



Departamento de Engenharia Química
Faculdade de Ciências e Tecnologia
Universidade de Coimbra



Computação Aplicada I

Introdução ao Cálculo Numérico

Pedro Nuno Simões

Ano Lectivo 2005/06

- 1 Introdução
 - Análise numérica e ciências da computação
- 2 Algoritmos numéricos e erros
 - Tipos (fontes) de erros. Erro absoluto e erro relativo
 - Condicionamento e estabilidade
- 3 Erros de arredondamento e aritmética computacional
 - Representação de vírgula flutuante
 - Erros na representação de vírgula flutuante
 - Acumulação de erros e cancelamento subtrativo
 - Representação de números em computador

Análise numérica: área da matemática e da *ciência da computação* que gera, analisa e põe em execução algoritmos capazes de resolver *numericamente* problemas de matemática contínua, cujas soluções analíticas, ou são muito difíceis, ou impossíveis.

Ciência da computação: recorre a algoritmos numéricos de modo eficiente (tirando partido do próprio "hardware") para resolver problemas matemáticos em inúmeras disciplinas das ciências e engenharias.

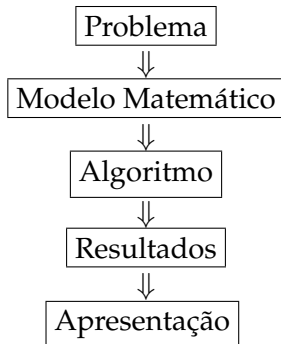


Figura: Conjunto de actividades típicas na computação científica.

Um problema é *bem formulado* se a solução para o mesmo

- existe;
- é única;
- depende continuamente dos seus dados.

Se a solução for sensível aos dados, o algoritmo deve ser de molde a não agravar essa sensibilidade.

A estratégia habitual consiste em transformar um problema difícil num equivalente, mais fácil, que conduza à mesma solução ou a uma solução aproximada.

Por exemplo:

- infinito (contínuo) \longrightarrow finito (discreto);
- diferencial \longrightarrow algébrico;
- não-linear \longrightarrow linear;
- complicado \longrightarrow simples.

Fontes de aproximação do problema:

- antes da resolução:
 - modelação;
 - medições empíricas;
 - cálculos anteriores;
- durante a resolução:
 - truncatura ou discretização;
 - arredondamento.

O grau de exactidão do resultado final é um reflexo de todos estes factores.

Incertezas nos dados do problema podem ser amplificadas pelo próprio problema.

Perturbações durante o cálculo podem ser amplificadas pelo algoritmo.

Exemplo

O cálculo da área superficial da Terra com base no modelo

$$A = 4\pi R^2$$

envolve várias aproximações:

- a Terra é modelada como uma esfera, o que é uma idealização da sua forma verdadeira;
- o valor do raio R é baseado em medições empíricas e em cálculos realizados previamente;
- o valor de π (dízima infinita) é representado de um modo truncado;
- os valores dos dados e das operações aritméticas realizadas são arredondados no computador.

Algoritmo. Conjunto de regras com vista à resolução de um determinado problema. Cada passo do algoritmo deve ser precisamente definido de modo a ser expresso numa linguagem de programação e executado em computador.

O problema contínuo é convertido num problema discreto (aproximado). Funções contínuas são aproximadas com base num conjunto finito de valores.

Os algoritmos devem ser projectados de modo a resolver o problema com eficiência, elevado nível de exactidão e de maneira fiável.

A computação científica procura projectar e desenvolver tais algoritmos. A análise numérica serve de suporte teórico a essas acções.

Após o desenvolvimento dos algoritmos há que colocá-los em prática, o que remete para questões como:

- linguagens de programação:

p. ex. C, C++, JAVA, `FORTRAN`, `MATLAB`, etc.;

- estruturas de dados;
- arquiteturas computacionais e sua exploração mediante algoritmos adequados, etc.

As fontes de erro a considerar têm diferentes origens.

- Erros associados ao problema a ser resolvido:
 - erros de aproximação associados ao modelo matemático;
 - erros nos dados.
- Erros de aproximação:
 - *erros de truncatura* devidos à *discretização* de processos contínuos (interpolação, diferenciação, integração ...);
 - *erros de convergência*, normalmente associados a processos iterativos.
- *Erros de arredondamento*:
 - resultantes da *representação finita* de qualquer número real em computador, o que afecta não só a representação de dados mas também a *aritmética computacional*.

Os erros podem expressar-se em termos absolutos ou relativos. Sejam: α = um valor exacto ou um valor de referência
 $\hat{\alpha}$ = um valor calculado

- **Erro absoluto:** $E_{\text{abs}}(\hat{\alpha}) = |\hat{\alpha} - \alpha|$

- **Erro relativo:** $E_{\text{rel}}(\hat{\alpha}) = \frac{|\hat{\alpha} - \alpha|}{|\alpha_{\text{ref}}|}$

Se $\alpha_{\text{ref}} = \alpha$: $E_{\text{rel}}(\hat{\alpha}) = \frac{|\hat{\alpha} - \alpha|}{|\alpha|}$

Exemplo

α	$\hat{\alpha}$	$E_{\text{abs}}(\hat{\alpha})$	$E_{\text{rel}}(\hat{\alpha})$
1	0.99	0.01	0.01
1	1.01	0.01	0.01
-1.5	-1.2	0.3	0.2
100	99.99	0.01	0.0001
100	99	1	0.01

Geralmente, é preferível utilizar o erro relativo!

Um importante teorema ao qual se recorre com frequência em análise numérica é o seguinte.

Teorema (Série de Taylor)

Seja $f(x)$ uma função contínua com $k + 1$ derivadas no intervalo $[x_0, x_0 + h]$. Então,

$$\begin{aligned} f(x_0 + h) = & f(x_0) + hf'(x_0) + \frac{h^2}{2!}f''(x_0) + \frac{h^3}{3!}f'''(x_0) + \cdots + \\ & + \frac{h^n}{n!}f^n(x_0) + \frac{h^{n+1}}{(n+1)!}f^{n+1}(\xi) \end{aligned}$$

em que $\xi \in [x_0, x_0 + h]$.

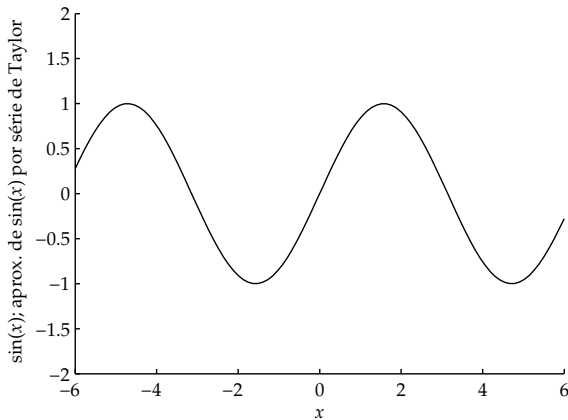


Figura: Aproximações de $f(x) = \sin(x)$ por expansão em série de Taylor, em torno de $x = x_0 = 0$.

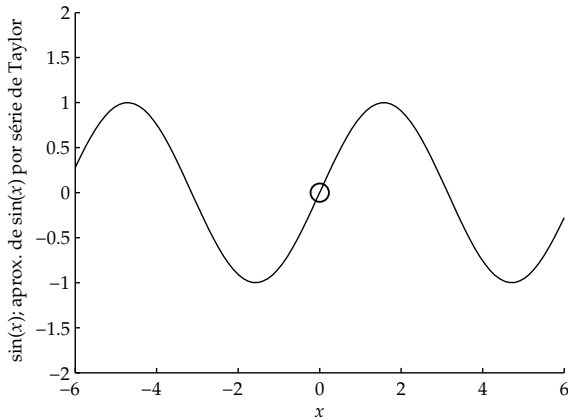


Figura: Aproximações de $f(x) = \sin(x)$ por expansão em série de Taylor, em torno de $x = x_0 = 0$.

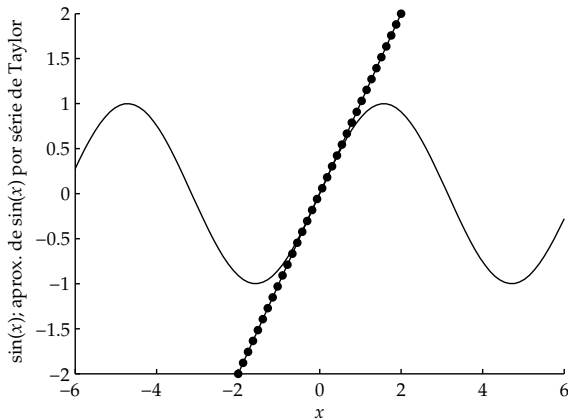


Figura: Aproximações de $f(x) = \sin(x)$ por expansão em série de Taylor, em torno de $x = x_0 = 0$.

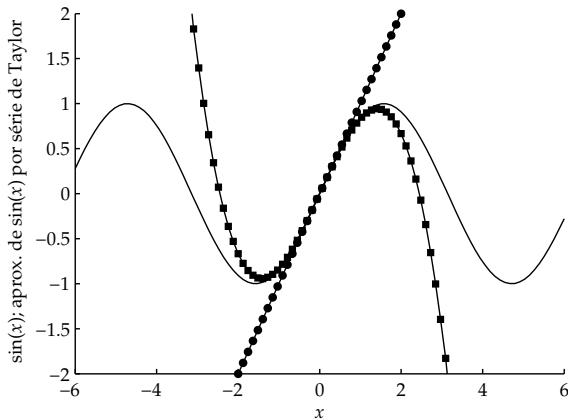


Figura: Aproximações de $f(x) = \sin(x)$ por expansão em série de Taylor, em torno de $x = x_0 = 0$.

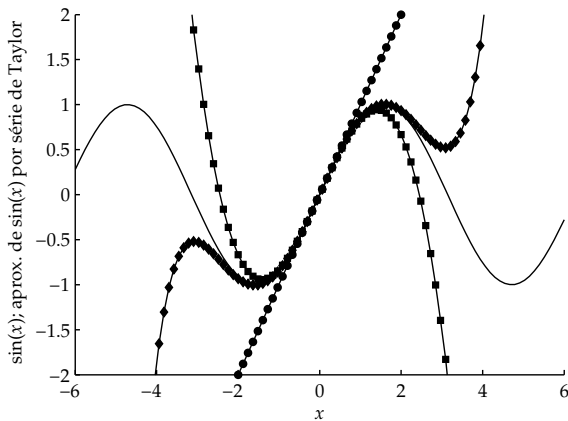


Figura: Aproximações de $f(x) = \sin(x)$ por expansão em série de Taylor, em torno de $x = x_0 = 0$.

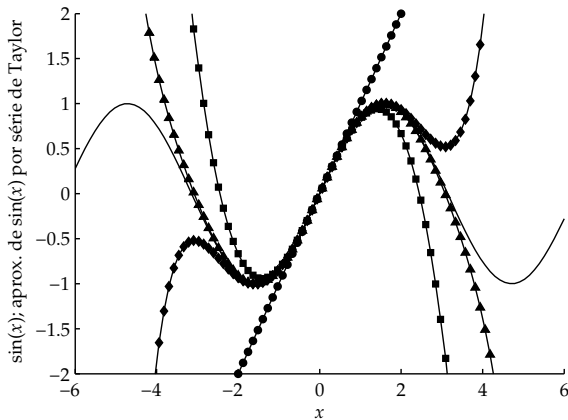


Figura: Aproximações de $f(x) = \sin(x)$ por expansão em série de Taylor, em torno de $x = x_0 = 0$.

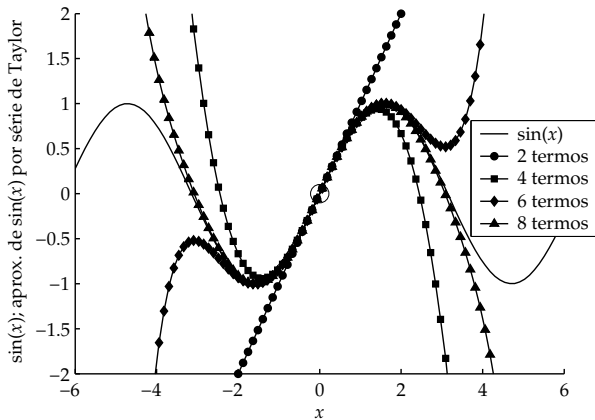


Figura: Aproximações de $f(x) = \sin(x)$ por expansão em série de Taylor, em torno de $x = x_0 = 0$.

Exemplo

Consideremos o *problema* da *aproximação* da derivada $f'(x_0)$ num ponto $x = x_0$, de uma dada função, $f(x)$.

O objectivo é desenvolver um *algoritmo* que permita o cálculo de $f'(x_0)$ por via *numérica*, e avaliar algumas questões que se prendem com os *erros*.

Seja, por exemplo, $f(x) = \sin(x)$ definida em todo o domínio dos números reais, $-\infty < x < +\infty$, e seja $x_0 = 1.2$.

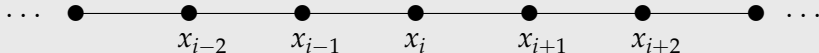
Exemplo (cont.)

Uma forma possível de construir um *algoritmo* para determinar $f'(x_0)$ numericamente consiste no recurso à série de Taylor.

Passagem do *domínio contínuo*:

$-\infty$ ————— $+\infty$

ao *domínio discreto*

\dots  \dots

Passo de discretização: $h = x_{i+1} - x_i$.

Exemplo (cont.)

Da expansão de $f(x)$ em série de Taylor em torno de x_0

$$f(x_0 + h) = f(x_0) + hf'(x_0) + \frac{h^2}{2!}f''(x_0) + \frac{h^3}{3!}f'''(x_0) + \dots$$

vem

$$f'(x_0) = \frac{f(x_0 + h) - f(x_0)}{h} - \left(\frac{h^2}{2!}f''(x_0) + \frac{h^3}{3!}f'''(x_0) + \dots \right)$$

Exemplo (cont.)

Então, um *algoritmo* possível para a *aproximação* de $f'(x_0)$ consiste no cálculo de

$$\frac{f(x_0 + h) - f(x_0)}{h}$$

A aproximação tem um *erro de discretização* ou de *truncatura* dado por

$$\left| f'(x_0) - \frac{f(x_0 + h) - f(x_0)}{h} \right| = \left| \frac{h^2}{2!} f''(x_0) + \frac{h^3}{3!} f'''(x_0) + \dots \right|$$

Exemplo (cont.)

Se $h \lll 1$ e $f''(x_0) \neq 0$:

$$\left| f'(x_0) - \frac{f(x_0 + h) - f(x_0)}{h} \right| \approx \frac{h^2}{2!} |f''(x_0)|$$

No nosso caso:

$$f(x) = \sin(x)$$

$$f'(x) = \cos(x)$$

$$f'(x_0) = \cos(1.2) = 0.362357754476674 \dots$$

Exemplo (cont.)

h	E_{abs}	h	E_{abs}
1.e-1	4.716676e-2	1.e-8	4.361050e-10
1.e-2	4.666196e-3	1.e-9	5.594726e-8
1.e-3	4.660799e-4	1.e-10	1.669696e-7
1.e-4	4.660256e-5	1.e-11	7.938531e-6
1.e-7	4.619326e-8	1.e-13	6.851746e-4
		1.e-15	8.173146e-2
		1.e-16	3.623578e-1

Exemplo (cont.)

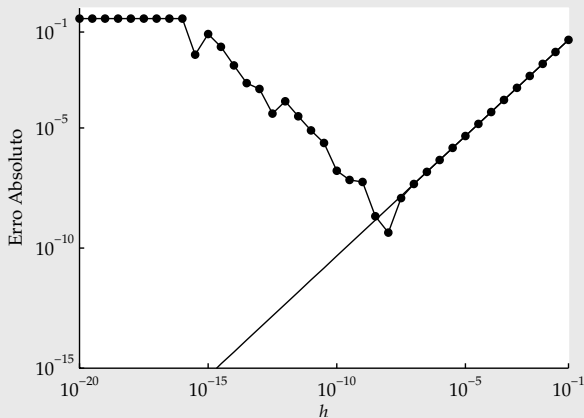


Figura: Efeito combinado dos erros de truncatura e arredondamento no cálculo numérico da derivada.

O facto de os erros serem inevitáveis em cálculo numérico levanta a questão da *sensibilidade da solução de um problema* a pequenas variações nos dados ou nos parâmetros do problema.

Um *problema* diz-se *sensível* ou *mal-condicionado* se pequenas perturbações nos seus dados ou nos seus parâmetros conduzem a grandes alterações nos resultados.

Um *problema* diz-se *insensível* ou *bem-condicionado* se pequenas perturbações nos seus dados ou nos seus parâmetros conduzem a pequenas alterações nos resultados.

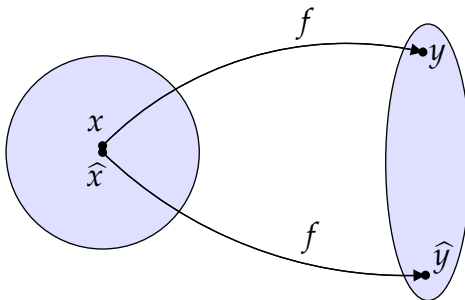


Figura: Ilustração de um problema mal-condicionado no cálculo de $y = f(x)$: quando o dado x é ligeiramente perturbado para \hat{x} , o resultado $\hat{y} = f(\hat{x})$ é muito diferente de y .

Exemplo

As raízes do polinómio

$$\begin{aligned}p(x) &= (x-1)(x-2)(x-3)(x-4)(x-5)(x-6)(x-7) \\ &= x^7 - 28x^6 + 322x^5 - 1960x^4 + 6769x^3 - 12132x^2 + 13068x - 5040\end{aligned}$$

são muito sensíveis a pequenas variações nos coeficientes. Por exemplo, se o coeficiente em x^6 for alterado para -28.002, as raízes reais originais 5 e 6 são alteradas para as raízes complexas $5.459 \pm 0.540i$.

Neste caso $p(x)$ é uma função *mal-condicionada*.

O *número de condição*, $\text{cond}[\cdot]$, é uma medida quantitativa do condicionamento de problemas numéricos.

$$\begin{aligned}\text{cond} &= \frac{|\text{erro relativo na solução}|}{|\text{erro relativo nos dados}|} \\ &= \left| \frac{\frac{f(x + \Delta x) - f(x)}{f(x)}}{\frac{\Delta x}{x}} \right|\end{aligned}$$

$$f(x + \Delta x) = f(x) + f'(x)\Delta x + \dots$$

$$f(x + \Delta x) - f(x) = f'(x)\Delta x + \dots$$

$$\frac{f(x + \Delta x) - f(x)}{f(x)} = \frac{f'(x)\Delta x}{f(x)} + \dots$$

Para Δx pequeno:

$$\frac{f(x + \Delta x) - f(x)}{f(x)} \approx \frac{f'(x)\Delta x}{f(x)}$$

Então:

$$\text{cond} \approx \left| \frac{\frac{f'(x)\Delta x}{f(x)}}{\frac{\Delta x}{x}} \right| = \left| \frac{xf'(x)}{f(x)} \right|$$

Definição

O *número de condição* de uma função $f(x)$ é dado por

$$\text{cond} [f(x)] = \left| \frac{xf'(x)}{f(x)} \right|$$

Exemplo

Função $\tan(x)$ para argumentos próximos de $\pi/2$:

$$\tan(1.57079) \approx 1.58058 \times 10^5$$

$$\tan(1.57078) \approx 6.12490 \times 10^4$$

A variação relativa na solução é $\approx 9.6 \times 10^4$ superior à variação relativa nos dados (argumento).

$$\text{cond} [\tan(x)] = \left| \frac{x(1 + \tan^2(x))}{\tan(x)} \right|$$

$$\text{cond} [\tan(1.57079)] \approx 2.48275 \times 10^5$$

O bom ou mau condicionamento de um problema é uma característica que lhe é intrínseca, portando, independente do método adoptado na sua resolução.

A *estabilidade* de um *algoritmo* é um conceito usado para indicar que o efeito dos erros computacionais não são agravados pelo próprio algoritmo.

Um *algoritmo* é *estável* se o resultado a que conduz é relativamente insensível a perturbações nos dados (considerando um problema bem-condicionado).

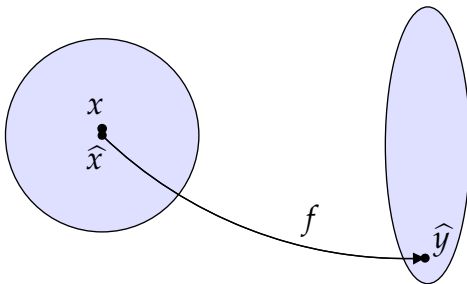


Figura: Ilustração de um algoritmo estável no cálculo de $y = f(x)$: o resultado \hat{y} é o resultado exacto para o dado ligeiramente perturbado \hat{x} , $\hat{y} = f(\hat{x})$. Assim, se o algoritmo é estável e o problema bem condicionado, o resultado calculado \hat{y} é próximo do exacto y .

Um sistema de numeração de base n é caracterizado pela utilização de n dígitos diferentes.

- **Sistema decimal:** $d_{10} = \{0, 1, 2, \dots, 9\}$.
- **Sistema binário:** $d_2 = \{0, 1\}$.
- Outros (octal, hexadecimal, ...).

O sistema binário é usado, quase invariavelmente, para representar todos os elementos de informação armazenados nos computadores.
Voltaremos a este assunto.

Consideremos, por ora, o sistema decimal.

Qualquer número real pode ser representado de modo exacto mediante uma sequência decimal *infinita*.

Exemplo

$$x = \frac{8}{3} = 2.6666\dots = \left(\frac{2}{10^1} + \frac{6}{10^2} + \frac{6}{10^3} + \frac{6}{10^4} + \frac{6}{10^5} + \dots \right) \times 10^1$$

Como veremos, a representação de números em computador é fisicamente *limitada* a um determinado número de dígitos.

Exemplo

A representação de x , por exemplo apenas com quatro dígitos por omissão dos restantes dígitos da representação infinita, será:

$$\hat{x} = \left(\frac{2}{10^1} + \frac{6}{10^2} + \frac{6}{10^3} + \frac{6}{10^4} \right) = 0.2666 \times 10^1$$

(note-se que \hat{x} é uma aproximação de x).

Generalizando a metodologia exemplificada, podemos falar em t *dígitos decimais* e chamar a t *precisão*.

Para qualquer número real x podemos associar uma *representação em vírgula flutuante*, denotada por $\text{fl}(x)$:

$$\begin{aligned}\text{fl}(x) &= \pm 0.d_1 d_2 \dots d_{t-1} d_t \times 10^e \\ &= \pm \left(\frac{d_1}{10^1} + \frac{d_2}{10^2} + \dots + \frac{d_{t-1}}{10^{t-1}} + \frac{d_t}{10^t} + \dots \right) \times 10^e\end{aligned}$$

Exemplo (cont.)

Retomando o nosso exemplo, para

$$\hat{x} = \left(\frac{2}{10^1} + \frac{6}{10^2} + \frac{6}{10^3} + \frac{6}{10^4} \right) = 0.2666 \times 10^1$$

tem-se $t = 4$ e $e = 1$.

Este exemplo serve também para mostrar que a representação não é única:

$$0.2666 \times 10^1 = 0.02666 \times 10^2 = 0.002666 \times 10^3 = \dots$$

Para evitar ambiguidades na representação, esta é *normalizada*, impondo que $d_1 \neq 0$. Assim,

$$\text{fl}(x) = \pm 0.d_1 d_2 \dots d_{t-1} d_t \times 10^e$$

em que

$$1 \leq d_1 \leq 9, \quad 0 \leq d_i \leq 9, \quad i = 2, \dots, t.$$

A gama do expoente também é restringida na representação de números em computador. Ou seja, existem inteiros $U > 0$ e $L < 0$ tais que todos os expoentes possíveis num dado sistema de representação em vírgula flutuante satisfaz

$$L \leq e \leq U$$

O maior número representável de modo preciso em tal sistema é:

$$0.99 \dots 99 \times 10^U \lesssim 10^U$$

e o número positivo mais pequeno é

$$0.10 \dots 00 \times 10^L = 10^{L-1}$$

Considere-se o número real

$$x = \pm(0.d_1d_2d_3 \dots d_t d_{t+1} d_{t+2} \dots) \times 10^e$$

Existem duas formas distintas de aproximar x tomando apenas t dígitos:

- **truncatura**: ignorar os dígitos $d_{t+1}d_{t+2}d_{t+3} \dots$;
- **arredondamento**: adicionar 1 ao dígito d_t se $d_{t+1} \geq \frac{10}{2} = 5$ e depois ignorar os dígitos $d_{t+1}d_{t+2}d_{t+3} \dots$

Exemplo

Truncatura e arredondamento de x , com $t = 3$:

x	Trunc. \hat{x}	Arred. \hat{x}
5.672	5.67	5.67
-5.672	-5.67	-5.67
5.677	5.67	5.68
-5.677	-5.67	-5.68

Recordar as expressões para os erros absoluto e relativo

- Erro absoluto: $E_{\text{abs}} = |\text{fl}(x) - x|$
- Erro relativo: $E_{\text{rel}} = \frac{|\text{fl}(x) - x|}{|x|}$

Seja $x \mapsto \text{fl}(x) = 0.f \times 10^e$ em que f é obtido como indicado atrás, por truncatura ou por arredondamento.

O erro absoluto cometido pela representação finita pode ser majorado, verificando-se que

$$E_{\text{abs}} \leq \begin{cases} 10^{-t} \cdot 10^e, & \text{no caso da truncatura} \\ \frac{1}{2} 10^{-t} \cdot 10^e, & \text{no caso do arredondamento} \end{cases}$$

Para o erro relativo, basta atender a que, devido à normalização:

$$|x| \geq \left(\frac{1}{10^1} + \frac{0}{10^2} + \dots + \frac{0}{10^t} + \dots \right) \times 10^e = 0.1 \times 10^e$$

Então, no caso de aproximação por truncatura:

$$\frac{|\text{fl}(x) - x|}{|x|} \leq \frac{10^{-t} \cdot 10^e}{0.1 \times 10^e} = 10^{1-t} =: \varepsilon_m$$

No caso de aproximação por arredondamento, o erro relativo é metade daquele que se observa no caso da aproximação por truncatura, donde:

$$\varepsilon_m = \frac{1}{2} 10^{1-t}$$

$$\varepsilon_m = \frac{1}{2}10^{1-t}$$

A quantidade ε_m é designada por *precisão da máquina*.

ε_m serve para quantificar a magnitude dos erros de arredondamento na representação de vírgula flutuante com precisão finita.

O simétrico do expoente, $t - 1$, corresponde, no caso da aproximação por arredondamento, ao número de *algarismos significativos*.

Casas decimais correctas. Uma aproximação \hat{x} de x tem k casas decimais correctas se

$$|\hat{x} - x| \leq 0.5 \times 10^{-k}.$$

Algarismos significativos correctos. Se

$$\frac{|\hat{x} - x|}{|x|} \leq 0.5 \times 10^{-k}$$

então \hat{x} é uma aproximação de x com k algarismos significativos correctos.

Assim, um algarismo significativo é correcto se o arredondamento do número aproximado depois desse dígito corresponder a um erro absoluto inferior a $1/2$ na posição daquele dígito.

Exemplo

$|22/7 - \pi| = 0.00126 \dots \leq 0.5 \times 10^{-2}$, donde $22/7 = 3.14286 \dots$ é uma aproximação de $\pi = 3.14159 \dots$ com 2 casas decimais correctas.

Também

$$\frac{|22/7 - \pi|}{|\pi|} = 4.025 \dots \times 10^{-4} \leq 0.5 \times 10^{-3}$$

donde a aproximação $22/7$ possui 3 algarismos significativos correctos.

Geralmente, é impossível eliminar a acumulação de erros de arredondamento.

Seja $E_{\text{rel},n}$ o erro relativo ao fim de uma operação n de um dado algoritmo, e sejam c_0 e c_1 constantes > 1 . Então.

$E_{\text{rel},n} = c_0 n E_{\text{rel},0}$ representa um crescimento *linear*;

$E_{\text{rel},n} = c_1^n E_{\text{rel},0}$ representa um crescimento *exponencial*;

Interessa garantir que o crescimento dos erros seja não mais que linear.

É necessário evitar um crescimento exponencial dos erros.

Potenciais problemas:

- se as magnitudes de x e y forem muito diferentes, o erro absoluto em $x + y$ é grande;
- se $|y| \gg 1$, os erros absoluto e relativo em $\frac{x}{y}$ são grandes;
- se $|y| \ll 1$, os erros absoluto e relativo em xy são grandes;
- se $x \simeq y$, o erro relativo em $x - y$ é grande (*cancelamento subtrativo*).

Exemplo

Raízes de

$$ax^2 + bx + c = 0$$

podem ser obtidas mediante:

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

Exemplo (cont.)

Seja

$$x^2 - 54.32x + 0.1 = 0$$

cujas raízes são (considerando doze casas decimais)

$$x_1 = 54.318158995042, \quad x_2 = 0.001841004958.$$

Importa notar que

$$b^2 = 2950.7 \gg 4ac = 0.4.$$

Exemplo (cont.)

Cálculo das raízes usando uma aritmética com quatro dígitos.

$$\begin{aligned}\sqrt{b^2 - 4ac} &= \sqrt{(-54.32)^2 - 0.4000} \\ &= \sqrt{2951 - 0.4000} \\ &= \sqrt{2951} \\ &= 54.32\end{aligned}$$

Exemplo (cont.)

$$x_{1,4\text{dig}} = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$$

$$= \frac{+54.32 + 54.32}{2.000}$$

$$= \frac{108.6}{2.000} = 54.30$$

$$\frac{|x_{1,4\text{dig}} - x_4|}{|x_4|} \times 100 \simeq 0.0\%$$

Exemplo (cont.)

$$x_{2,4\text{dig}} = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

$$= \frac{+54.32 - 54.32}{2.000}$$

$$= \frac{0.0000}{2.000} = 0$$

$$\frac{|x_{2,4\text{dig}} - x_4|}{|x_4|} \times 100 = 100\%$$

Exemplo (cont.)

A fraca aproximação obtida para a raiz $x_{2,4\text{dig}}$ é o resultado do erro de arredondamento no cálculo de $\sqrt{b^2 - 4ac}$, em particular devido ao cancelamento subtrativo.

É possível contornar o problema?

Exemplo (cont.)

$$\begin{aligned}x_1 &= \frac{-b + \sqrt{b^2 - 4ac}}{2a} \left(\frac{-b - \sqrt{b^2 - 4ac}}{-b - \sqrt{b^2 - 4ac}} \right) \\&= \frac{2c}{-b - \sqrt{b^2 - 4ac}}\end{aligned}$$

$$\begin{aligned}x_2 &= \frac{-b - \sqrt{b^2 - 4ac}}{2a} \left(\frac{-b + \sqrt{b^2 - 4ac}}{-b + \sqrt{b^2 - 4ac}} \right) \\&= \frac{2c}{-b + \sqrt{b^2 - 4ac}}\end{aligned}$$

Exemplo (cont.)

$$\begin{aligned}x_{2,4\text{dig}} &= \frac{2c}{-b + \sqrt{b^2 - 4ac}} \\&= \frac{0.2000}{+54.32 + 54.32} \\&= \frac{0.2000}{108.6} = 0.001842\end{aligned}$$

$$\frac{|x_{2,4\text{dig}} - x_4|}{|x_4|} \times 100 = 0.05\%$$

Exemplo (cont.)

$$\begin{aligned}x_{1,4\text{dig}} &= \frac{2c}{-b - \sqrt{b^2 - 4ac}} \\&= \frac{0.2000}{+54.32 - 54.32} \\&= \frac{0.2000}{0} = \infty\end{aligned}$$

A precisão limitada no cálculo de $\sqrt{b^2 - 4ac}$ origina um cancelamento subtrativo fatal.

Exemplo (cont.)

O recurso a uma fórmula que tenha em consideração o sinal de b será uma boa opção para prevenir o cancelamento subtrativo fatal.

$$X = -\frac{1}{2} \left(b + \text{sign}(b) \sqrt{b^2 - 4ac} \right)$$

em que

$$\text{sign}(b) = \begin{cases} 1 & \text{se } b \geq 0 \\ -1 & \text{se } b < 0 \end{cases}$$

As raízes serão então:

$$x_1 = \frac{X}{a}, \quad x_2 = \frac{c}{X}$$

Como atrás se disse, um sistema de numeração de base n é caracterizado pela utilização de n dígitos diferentes.

- **Sistema decimal:** $d_{10} = \{0, 1, 2, \dots, 9\}$.
- **Sistema binário:** $d_2 = \{0, 1\}$.
- Outros (octal, hexadecimal, ...).

O sistema binário é usado, quase invariavelmente, para representar todos os elementos de informação armazenados nos computadores.

bit: (*binary digit*) quantidade elementar de informação que pode ser representada através de um dígito binário

byte (B): agrupamento de 8 bites consecutivos;
1 byte (1 B) = 8 bits.

$$\begin{aligned}1 \text{ kilobyte (kB)} &= 1\,000 \text{ bytes} = 10^3 \text{ B} \\1 \text{ megabyte (MB)} &= 1\,000 \text{ kilobytes} = 10^6 \text{ B} \\1 \text{ gigabyte (GB)} &= 1\,000 \text{ megabytes} = 10^9 \text{ B}\end{aligned}$$

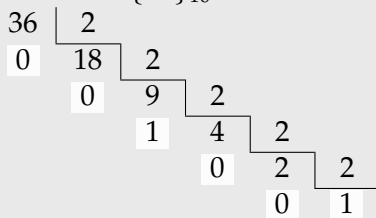
$$\begin{aligned}1 \text{ kibibyte (KiB)} &= 1\,024 \text{ bytes} = 2^{10} \text{ bytes} = 1\,024 \text{ B} \\1 \text{ mebibyte (MiB)} &= 1\,024 \text{ kibibytes} = 2^{20} \text{ bytes} = 1\,048\,576 \text{ B} \\1 \text{ gibibyte (GiB)} &= 1\,024 \text{ mebibytes} = 2^{30} \text{ bytes} = 1\,073\,741\,824 \text{ B}\end{aligned}$$

Tabela: Algumas correspondências entre as representações decimal e binária.

base 10	base 2
1	0000 0001
2	0000 0010
4	0000 0100
8	0000 1000
9	0000 1001
10	0000 1010
27	<u>0001 1011</u> um <i>byte</i>

Exemplo

Converter $\{36\}_{10}$ à base binária (e vice-versa).



$$\{36\}_{10} = \{100100\}_2$$

$$\begin{aligned} 1 \times 2^5 + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 &= \\ &= 32 + 0 + 0 + 4 + 0 + 0 = 36 \end{aligned}$$

Exemplo

Converter $\{0.3125\}_{10}$ à base binária.

$$0.3125 \times 2 = 0.6250$$

$$0.6250 \times 2 = 1.2500$$

$$0.2500 \times 2 = 0.5000$$

$$0.5000 \times 2 = 1.0000$$

$$\{0.3125\}_{10} = \{0.0101\}_2$$

Exemplo

Converter $\{0.3\}_{10}$ à base binária.

$$0.3 \times 2 = 0.6$$

$$0.6 \times 2 = 1.2$$

$$0.2 \times 2 = 0.4$$

$$0.4 \times 2 = 0.8$$

$$0.8 \times 2 = 1.6$$

\vdots

$$\{0.3\}_{10} = \{0.01001 \dots\}_2$$

- Valores numéricos com partes fraccionais não nulas são armazenados como números de vírgula flutuante.
- Um conjunto fixo de bits atribuído para armazenar cada número.
- O número total de bits divide-se em partes separadas para armazenar a mantissa e o expoente.
- Todos os valores de vírgula flutuante são representados através de notação científica normalizada:

$$\text{fl}(x) = \pm \underbrace{0.d_1d_2 \dots d_{t-1}d_t}_{\text{mantissa}} \times 10^e$$

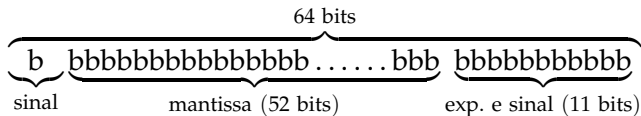
em que

$$1 \leq d_1 \leq 9, \quad 0 \leq d_i \leq 9, \quad i = 2, \dots, t.$$

- **Precisão simples** – 32 bits por número de vírgula flutuante.
- **Precisão dupla** – 64 bits por número de vírgula flutuante.

Precisão	Mantissa (bits)	Expoente (bits)
Simples	24	8
Dupla	53	11

Um número de vírgula flutuante com precisão dupla pode ser esquematicamente representado por:



Consequências:

- O limite no número de bits atribuído à representação do expoente implica a existência de um limite superior e de um limite inferior na magnitude dos números de vírgula flutuante.
- O limite no número de bits atribuído à representação da mantissa limita a precisão (número de algarismos significativos) de qualquer número de vírgula flutuante.
- A maioria dos números reais não pode ser armazenado de modo exacto.
 - Inteiros inferiores a 2^{52} podem ser armazenados de modo exacto.
 - Números com 15 dígitos decimais que seja a soma exacta de potências de $(1/2)$ podem ser armazenados de modo exacto.

Tabela: Comparação entre a recta dos números reais e a recta dos números de vírgula flutuante.

	Reais	Vírgula flutuante
Gama	Infinita: existem números reais arbitrariamente grandes e arbitrariamente pequenos.	Finita: o número de bits atribuídos ao expoente limitam a magnitude dos valores de vírgula flutuante.
Precisão	Infinita: existe um número infinito de números reais entre quaisquer dois números reais.	Finita: existe um número finito de números de vírgula flutuante entre quaisquer dois valores de vírgula flutuante.