão): magnitude do espalhamento

Exatidão e Precisão
Instrumentação

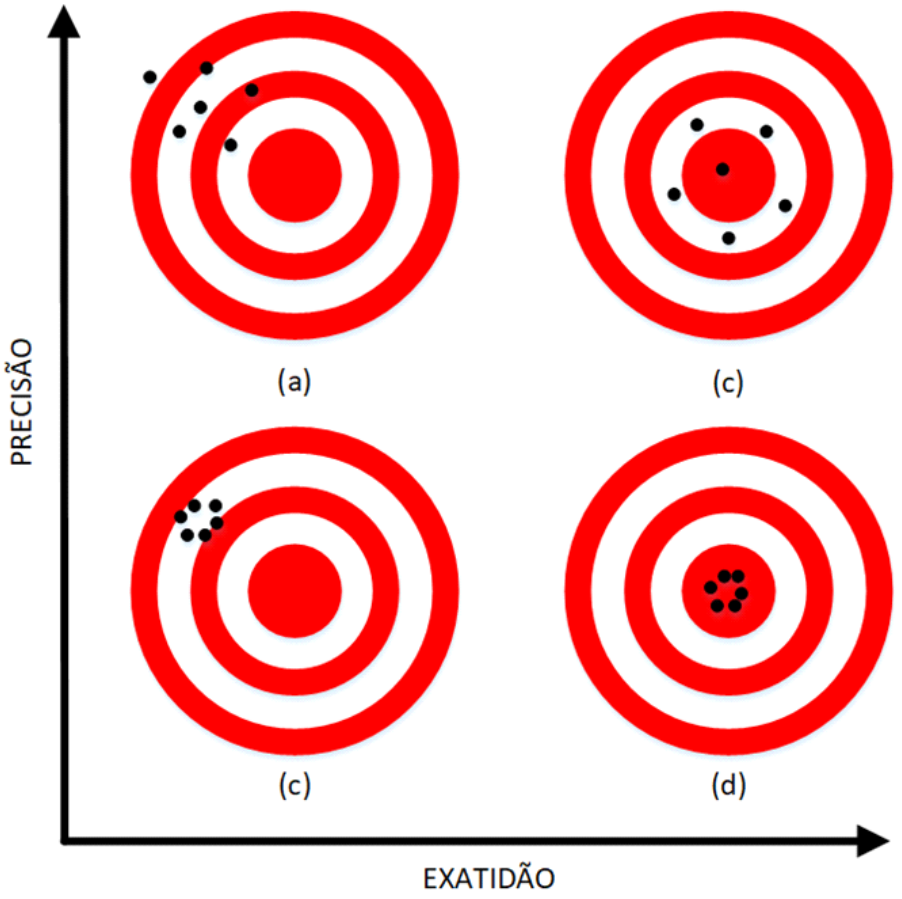
Na figura ao lado, os buracos no alvo correspondem as medidas realizadas e a "mosca" o valor real:

(a) Inexato e impreciso;

(b) Exato e impreciso;

(c) Inexato e preciso

(d) Exato e Preciso



Exatidão e Precisão
Computação Científica

- Precisão: refere-se ao número de dígitos utilizados nas operações aritméticas básicas.
- Exatidão: refere-se ao erro absoluto ou relativo de uma solução numérica.

Definições de erro

- Erro real ou absoluto

$$E_t = valor\ \ real - valor\ \ aproximado \quad (1.1)$$

  - O problema desta definição de erro é que ela não leva em conta a magnitude do valor em exame.

- Erro relativo real

$$\varepsilon_t = \frac{valor\ \ real - valor\ \ aproximado}{valor\ \ real} \quad (1.2)$$

O erro relativo pode também ser multiplicado por 100 para ser expresso percentualmente

$$e_t = \frac{valor\ \ real - valor\ \ aproximado}{valor\ \ real} x\,100\,\% \quad (1.3)$$

Definições de erro

- A informação do valor real raramente está disponível.
  - Para métodos numéricos, o valor real só será conhecido quando se trabalha com funções que podem ser resolvidas analiticamente
- Quando não se conhece o valor real a priori, a alternativa é normalizar o erro usando a melhor estimativa disponível do valor real:

$$\varepsilon_a = \frac{erro \quad aproximado}{valor \quad aproximado} x\,100\,\% \quad (1.4)$$

ou seja, ε_a é o **erro aproximado relativo.**

Definições de erro

- Em processos computacionais iterativos:

$$\varepsilon_a = \frac{valor\ \ aproximado\ \ atual - valor\ \ aproximado\ \ anterior}{valor\ \ aproximado\ \ atual} x\,100\% \quad (1.5)$$

O erro pode ser positivo ou negativo. Geralmente, se está interessado se o valor absoluto do erro relativo percentual é menor que uma tolerância pré-definida, sendo a computação repetida até:

$$|\varepsilon_a| < \varepsilon_s \quad (1.6)$$

relação conhecida como *critério de parada*.

Definições de erro

- Se um número é correto para *n* algarismos significativos, é evidente que o erro absoluto não pode ser superior a metade de uma unidade no *n-ésimo* lugar.
- Por exemplo, se o número 4,629 for correto para quatro algarismos, o erro absoluto não é superior a 0,001 X (1/2) = 0,0005.

Definições de erro

- Scarborough (1930, p. 7) afirma que se o erro relativo de qualquer número não é maior que $1/(2 \cdot 10^n)$, o número está com certeza correto para n algarismos significativos.
- Assim, se o critério abaixo for obedecido, assume-se que o resultado alcançado é correto para pelo menos *n* algarismos significativos.

$$\varepsilon_s = 0{,}5 \cdot 10^{2-n} \ \% \quad (1.7)$$

Exemplo

- As funções matemáticas frequentemente podem ser representadas por séries de potências (séries de Maclaurin).
- É o caso da função e^x:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ \ldots \ + \frac{x^n}{n!} \quad (1.8)$$

Exemplo

- Se for estabelecido um critério de erro que garanta que o resultado está correto para pelo menos 3 algarismos significativos de (1.7) resulta:

$$\varepsilon_s = 0{,}5 \cdot 10^{2-3} = 0{,}05\,\%$$

- Então, o número de termos calculados deve ser o suficiente para que o $|\varepsilon_a|$ caia abaixo deste nível.

Exemplo

- O valor real de $e^{0,5}$ é 1,64872127. Se forem utilizados 3 termos para avaliar $e^{0,5}$, utilizando a equação (1.8) resulta:

$$e_3^{0,5} = 1 + 0,5 + \frac{0,5^2}{2} = 1,625$$

- Com 4 termos fica:

$$e_4^{0,5} = 1,625 + \frac{0,5^3}{6} = 1,645833333$$

- O erro aproximado relativo será:

$$\varepsilon_a = \frac{1,64583333 - 1,625}{1,64583333} x\, 100 = 1,27\ \%$$

- O erro real relativo será:

$$\varepsilon_t = \frac{1,64872127 - 1,64583333}{1,64872127} x\, 100 = 0,175\ \%$$

Exemplo

- Pelo critério de parada (1.6), o processo de computação continuará até:

$$|\varepsilon_a| < 0{,}05\%$$

A tabela abaixo resume o processo de computação.

Termo	Resultado	ε_t %	ε_a %
1	1	39,3	-
2	1,5	9,02	33,3
3	1,625	1,44	7,69
4	1,64583333	0,175	1,27
5	1,64843750	0,00172	0,158
6	1,64869791	0,00142	0,0158

Exemplo

Escrever uma função *Scilab* que calcule uma função matemática por séries de potências.

Teste seu código com a função *cos x*, com *x* real, utilizando a seguinte série de potências:

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

Com condição de parada expressa por (1.7) para pelo menos 4 algarismos significativos.

- Lembre que o valor de x deve ser em radianos.
- Durante o processo de cálculo, compute o resultado e os erros relativos percentuais real e aproximado a cada passo.

Exemplo - código

```
function [fx, term]=series(fun, x, n_sig)
//
// Cálculo de funções com serie de potências
// function [fx,term]=series(funcao, x, n_sig)
// onde fx é o valor da função e x
//      term é o número de termos usados para o erro especificado
//      fun é a função de entrada literal em x e n
//      x é o número no qual se deseja obter o valor da função
//      n é o numero do termo
//      n_sig é o numero de alg. significativos da resposta -
//      opcional
// Exemplo de chamada:
// exec('path\series.sci',-1)
// fun ='(-1)^(n)*x^(2*n)/factorial(2*n)'
// [fx,term]=series(fun, %pi/3, 3)
//
```

Exemplo - código

```
if argn(2) < 3 then
      n_sig = 3;
end
//
// argn fornece os números de parâmetros de entrada (argn(2)) e
// saída (argn(1)) passados à função quando esta é chamada. Aqui
// está sendo usada para lidar com parâmetros opcionais.
//
es = 0.5*10^(2-n_sig); // condição de parada relativa %
fx_old = 0; fx = 0;n = 0; vreal=cos(x);
printf("Termo      fx       et %%      erro %%\n");
```

Exemplo - código

```
while 1 do
      fx = fx + evstr(fun);
      erro = abs((fx - fx_old)/fx)*100;
      errot = abs((vreal - fx)/vreal)*100;
      printf("%-8d %10.6f %10.6f %10.6f\n",n+1,fx,errot,erro);
      if(erro <= es) then
          break;
      end
      fx_old = fx;
      n = n + 1;
end
term = n+1;
endfunction
```

Exercício

- O método “divisão e média”, um antigo método para a aproximação da raiz quadrada de um número positivo a, pode ser formulado por:

$$x_{i+1} = \frac{x_i + a / x_i}{2}$$

Com i = 0, 1, 2, …, n *e* condição de parada expressa por (1.7) para pelo menos 4 algarismos significativos, escreva um *script scilab* para implementar este método, onde:

x_0 é o valor inicial estimado para a raiz de *a*

Erros de arredondamento

- Números como o π e $\sqrt{2}$ não podem ser expressos por um número fixo de algarismos significativos. Logo eles não podem ser representados com exatidão por computadores, pois os mesmos apresentam uma palavra de tamanho fixo.
- A discrepância introduzida pela omissão de algarismos significativos é chamada de erro de arredondamento.

Erros de arredondamento

- Se π = 3,141592653... deve ser armazenado em um sistema decimal com apenas 7 algarismos significativos, surgem duas possibilidades:
  - **Chopping** (corte):

    $\pi = 3{,}141592 \rightarrow$ erro: $E_t = 0{,}00000065$
  - **Rounding** (arredondamento):

    $\pi = 3{,}141593 \rightarrow$ erro: $E_t = -0{,}00000035$
- Os números seguintes são arredondados aplicando o critério mais conhecido de arredondamento:

  3,6543 = 3,65; 0,49781 = 0,498

  22,65 = 22,6; 1,735 = 1,74

Erros de arredondamento
Representação em ponto flutuante

- Similar a notação científica, na representação em ponto flutuante um número real é expresso por:

$$\pm s \cdot b^e$$

onde s = significando

b = base do sistema numérico

e = expoente

- Uma forma normalizada elimina os zeros imediatamente a direita do ponto decimal :

$$0.0456 = 0{,}456 \; x \; 10^{-1}$$

Erros de arredondamento

Implicações da representação em ponto flutuante

- Suponha um computador hipotético de base 10 e palavra de 5 dígitos, onde um dígito é reservado para o sinal, dois para o expoente e dois para o significando. Um dos dígitos do expoente é reservado para o seu sinal.
- A representação normalizada do número será:

$$s_1 0.d_1 d_2 \times 10^{s2d3}$$

onde

- s_1 e s_2 são sinais
- d_3 a magnitude do expoente
- d_1 e d_2 a magnitude do significando

Erros de arredondamento

Implicações da representação em ponto flutuante

- Nesse computador:
  - O maior valor positivo representado é

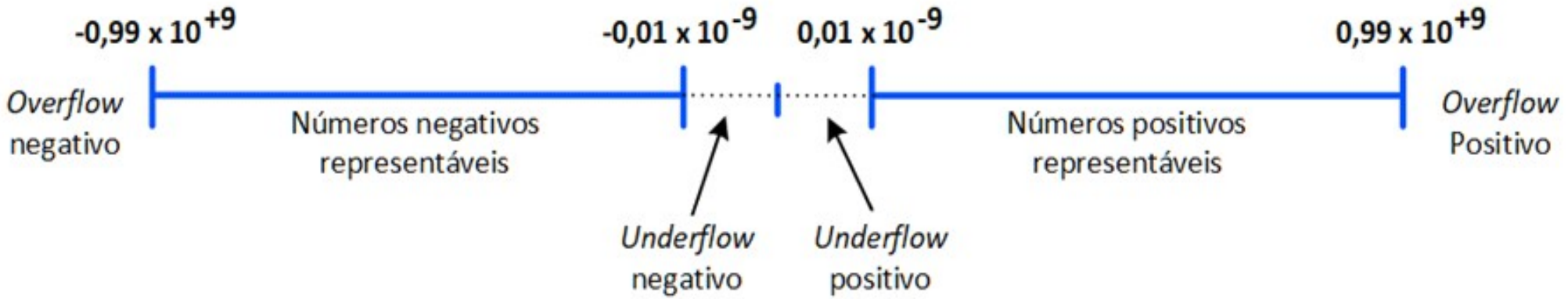
    $+0{,}99x\ 10^{+9} = +9{,}9\ x\ 10^{+8}$
  - O menor valor positivo representado é

    $+0{,}01x\ 10^{-9} = +1{,}0\ x\ 10^{-11}$

Erros de arredondamento

Implicações da representação em ponto flutuante

- O Diagrama abaixo mostra todos os possíveis números representados pelo computador hipotético.



Erros de arredondamento

Implicações da representação em ponto flutuante

- Números positivos ou negativos muito grandes saem da faixa de representação causando um erro por *overflow.*
- Os números muito pequenos, que não podem ser representados, usualmente são convertidos em zero.

Erros de arredondamento

Implicações da representação em ponto flutuante

- A utilização de apenas dois dígitos para o significando afeta dramaticamente a exatidão:
  - O número 0,01845 seria armazenado como 0,18 x 10^{-1} gerando um erro relativo de

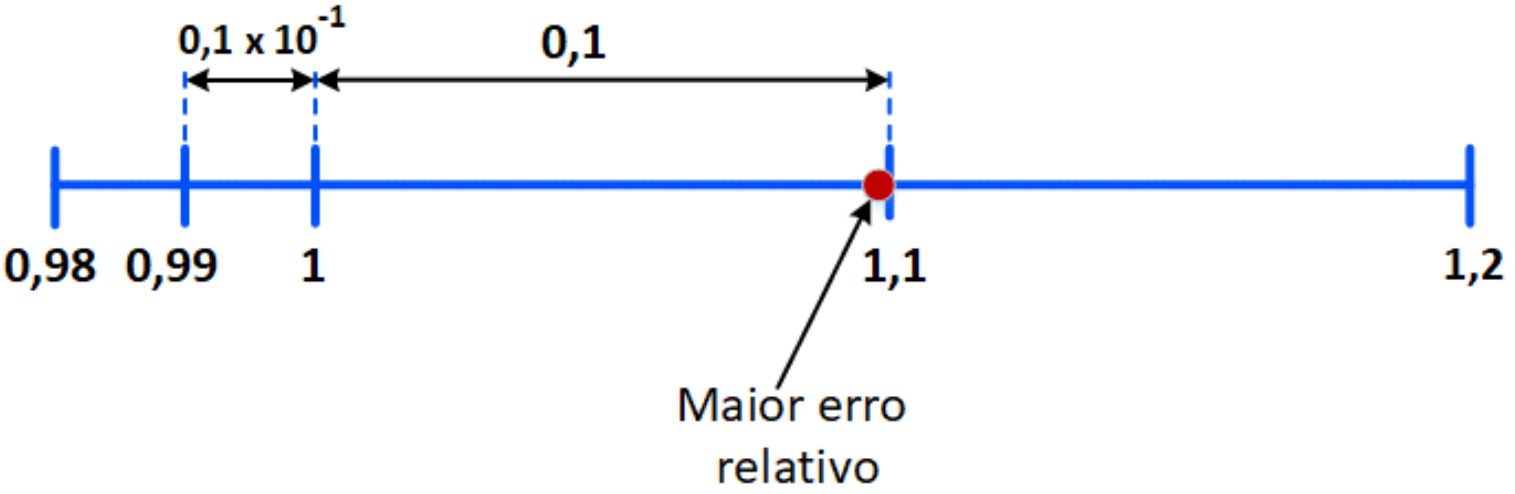
$$e_t = \frac{0{,}01845 - 0{,}018}{0{,}01845} x\, 100 = 2{,}44\ \%$$

- Embora seja possível armazenar exatamente um número como 0,01845, expandindo-se os dígitos do significando, quantidades com infinitos dígitos sempre devem ser aproximadas, como no caso do π.
- Logicamente quanto maior o número de dígitos do significando, melhor será essa aproximação. Entretanto, tais números sempre apresentarão um erro de arredondamento.

Erros de arredondamento

Implicações da representação em ponto flutuante

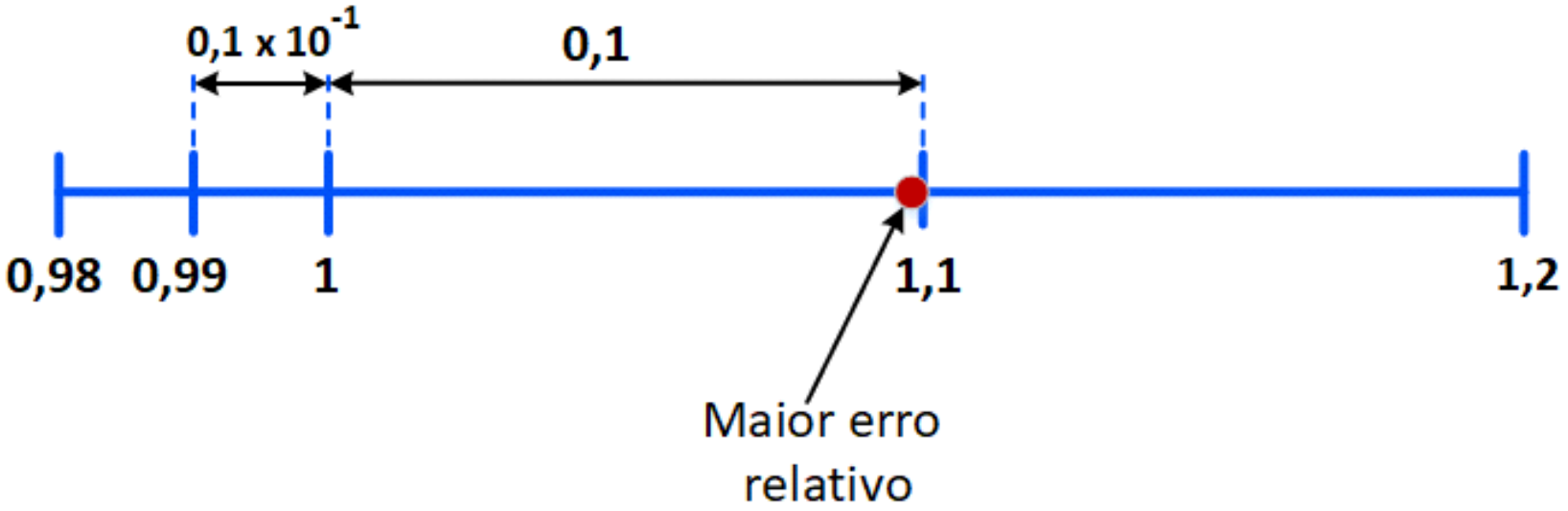
- Outro efeito mais sutil da representação em ponto flutuante é mostrado pelo diagrama abaixo.
- Observa-se que o intervalo entre números incrementa quando se move entre faixas de magnitude:
  - Entre 0,1 e 1, o intervalo é de $0{,}01 = 0{,}1 \times 10^{-1}$
  - Entre 1 e 10, o intervalo é de $0{,}1 = 0{,}1 \times 10^{0}$



Erros de arredondamento

Implicações da representação em ponto flutuante

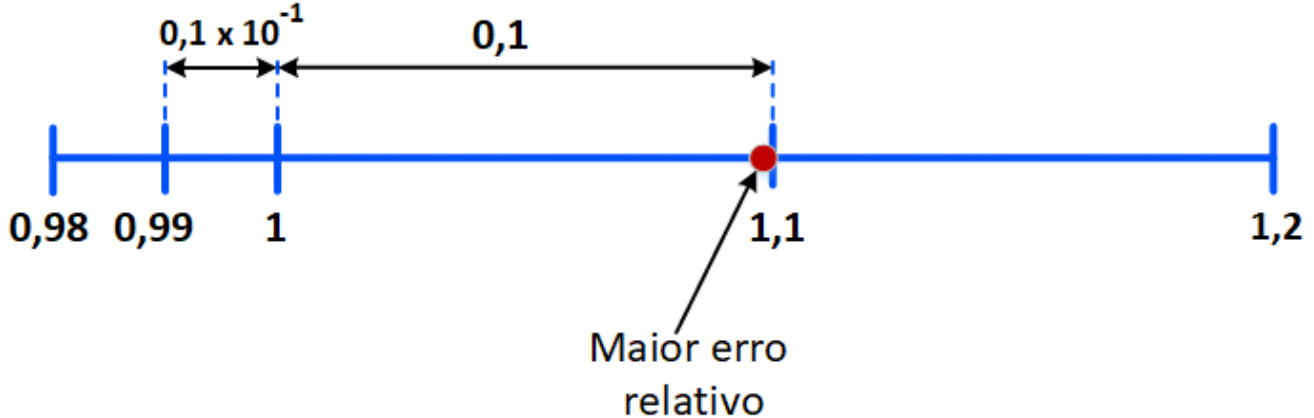
- Isso permite que a representação em ponto flutuante preserve o número de algarismos significativos.
- Em termos de erro de arredondamento, significa que dentro de uma faixa, o erro absoluto será limitado ao tamanho do intervalo.



Erros de arredondamento

Implicações da representação em ponto flutuante

- O maior erro relativo, por sua vez, ocorrerá para os números situados imediatamente abaixo do limite superior do primeiro de uma série de intervalos igualmente espaçados. Utilizando-se o corte, erro relativo será, portanto, limitado pela diferença entre 1 e o próximo número que pode ser representado.
- Este valor é chamado de *épsilon da máquina (ε)* ou precisão da máquina, que em nosso computador hipotético vale 0,1.
- Foi visto que para o número 0,01845 o erro relativo vale 0,024, um número menor que ε.



IEEE 754

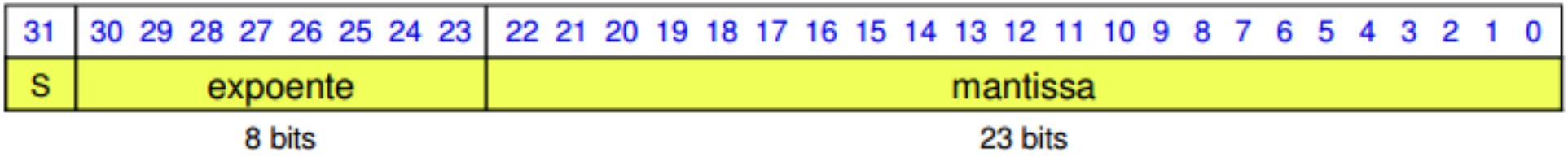
- É o padrão de base 2 utilizado atualmente para representar números em ponto flutuante em computadores :
  - Na forma normalizada, o bit à esquerda do ponto sempre será 1, ou seja, ele é implícito, não precisando ser armazenado.
  - Assim, qualquer número, diferente de zero será representado na forma normalizada por

$$(-1)^s \times (1.f) \times 2^E$$

onde s é o bit de sinal, f é a mantissa (parte fracionária do significando) e E o expoente representado em “excesso de”.

IEEE 754
Precisão Simples

- O padrão IEEE 754 de precisão simples:



- Bit de sinal (S): 1bit
  - S = 0 → número positivo
  - S = 1 → número negativo
- Expoente: 8 bits
- Significando: 24 bits (23 armazenados explicitamente)

IEEE 754
Precisão Simples

- Expoente E em excesso de 127 ($2^{(n-1)}-1$):

  E = expoente real + 127

- Para representar números, utiliza-se E entre 1 e 254, resultando em expoentes reais entre -126 e +127.

IEEE 754
Precisão Simples

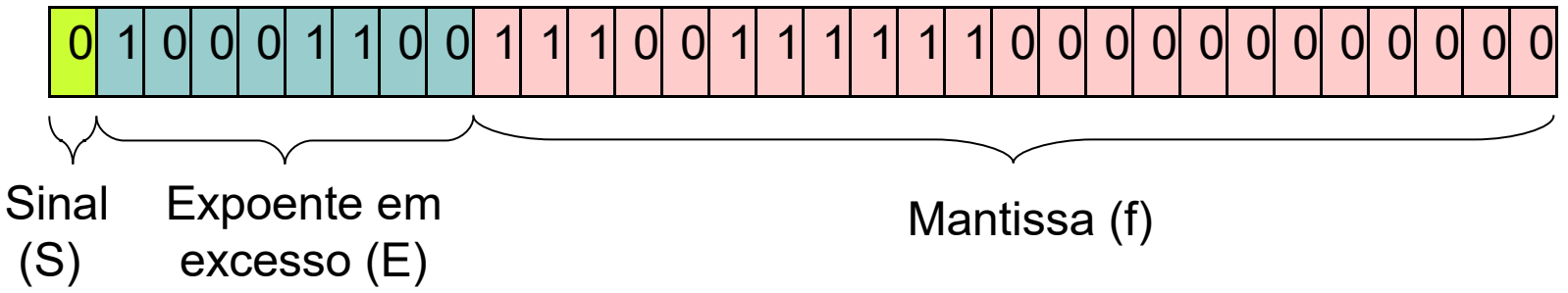
- O número decimal 15612 é representado por:

$$15612 = 11110011111100_2 = 1{,}1110011111100_2 \times 2^{13}$$

Mantissa (f) = 11100111111000000000000

Expoente (E) = $(13 + 127)_{10}$ = 10001100

Sinal = 0



IEEE 754
Precisão Simples

- Os expoentes *0* (todos os bits em *0*) e *255* (todos os bits em *1*) são reservados para denotar os valores especiais:
  - Zero: E = 0, S = 0 e f = 0
  - Números (subnormais): E = 0,f ≠ 0.
  - +∞ : E = 255, S = 0 e f = 0
  - -∞ : E = 255, S = 1 e f = 0
  - NaN (Not a Number): E = 255, S = 0 e f ≠ 0

    Resultados de operações indeterminadas (0/0, $\sqrt{-1}$, ∞ - ∞, etc.) ou exceções.

IEEE 754
Precisão Simples

- Números desnormalizados ou subnormais:
  - Possuem um bit 0 implícito à esquerda do ponto binário, permitindo a representação de números menores do que os possíveis na forma normalizada com redução da perda de significância quando ocorre um underflow.
  - Qualquer número diferente de zero será representado na forma desnormalizada por

$$(-1)^s \times (0.f) \times 2^{-126}$$

  - Perdem gradualmente a precisão à medida que diminuem (bits da esquerda da mantissa se tornam zero).

IEEE 754
Precisão Simples

- No padrão IEEE 754 de precisão simples:
  - Como a mantissa tem 23 bits, o *epsilon* ou precisão da máquina é:

$$\varepsilon = 2^{-23} = 1{,}192 \times 10^{-7}$$

  - O maior número normalizado em valor absoluto que pode ser representado é:

$$\text{Maior valor} = 1{,}111111 \ldots \times 2^{127} = (2 - \varepsilon) \times 2^{127} = 3{,}403 \times 10^{38}$$

  - O menor número normalizado em valor absoluto :

$$\text{Menor valor} = 1{,}00000 \ldots \times 2^{-126} = 1{,}175 \times 10^{-38}$$

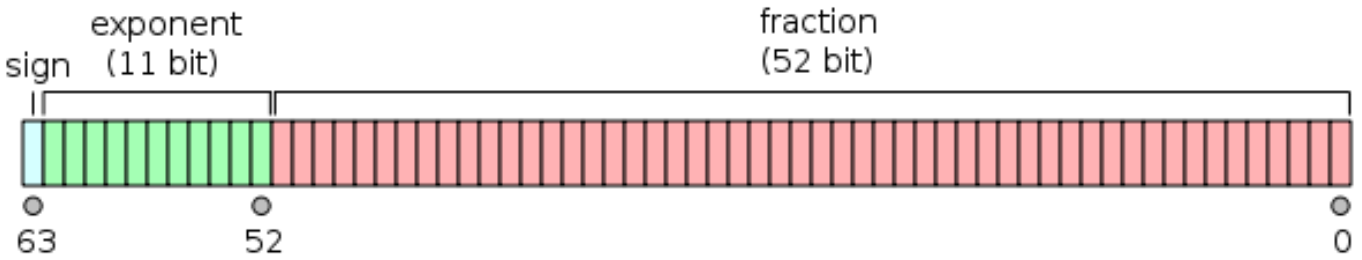
IEEE 754
Precisão Simples

- No padrão IEEE 754 de precisão simples:
  - O menor número subnormal em valor absoluto :

    Menor valor $= \varepsilon \times 2^{-126} = 2^{-23} \times 2^{-126} = 1{,}401 \times 10^{-45}$

IEEE 754
Dupla precisão

- O padrão IEEE 754 de dupla precisão:



- Utiliza 52 bits para a mantissa (M);
- Expoente E em excesso de 1023:

E = expoente real + 1023

- Para representar números, utiliza-se E entre 1 e 2046, resultando em expoentes reais entre -1022 e 1023.
- Casos especiais com E = 0 e E = 2047

IEEE 754
Dupla precisão

- No padrão IEEE 754 de precisão dupla:
  - Como a mantissa tem 52 bits, o *epsilon* ou precisão da máquina é:

$$\varepsilon = 2^{-52} = 2{,}22 \times 10^{-16}$$

  - O maior número que pode ser representado é:

$$\text{Maior valor} = 1{,}111111 \ldots \times 2^{1023} = (2 - \varepsilon) \times 2^{1023} = +1{,}80 \times 10^{308}$$

  - O menor número é:

$$\text{Menor valor} = 1{,}00000 \ldots \times 2^{-1022} = 2{,}23 \times 10^{-308}$$

IEEE 754
Dupla precisão

- No padrão IEEE 754 de precisão dupla:
  - O menor número subnormal em valor absoluto :

    Menor valor $= \varepsilon \times 2^{-1022} = 2^{-52} \times 2^{-1022} = 4{,}94 \times 10^{-324}$

IEEE 754
Algarismos significativos

- No padrão IEEE 754 de precisão simples:

  Há 23 bits denotando a mantissa mais o bit implícito, totalizando 24 bits significativos. Em dígitos decimais, tem-se aproximadamente:

$$Número\,de\,dígitos = \frac{24 \cdot \log(2)}{\log(10)} \approx 7$$

- Similarmente, no padrão IEEE 754 de dupla precisão:

$$Número\,de\,dígitos = \frac{53 \cdot \log(2)}{\log(10)} \approx 15$$

- Em muitos casos, o uso da precisão dupla pode atenuar os efeitos dos erros de arredondamento, às custas de consumo de memória e tempo de execução, consumo que pode ser insignificante para pequenos programas, mas considerável para grandes aplicações.

IEEE 754
Precisão estendida

- Conforme foi visto, 0 padrão IEEE 754 especifica dois formatos básicos de ponto flutuante: precisão simples e dupla.
- O formato de precisão simples tem um significando com precisão de 24 bits e ocupa 32 bits no total.
- O formato de dupla precisão tem um significando com precisão de 53 bits e ocupa 64 bits.
- Especifica, também, duas classes de formatos de ponto flutuante estendidos: precisão simples extendida e dupla estendida. O padrão não prescreve a precisão e o tamanho exatos desses formatos, mas especifica a precisão e o tamanho mínimos. Por exemplo, um formato IEEE precisão estendida dupla deve ter um significando com precisão de pelo menos 64 bits e ocupar pelo menos 79 bits no total.

Representações padrão IEEE 754

incluindo a representação de precisão estendida de 80 bits *x86*

	Precisão Simples	Dupla Precisão	Precisão estendida
Número de bits	32	64	80
Extensão do expoente em bits	8	11	15
Offset do expoente	127	1023	16383
Extensão da mantissa em bits	23	52	63
Precisão em bits	24	53	64
Precisão em dígitos decimais	7	16	19
Épsilon da máquina	$1{,}192 \times 10^{-7}$	$2{,}22 \times 10^{-16}$	$1{,}08 \times 10^{-19}$
Maior número em valor absoluto	$3{,}40 \times 10^{38}$	$1{,}80 \times 10^{308}$	$1{,}19 \times 10^{4932}$
Menor número normalizado	$1{,}18 \times 10^{-38}$	$2{,}23 \times 10^{-308}$	$3{,}36 \times 10^{-4932}$
Menor número subnormal	$1{,}401 \times 10^{-45}$	$4{,}94 \times 10^{-324}$	$3{,}65 \times 10^{-4951}$

Exercício

- Para computadores, o épsilon da máquina pode ser pensado como o menor número que quando adicionado a 1 resulta um número maior que 1. Um algoritmo baseado nesta ideia pode ser desenvolvido da seguinte maneira:

  Passo 1: Faça $\varepsilon = 1$;

  Passo 2: Se $(1 + \varepsilon) \leq 1$, então vá para o passo 5

  Passo 3: Faça $\varepsilon = \varepsilon / 2$

  Passo 4: Retorne ao passo 2

  Passo 5: Imprima $2 * \varepsilon$

- Escreva um script Scilab para determinar o épsilon da máquina e valide o resultado comparando-o valor calculado com o valor da constante %eps.

Aritmética de ponto flutuante
Adição/subtração

- Devido a necessidade de igualar os expoentes para alinhar o ponto decimal, a adição e a subtração são especialmente suscetíveis a perda potencial de algarismos significativos. A adição/subtração de ponto flutuante segue 4 etapas:
  - Alinhamento da mantissa se os expoentes dos números forem diferentes.
  - Adição / subtração das mantissas alinhadas.
  - Normalização do resultado se não normalizado.
  - Arredondamento do resultado.

Aritmética de ponto flutuante
Adição/subtração

- Para simplificar a discussão, suponha que se deseja adicionar dois números reais usando um computador decimal com uma mantissa normalizada de 4 dígitos.
- Para adicionar 0,2556 x 10^{1} e 0,4441 x 10^{-1}, o computador ajusta o menor número para que seu expoente coincida com o expoente do maior número:

	0	,	2	5	5	6			×	1	0	1
+	0	,	0	0	4	4	4	1	×	1	0	1
	0	,	2	6	0	0	4	1	×	1	0	1

- O resultado da adição é 0,2600 x 10^{1}.

Aritmética de ponto flutuante
Adição/subtração

- Observa-se que para ajustar o expoente do menor número, seus dois últimos dígitos foram deslocados para a direita.
- Isso é permitido porque a mantissa de um número pode ser estendida pela adição de *guard bits* para preservar a exatidão durante o arredondamento.
- Assim, os *guard bits* permitem que um número de ponto flutuante possa ser expandido durante as operações internas antes de ser armazenado em 4 bits.
- Assim, a falta de precisão (o número de dígitos ou bits mantidos ao final de um cálculo) afeta a exatidão da computação.

Aritmética de ponto flutuante
Adição/subtração

- Para subtrair 36,36 de 26,41, se deve normalizar os operandos:

	0	,	3	6	3	6	×	1	0	^{2}
-	0	,	2	6	4	1	×	1	0	^{2}
	0	,	0	9	9	5	×	1	0	^{2}

- Normalizando o resultado, obtém-se 0,9950 x 10^1. Observe que zero colocado no fim da mantissa não é significativo, ele serve apenas para preenchê-la.

Aritmética de ponto flutuante
Regra 1

- Esse resultado permite formular uma primeira regra importante da aritmética de ponto flutuante:

  Sempre que se subtrair dois números de mesmo sinal ou adicionar dois números com sinais diferentes, a precisão do resultado pode ser menor que a precisão disponibilizada pelo formato de ponto flutuante.

Aritmética de ponto flutuante
Multiplicação/divisão

- Na multiplicação, somam-se os expoentes e multiplicam-se as mantissas.
- Duas mantissas com *n* dígitos produzem um resultado com *2n* dígitos. Assim, o resultado intermediário deve ser armazenado em um registrador com o dobro de extensão. Por exemplo:

$$\begin{array}{rl} & 0{,}3641 \times 10^{3} \\ \times & 0{,}2686 \times 10^{-1} \\ \hline & 0{,}09779726 \times 10^{2} \end{array}$$

- Normalizando o resultado: $0{,}97797260 \times 10^{1}$
- Cortando: $0{,}9779 \times 10^{1}$
- O procedimento de divisão é similar, porém as mantissas são divididas e os expoentes subtraídos.

Aritmética de ponto flutuante
Regra 2

- Por si só, multiplicação e divisão não produzem resultados ruins. No entanto, eles tendem a incrementar qualquer erro que já exista em um valor.
- Por exemplo, multiplicando-se 0,453 por dois, quando se deveria estar multiplicando 0,454 por dois, o resultado é ainda menos preciso. Isso traz uma segunda regra importante quando se trabalha com aritmética de ponto flutuante:
  - *Ao realizar uma cadeia de cálculos envolvendo adição, subtração, multiplicação e divisão, deve se executar primeiro as operações de multiplicação e divisão.*

Aritmética de ponto flutuante
Regra 3

- Entretanto, a multiplicação e divisão também apresentam problemas.
- Ao multiplicar dois números muito grandes ou ao dividir um número grande por um número pequeno pode ocorrer um overflow.
- Por outro lado, a multiplicação de dois números pequenos e a divisão de um número pequeno por um número grande podem levar a um underflow. Isso nos leva a formular uma terceira regra que a ser levada em consideração:
  - *Procurar organizar os cálculos de forma que os números grandes sejam multiplicados por números pequenos e que a divisão ocorra entre números da mesma ordem de grandeza.*

Aritmética de ponto flutuante
Grandes computações

- A perda de precisão durante um único cálculo geralmente não é de grande preocupação, a menos que se esteja muito preocupado com a exatidão dos cálculos.
- No entanto, quando for calculado um valor que é o resultado de uma sequência de operações de ponto flutuante, mesmo que o erro de arredondamento de cada operação seja pequeno, o efeito cumulativo no curso de toda a computação pode ser significativo.

Aritmética de ponto flutuante
Grandes computações

- Seja o programa em linguagem C abaixo:

```
#include <stdio.h>
int main()
{
    float s = 0;
    int i;
   for (i=1;i<=10000;i++)
        s= s+0.0001;
   printf("Resultado = %f",s);
}
```

que ao ser executado, resulta em:

```
Resultado = 1.00054
```

Aritmética de ponto flutuante
Grandes computações

- Poderia ser esperado o resultado igual 1. Entretanto,embora 0,0001 seja um número redondo na base 10, ele não é redondo na base 2:

$$0{,}0001_{10} = (0.0000000000000110100011101000 \ldots)_2$$

- Assim, usando a representação IEEE 754 de precisão simples (`float` em C), as somas sucessivas levam a um resultado ligeiramente diferente de 1.

Aritmética de ponto flutuante

Adicionando um número pequeno e um grande

- Suponha que, utilizando nosso computador hipotético de 4 dígitos, adiciona-se 0,0010 a 4000:

$$\begin{array}{rl} 0{,}4000 & x\ 10^4 \\ +\ 0{,}0000001 & x\ 10^4 \\ \hline 0{,}4000001 & x\ 10^4 \end{array}$$

- Efetuando o corte, o resultado final é $0{,}4000\ x\ 10^4$, parecendo que a adição não foi realizada!
- Este tipo de erro pode ocorrer na computação de séries infinitas, onde os termos iniciais são frequentemente grandes quando comparados aos termos posteriores.

  Uma maneira de minimizar esse tipo de erro é somar os termos da série em ondem inversa, do menor para o maior.

Exemplo

- Use aritmética de 3 dígitos com truncamento para calcular a soma de

$$\sum_{i=1}^{10} \frac{1}{i^2}$$

primeiro com

$$\frac{1}{1}+\frac{1}{4}+\cdots+\frac{1}{100}$$

e depois com

$$\frac{1}{100}+\frac{1}{81}+\cdots+\frac{1}{1}$$

Qual método é mais preciso. Por quê?

- Obtenha uma aproximação melhor com o auxilio de um computador (Resposta: 1,5497677).

Exemplo

Soma dos termos da série do maior para o menor

- Erro verdadeiro:

$$\varepsilon_a = \frac{1{,}5497677 - 1{,}52}{1{,}5497677} x\, 100 = 1{,}92\,\%$$

	0,	1	0	0	x	1	0	1	1,000
+	0,	0	2	5	x	1	0	1	0,250
=	0,	1	2	5	x	1	0	1	
+	0,	0	1	1	x	1	0	1	0,111
=	0,	1	3	5	x	1	0	1	
+	0,	0	0	6	x	1	0	1	0,0625
=	0,	1	4	1	x	1	0	1	
+	0,	0	0	4	x	1	0	1	0,040
=	0,	1	4	5	x	1	0	1	
+	0,	0	0	2	x	1	0	1	0,0277
=	0,	1	4	7	x	1	0	1	
+	0,	0	0	2	x	1	0	1	0,0204
=	0,	1	4	9	x	1	0	1	
+	0,	0	0	1	x	1	0	1	0,0156
=	0,	1	5	0	x	1	0	1	
+	0,	0	0	1	x	1	0	1	0,0123
=	0,	1	5	1	x	1	0	1	
+	0,	0	0	1	x	1	0	1	0,010
=	0,	1	5	2	x	1	0	1	**1,52**

Exemplo

Soma dos termos da série do menor para o maior

- Erro verdadeiro:

$$\varepsilon_a = \frac{1{,}5497677 - 1{,}54}{1{,}5497677} x\, 100 = 0{,}63\,\%$$

	0,	1	0	0	x 1 0	-1	0,01	
+	0,	1	2	3	x 1 0	-1	0,0123	
=	0,	2	2	3	x 1 0	-1		
+	0,	1	5	6	x 1 0	-1	0,0156	
=	0,	3	7	9	x 1 0	-1		
+	0,	2	0	4	x 1 0	-1	0,0204	
=	0,	5	8	3	x 1 0	-1		
+	0,	2	7	7	x 1 0	-1	0,0277	
=	0,	8	6	0	x 1 0	-1		
+	0,	4	0	0	x 1 0	-1	0,04	
=	0,	1	2	6	x 1 0	0	$1{,}260 \times 10^{-1}$	**Ajuste**
+	0,	0	6	2	x 1 0	0	0,0625	
=	0,	1	8	8	x 1 0	0		
+	0,	1	1	1	x 1 0	0	0,111	
=	0,	2	9	9	x 1 0	0		
+	0,	2	5	0	x 1 0	0	0,25	
=	0,	0	5	4	x 1 0	1	$0{,}549 \times 10^{0}$	**Ajuste**
+	0,	1	0	0	x 1 0	1	1,000	
=	0,	1	5	4	x 1 0	1	**1,54**	

Exemplo
Conclusão – Regra 4

- Um fato muito importante que deve ser conhecido quando se trabalha com aritmética de ponto flutuante é:

  A ordem da avaliação pode afetar a exatidão do resultado.

Exercício

- A série infinita

$$f(n) = \sum_{i=1}^{n} \frac{1}{i^4}$$

converge para o valor de f(n) = $\pi^4/90$ quando n se aproxima do infinito.

- Escreva um programa em C em precisão simples para calcular f(n) para n = 10000 computando a soma de i = 1 a 10000 usando incrementos de 1
- Repita o cálculo mas na ordem inversa, isto é de i = 10000 a 1, usando incrementos de -1.
- Em cada caso estime o erro relativo percentual real. Explique os resultados.

Resposta: Crescente – f(n) = 1,082322, ε_t = 0,000103 %

Decrescente – f(n) = 1,082323, ε_t = 0,000004 %

Aritmética de ponto flutuante

Cancelamento subtrativo

- É o arredondamento induzido quando se subtrai dois números de ponto flutuante muito próximos.
- Pode ocorrer, por exemplo, quando se calcula as raízes de uma equação quadrática com a fórmula de Baskara:

$$x_{1,}\, x_2 = \frac{-b \pm \sqrt{\left(b^2 - 4\mathrm{ac}\right)}}{2\mathrm{a}} \qquad (1.9)$$

- Quando b^2 >> 4ac, as magnitudes do discriminante e de b podem ser praticamente as mesmas, isto é

$$b \cong \sqrt{\left(b^2 - 4\mathrm{ac}\right)}$$

Aritmética de ponto flutuante
Regra 5

- Nessa situação, o cancelamento subtrativo certamente levará a um erro significativo em uma das raízes. Assim, uma quinta regra a ser considerada é

  Evitar a subtração de números muito próximos.

Aritmética de ponto flutuante

Cancelamento subtrativo

- Para minimizar o problema:
  - Usar dupla precisão;
  - Multiplicar a equação (1.9) por

$$-b \mp \sqrt{(b^2 - 4\mathrm{ac})} \;\; / \;\; -b \mp \sqrt{(b^2 - 4\mathrm{ac})}$$

para obter a expressão alternativa:

$$x_{1,}\, x_2 = \frac{-2\mathrm{c}}{b \pm \sqrt{(b^2 - 4\mathrm{ac})}} \qquad (1.10)$$

Exemplo

- Calcule os valores das raízes de uma equação quadrática com a = 1, b = 3000.001 e c =3.

  Compare os valores computados em precisão simples em Linguagem C com as raízes reais x_1 = - 0.001 e x_2 = -3000:

  - Utilizando *Baskara*;
  - Utilizando a equação alternativa (1.10);
- Explique os resultados.

Solução - Código

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main()
{
    float a = 1, b = 3000.001, c = 3, x1, x2,delta;
    delta = sqrt(b*b-4*a*c);
    x1 = (-b + delta)/(2*a);
    x2 = (-b - delta)/(2*a);
    printf("As raizes pela eq.  de Baskara sao x1 =  %f e x2 = %f\n",x1,x2);
    x1 = (-2*c)/ (b+delta);
    x2 = (-2*c)/ (b-delta);
    printf("As raizes pela eq. alternativa sao x1 =  %f e x2 = %f\n",x1,x2);
    return 0;
}
```

```
As raizes pela eq.  de Baskara sao x1 =  -0.000977 e x2 = -3000.000000
As raizes pela eq. alternativa sao x1 =  -0.001000 e x2 = -3072.000000

Process returned 0 (0x0)   execution time : 0.328 s
Press any key to continue.
```

Solução - explicação

- Empregando Báskara, embora os resultados para x_2 são adequados, o percentual de erro relativo para x_1 é alto, $\varepsilon_t = 2{,}4\%$. Este nível poderia ser inadequado para muitos problemas de engenharia aplicada.
- Este resultado é particularmente surpreendente porque se está empregando uma fórmula analítica para obter a solução.
- A perda de significância ocorre onde duas quantidades muito próximas são subtraídas, o que não ocorre quando os mesmos número são adicionados.

Solução - explicação

- Com base no exposto, podemos deduzir a conclusão geral de que a fórmula de Báskara será suscetível ao cancelamento subtrativo sempre que $b^2 >> 4ac$.
- Uma forma de contornar o problema seria usar dupla precisão, minimizando o cancelamento subtrativo.
- Outra, seria reformular a fórmula quadrática no formato da Eq. 1.10 para calcular x_1, evitando o cancelamento subtrativo.

Aritmética de Ponto Flutuante
Regra 6

- Já que as operações aritméticas podem potencialmente contribuir para o erro total de arredondamento, então faz sentido tornar os procedimentos mais eficientes. Isso nos leva a formular uma sexta regra:

Minimize o número de operações aritméticas.

Regra de Horner

- Suponha que deva ser calculado o seguinte polinômio para um valor específico de n e x:

$$p(x) = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$$

- Pouco eficiente, pois o número total de multiplicações e adições realizadas será, respectivamente *n(n+1)/2* e *n*. Alternativamente, a regra de Horner pode ser utilizada para reescrever o polinômio como:

$$p(x) = a_0 + x(a_1 + x(a_2 + \ldots x(a_{n-2} + x(a_{n-1} + x\,a_n))\ldots))$$

reduzindo para *2n* o número total de operações de multiplicação e adição combinadas.

Exercícios

1. Considere a função

$$f(x) = \sqrt{(x^2 + 1)} - 1$$

- Reescreva a função racionalizando o numerador, isto é, removendo a raiz quadrada do numerador.
- Usando aritmética decimal de 5 dígitos com corte, calcule f(0,001) utilizando a função original e a função racionalizada.

  Respostas: 0 e 0,5 *x* 10^{-6}.

- Utilizando aritmética de dupla precisão (10 dígitos), calcule novamente f(0,001) utilizando as duas formas da função.
- Utilizando aritmética de dupla precisão, calcule f(1 *x* 10^{-6}) utilizando as duas formas da função.
- Explique os resultados.

Exercícios

2. Codifique a regra de Horner em linguagem C, para avaliar

$$f(x) = x^3 - 6{,}1\, x^2 + 3{,}2\, x + 1{,}5 \text{ em } x = 4{,}71$$

Exercícios

3. Conforme foi visto, a perda de exatidão devido a erros de arredondamento pode ser reduzida pelo rearranjo dos cálculos. Pede-se:

- Avalie $f(x) = x^3 - 6{,}1\,x^2 + 3{,}2\,x + 1{,}5$ em $x = 4{,}71$ usando aritmética de 3 dígitos com arredondamento e corte e calcule os respectivos erros relativos.

  `Respostas: -13,4; -13,5; 6%, 5%`

- Reescreva o polinômio empregando a regra de Horner, e avalie em usando aritmética de 3 dígitos com arredondamento e corte e calcule os respectivos erros relativos.

  `Respostas: -14,3; -14,2; 0,25%; 0;45%`

- Discuta os resultados.

Teorema da perda de precisão

- Exatamente quantos dígitos binários significativos são perdidos na subtração x - y quando x está muito próximo de y?
- A proximidade de x e y é convenientemente medida por | 1 - (y / x) |.

- Teorema

  Deixe x e y ser números de máquina de ponto flutuante normalizados, onde x > y > 0. Se $2^{-p} \leq 1 - (y / x) \leq 2^{-q}$ para alguns inteiros positivos p e q, então no máximo p e pelo menos q bits binários significativos são perdidos na subtração x − y.

Teorema da perda de precisão

- Exemplo

  Na subtração 37,593621 − 37,584216, quantos bits de significância são perdidos?

- Solução:

  Deixe x como o primeiro número e y o segundo. Então,

$$1 - y/x = 0.0002501754$$

- Isto está entre 2^{-12} e 2^{-11}, ou entre 0,000244 e 0,000488.

  Portanto, pelo menos 11 mas não mais de 12 bits são perdidos.

Bibliografia e crédito das figuras

- CHAPRA, Steven. **Applied numerical methods with MATHLAB for engineers and scientists**. 3 ed. McGraw-Hill, 012.
- SCARBOROUGH, James. **Numerical Mathematical Analysis.** London: Oxford Press, 1930.
- CHENEY, Ward. **Numerical Mathematics and Computing.** 6 Ed. Belmont, CA : Brooks/Cole, 2007.