1.5 Aritmética de Ponto Flutuante

A representação em aritmética de ponto flutuante é muito utilizada na computação digital. Um exemplo é a caso das calculadoras científicas.
Exemplo: 2,597 –03.

Este número representa: $2{,}597 \times 10^{-3}$.

A principal vantagem da representação em ponto flutuante é que ela pode representar uma grande faixa de números se comparada a representação de ponto fixo.

Seja uma representação com 6 (seis) dígitos:

a) utilizando representação de ponto fixo.

O maior número representável $= 9{,}99999 \approx 10$

O menor número representável $= 0{,}00001 = 10^{-5}$

b) utilizando representação com ponto flutuante, aloca-se dois dos seis dígitos para representar a potência de 10.

O maior número representável = $9{,}999 \times 10^{99}$.

O menor número representável = $0{,}001 \times 10^{-99}$.

A representação em ponto flutuante permite representar uma faixa muito maior de números. O preço a ser pago é que esta representação tem quatro dígitos de precisão, em oposição à representação por ponto fixo que possui 6 dígitos de precisão.

Definição:

Um sistema de ponto flutuante $F \subset \Re$ é um subconjunto dos números reais cujos elementos tem a forma:

$$y = \pm \left(\frac{d_1}{\beta^1} + \frac{d_2}{\beta^2} + \frac{d_3}{\beta^2} + \ldots + \frac{d_t}{\beta^t} \right) \beta^e = \pm (.\, d_1 d_2 d_3 \ldots d_t)\, \beta^e$$

Onde $0 \le d_i < \beta, \quad i = 1,\ldots,t$

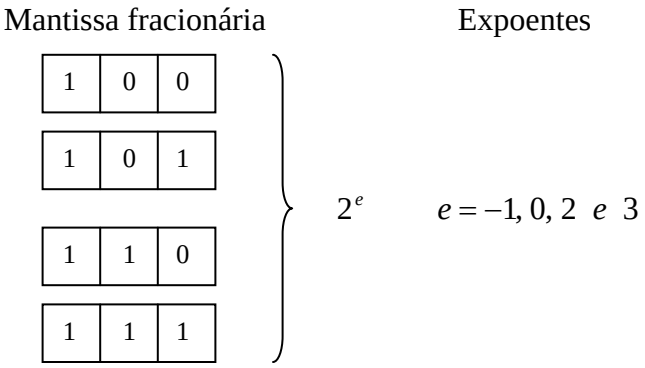
A aritmética de ponto flutuante F é caracterizada por quatro números inteiros:

- base β (binária, decimal, hexadecimal e etc..);
- precisão t (número de algarismos da mantissa);
- limites do expoente e ($e_{\min} \le e \le e_{\max}$);

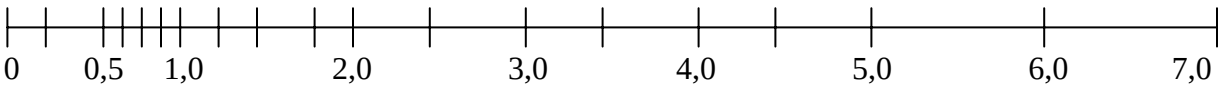
Portanto a definição de *F* é dada por $F(\beta, t, e_{\min}, e_{\max})$. A mantissa é fracionária nesta representação (<1).

A fim de assegurar representação única para cada $y \in F$, faz-se uma normalização no sistema de forma que $d_1 \neq 0$ para $y \neq 0$.

Exemplo de uma aritmética de ponto flutuante com $\beta = 2, t = 3, e_{\min} = -1 \quad e \quad e_{\max} = 3$.



Considerando apenas a parte positiva, tem-se os seguintes números: 0; 0,25; 0,3125; 0,4375; 0,5; 0,625; 0,750; 0,875; 1,0; 1,25; 1,5; 1,75; 2,0; 2,5; 3,0; 3,5; 4,0; 5,0; 6,0; 7,0, que podem ser representados na reta numerada:



Observe que os números em uma aritmética de ponto flutuante não são igualmente espaçados. Cada número na aritmética representa um intervalo de números reais.

O número total de elementos de uma aritmética de ponto flutuante é dado por:

$$numero\ de\ elementos = 2(\beta - 1)\beta^{t-1}(e_{\max} - e_{\min} + 1) + 1$$

A Tabela 1.1 mostra os parâmetros de aritméticas de ponto flutuante utilizadas em alguns computadores digitais.

Tabela 1.1 – Parâmetros de Aritméticas de Ponto Flutuante

Máquina e Aritmética	β	t	$e_{\min}$	$e_{\max}$	u
Cray-1 Precisão Simples	2	48	-8192	8191	4×10^{-15}
Cray-1 Precisão Dupla	2	96	-8192	8191	1×10^{-29}
DEC VAX formato G Dupla	2	53	-1023	1023	1×10^{-16}
DEC VAX formato D Dupla	2	56	-127	127	1×10^{-17}
Calculadoras HP 28 e 48G	10	12	-499	499	5×10^{-12}
IBM 3090 Precisão Simples	16	6	-64	63	5×10^{-07}
IBM 3090 Precisão Dupla	16	14	-64	63	1×10^{-16}

IBM 3090 Precisão Estendida	16	28	-64	63	2×10^{-33}
IEEE Precisão Simples	2	24	-126	127	6×10^{-8}
IEEE Precisão Dupla	2	53	-1022	1023	1×10^{-16}
IEEE Precisão Extendida	2	64	-16381	16384	5×10^{-20}
PDP 11	2	24	-128	127	$1{,}19\times10^{-7}$
Control Data 6600	2	48	-976	1070	$7{,}11\times10^{-15}$

1.5.1 Overflow e Underflow

O conjunto de números de números reais é infinito, entretanto, a sua representação em um sistema de ponto flutuante é limitada, pois é um sistema finito, o que não a representação exata da totalidade dos números reais.

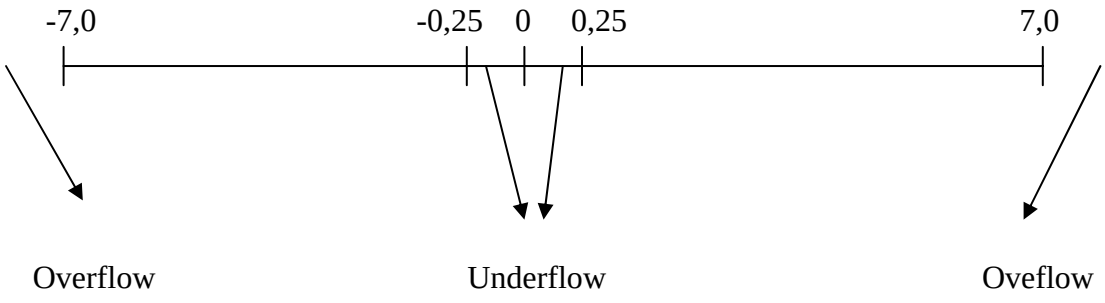
Essa limitação tem duas origens:

- a faixa dos expoentes é limitada ($e_{\min} \le e \le e_{\max}$);
- a mantissa pode representar um número finito de números ($\beta^{-1} \le m \le 1-\beta^{-t}$)

A primeira limitação leva aos fenômenos chamados de “overflow” e “underflow”. A Segunda leva aos erros de arredondamentos, que será visto na próxima seção.

Sempre que uma operação aritmética produz um número com expoente superior ao expoente máximo, tem-se o fenômeno de “overflow”. De forma similar, operações que resultem em expoente inferior ao expoente mínimo tem-se o fenômeno de “underflow”.

No caso do exemplo dado, pode-se observar qual as regiões que ocorrem o overflow e o underflow. Neste caso, considera-se a parte positiva e negativa da aritmética do exemplo.



Observe que, se o expoente for maior que 3 ou menor que -1, não tem-se representação no conjunto formado pela aritmética de ponto flutuante. No primeiro caso, tem-se o overflow, no segundo caso, tem-se o underflow.

Quando da ocorrência de overflow ou underflow, a máquina realiza alguma ação. Cada máquina responde de alguma forma. As principais ações são:

Overflow
- a) Pára o cálculo
- b) Retorna um número que representa o infinito da máquina (IEEE)

Underflow
- a) Pára o cálculo
- b) Arredonda para zero
- c) Arredonda para um número subnormal

As duas maneiras que a máquina trata o overflow possuem aspectos indesejáveis. No primeiro caso não possui resposta. No segundo caso também não é muito útil, exceto para interpretações físicas ou aplicações específicas, pois não apresenta uma resposta numérica.

No caso do underflow, o primeiro caso é indesejado. Os segundo e terceiro caso resulta em uma resposta útil, mas existe o perigo de numa operação seguinte surgir um overflow.

Deve-se procurar evitar overflow e underflow em implementação computacionais. Uma maneira prática de evitar o overflow é o escalonamento, que pode também evitar o underflow.

Exemplo:

$c = \sqrt{a^2 + b^2}$

Suponha uma máquina com 10 dígitos decimais com expoentes [-99,99]

$a = b = 10^{60}$

$a^2 \quad e \quad b^2$ ambos estão em overflow e a computação pode ser parada, mesmo que o resultado seja $\sqrt{2} \times 10^{60}$ e seja representável na aritmética de ponto flutuante.

Similarmente:

$a = b = 10^{-60}$

Se for arredondado para zero a resposta será c=0, o que é um resultado pobre considerando que a resposta é $\sqrt{2} \times 10^{-60}$.

Pode-se evitar o overflow, utilizando um escalonamento:

$s = |a| + |b|$

$$c = s\sqrt{\left(\frac{a}{s}\right)^2 + \left(\frac{b}{s}\right)^2}$$

Esta formulação também evita o underflow.

Números Subnormais

Aritméticas de ponto flutuantes mais modernas, como é o caso do IEEE, adotam os chamados números subnormais para melhorar as aproximações nos casos de underflow. Neste caso fazem parte da aritmética de ponto flutuante os números formados com a mantissa não normalizada e o mínimo expoente. No caso do exemplo dado, tem-se um acréscimo de números na aritmética.

0	1	1	$\times 2^{-1} = 0{,}1875$
0	1	0	$\times 2^{-1} = 0{,}125$
0	0	1	$\times 2^{-1} = 0{,}0625$

Aritmética IEEE – Exceções e Resultados Padrões

Tipo	**Exemplo**	**Resultado**
Operação Inválida	$0/0, \quad 0\times\infty, \quad \sqrt{-1}$	**NAN**
Overflow		$\pm\infty$
Divisão por Zero		$\pm\infty$
Underflow		Arredondamento par números subnormais
Inexatidão	$fl(x\ o\ y) \neq (x\ o\ y)$	Arredondamento do resultado

1.5.2 Erros de Arredondamentos

O comprimento de uma palavra é fixo, o que impõe que a maioria dos números não possui representação exata em um sistema de ponto flutuante.

Como exemplo, seja a raiz quadrada de 7:

$\sqrt{7} = 2{,}6457513.....$

Seja um computador com base decimal de 5 dígitos. Dígitos além do quarto decimal devem ser descartados. Tem-se dois modos de aproximação:

a) arredondamento $\Longrightarrow$ 2,6458
b) chopping (truncamento) $\Longrightarrow$ 2,6457

Como pode-se observar, em qualquer das duas aproximações, tem-se um erro na representação. É importante que se conheça os limites do erro nestas representações.

Seja o número a=X.XXXXY

Supondo que seja arredondado para o número b=X.XXXZ, com arredondamento para cima se $Y \geq 5$ e para baxo se $Y < 5$. Fica fácil observar-se que:

$$|b-a| \leq 5 \times 10^{-5}$$

Supondo que o dígito guia seja diferente de zero, ou seja, $|a| \geq 1$, resulta:

$$\frac{|b-a|}{|a|} \leq 5 \times 10^{-5} = \frac{1}{2} \times 10^{-4}$$

Supondo genericamente *t* dígitos decimais:

$$\frac{|b-a|}{|a|} \leq \frac{1}{2} \times 10^{-t+1}$$

Similarmente , quando o número a é truncado (“chopped”), tem-se:

$$\frac{|b-a|}{|a|} \leq \times 10^{-t+1}$$

Para uma base genérica β, tem-se:

a) Para arredondamento

$$\frac{|fl(x)-x|}{|x|} \leq \frac{1}{2} \times \beta^{-t+1} = u$$

b) Para chopping

$$\frac{|fl(x)-x|}{|x|} \leq \beta^{-t+1} = u$$

Estes limites podem ser apresentados de forma diferente, facilitando a análise de erros de arredondamentos.

$$\frac{(fl(x)-x)}{x} = \delta$$

Considerando a inegualdade:

$$fl(x) = x(1+\delta) \quad para \quad \delta \le u$$

1.5.3 Epsílon da Máquina (ε)

O ε da máquina é definido como o menor número de ponto flutuante, tal que $1+\varepsilon > 1$. O valor de ε é muito próximo do valor de u. O programa a seguir permite obter uma aproximação para o ε da máquina.

```
      EPS=1
10    EPS+0,5*EPS
      EPSP1=EPS+1
      IF(EPSP1.GT.1)GO TO 10
      PRINT EPS
```

Linguagens mais recentes já possue comandos para determinar-se o valor de ε

MatLab ⟶ eps
Fortran90 ⟶ EPSILON

1.6 Instabilidades Numéricas em Algoritmos

Desde o desenvolvimento dos primeiros computadores, era comum achar-se que , como os computadores desenvolvem milhões de operações de ponto flutuante para desenvolver um determinado problema, e essas operações carregam aproximações, os erros de arredondamento podem se acumular de forma desastrosa. Este sentimento parece ser verdadeiro, mas é enganoso. Na grande maioria dos casos, as instabilidades numéricas não são causadas pela acumulção de milhões de operações, mas o crescimento traiçoeiro de poucas operações.

Seja o exemplo:
$e = \exp(1) = 2{,}71828.....$
Aproxima-se o valor de e pela expressão:

$$e := \lim_{n\to\infty}(1+\frac{1}{n})^n$$

O problema será resolvido utilizando Fortran90 com precisão simples: $u = 6\times10^{-8}$, ou seja:

$$\hat{f}_n := fl\left[(1+\frac{1}{n})^n\right]$$

n	$\hat{f}_n$	$\left\|e-\hat{f}_n\right\|$
10^1	2,593743	$1{,}25\times10^{-1}$
10^2	2,704811	$1{,}35\times10^{-2}$
10^3	2,717051	$1{,}23\times10^{-3}$
10^4	2,718597	$3{,}15\times10^{-4}$
10^5	2,721962	$3{,}68\times10^{-3}$
10^6	2,595227	$1{,}23\times10^{-1}$
10^7	3,293968	$5{,}76\times10^{-1}$

A aproximação é pobre e degrada a medida que n se aprosima do inverso da unidade de arredondamento. Quando $(1+\frac{1}{n})$ é formado para n grande, poucos dígitos significativos de $\frac{1}{n}$ são retidos, e mesmo que a potencialização é realizada de forma exata, o resultado é pobre.

Nesta seção será apresentado alguns casos muito conhecidos de ocorrência de instabilidades numéricas, para erros advindos de arredondamento.

Cancelamento Catastrófico

Acontece quando dois números e que apresentam erros são subtraídos.

Seja a equação:

$$f(x)=\frac{(1-\cos x)}{x^2}$$

Considerando $x=1{,}2\times10^{-5}$ e uma máquina com 10 dígitos significativos, tem-se:

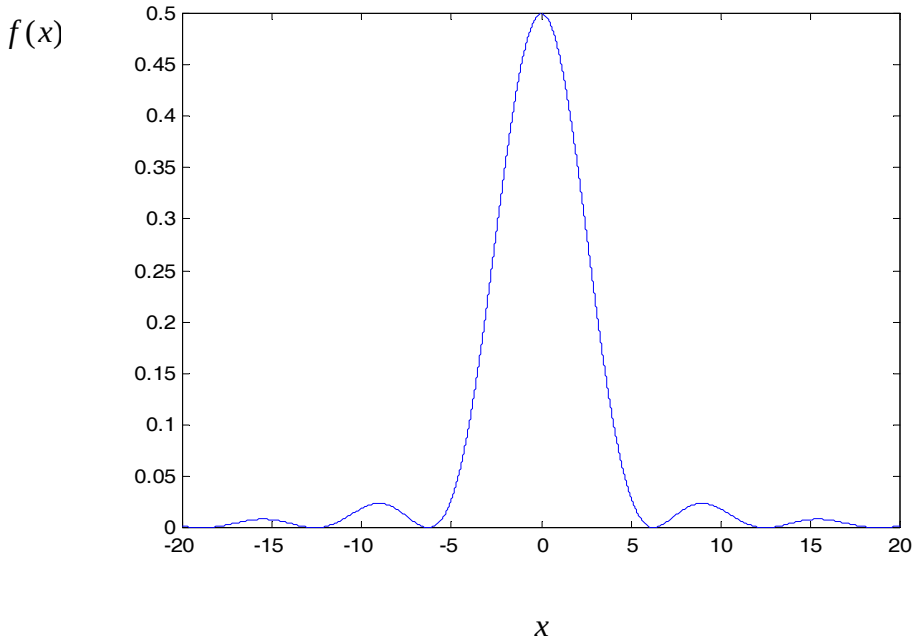
$c=\cos x=0{,}9999999999$

$1-c=0{,}0000000001$

Portanto:

$$\frac{(1-c)}{x^2} = \frac{10^{-10}}{1{,}44\times10^{-10}} = 0{,}6944$$

Entretanto, para $x \neq 0$, tem-se: $0 \leq 0 \leq 0{,}5$.



Os 10 dígitos significativos são insuficientes para aproximar o valor de $f(x)$. A subtração $(1-c)$ possui um dígito significativo. A subtração é exata, mas produz resultado da ordem do erro de *c*. A subtração supervalorizou a importância do erro anterior.

Na realidade o problema não está na subtração, mas no arredondamento anterior. Uma modificação ma equação pode tornar o cálculo numericamente estável.

$$f(x) = \frac{1}{2}\left(\frac{sin(\frac{x}{2})}{x/2}\right)^2$$

Considerando $x = 1{,}2\times10^{-5}$ e a nova expressão, chega-se a $f(x) = 0{,}5$.

|O Problema do cancelamento catastrófico pode genericamente ser mostrado por:

Seja $x = (a - b)$ e $\hat{x} = (\hat{a} - \hat{b})$, sendo $\hat{a} = a(1 + \Delta a)$ e $\hat{b} = b(1 + \Delta b)$. Os valores Δa e Δb são incertezas ou erros de arredondamentos por armazenamento ou computações anteriores. A partir dos dados, pode-se chegar a seguinte expressão:

$$\left|\frac{x - \hat{x}}{x}\right| = \left|\frac{-a\Delta a - b\Delta b}{a - b}\right| \leq \max(|\Delta a|, |\Delta b|) \frac{|a| + |b|}{|a - b|}$$

O erro relativo é grande quando:
$|a - b| << |a| + |b|$
e ocorre quando tem-se o cancelamento catastrófico. Observe por esta análise que só existe um cancelamento catastrófico quando existirem erros nos dados.

Resolvendo Equações do Segundo Grau

Seja a equação do segundo grau:
$ax^2 + bx + c = 0$
Utilizando a conhecida fórmula de Báskara:

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

Se $b^2 >> 4ac$, então $\sqrt{b^2 - 4ac} \approx |b|$. Para uma escolha de sinal, tem-se cancelamento catastrófico, pois $fl\left(\sqrt{b^2 - 4ac}\right)$ não é exato e a subtração leva a uma supervalorização do erro.

Para evitar este problema, pode-se utilizar as seguintes expressões alternativas:

$$x_1 = \frac{-b + sign(b)\sqrt{b^2 - 4ac}}{2a}$$

$$x_1 x_2 = \frac{c}{a}$$

Outra fonte de erro é quando $b^2 \approx 4ac$, neste caso nenhum rearranjo algébrico pode evitar o problema. Uma tentativa para melhorar a resposta é aumentar a precisão da solução.

Fatoração LU sem Pivoteamento

$$A = \begin{bmatrix} \varepsilon & -1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ l_{21} & 1 \end{bmatrix} \begin{bmatrix} u_{11} & u_{12} \\ 0 & u_{22} \end{bmatrix}$$

Supõe-se que $0 < \varepsilon << 1$, o que resulta: $u_{11} = \varepsilon,\ u_{12} = -1,\ l_{21} = \varepsilon^{-1},\ u_{22} = 1 - l_{21}u_{12} = 1 + \varepsilon^{-1}$

Como ε é muito pequena, $fl(u_{22}) = \varepsilon^{-1}$, portanto:

$$A - \hat{L}\hat{U} = \begin{bmatrix} \varepsilon & -1 \\ 1 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 0 \\ \varepsilon^{-1} & 1 \end{bmatrix} \begin{bmatrix} \varepsilon & -1 \\ 0 & \varepsilon^{-1} \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$$

Pode-se observar que a matriz fatorada $\hat{L}\hat{U}$ não é igual a original. Neste caso não tem-se o cancelamento catastrófico, mas operações com números com ordem de grandeza muito diferentes. A fonte de erro é apenas a utilização de um pivô muito pequeno, sendo, portanto, um problema do algoritmo e não da matriz. A matriz ***A*** é bem comportada. Para gerar um algoritmo de fatoração LU numericamente estável, deve-se evitar a utilização de pivôs com pequeno valor absoluto.

1.7 Condicionamento Numérico

Como viu-se, erros nas respostas computadas podem ser originados por problemas de instabilidades numéricas nos algoritmos utilizados, entretanto muitas vezes utiliza-se algoritmos numericamente estáveis e mesmo assim, pode-se chegar a respostas bem diferentes da esperada. A fonte desses erros está no próprio problema.

Supõe-se que $\hat{y}$ satisfaça $\hat{y} = f(x + \Delta x)$ e f seja duas vezes continuamente diferenciável. A partir da Série de Taylor, chega-se a expressão:

$$\hat{y} - y = f(x + \Delta x) - f(x) = f'(x)\Delta x + \frac{f''(x + \theta\,\Delta x)}{2}\Delta x^2 \quad onde \quad \theta \in (0,1)$$

Rearranjando a expressão:

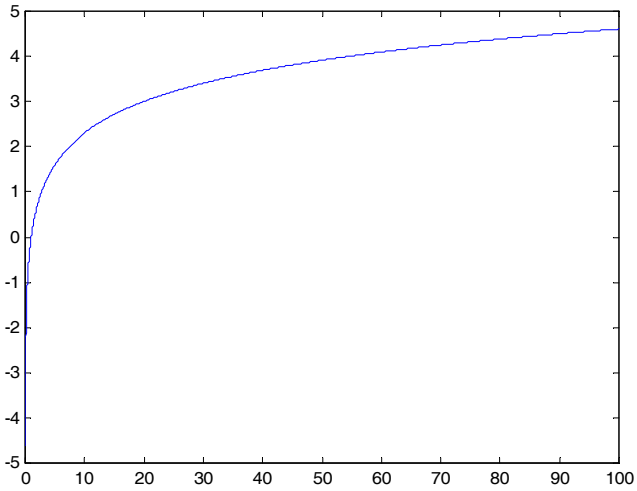
$$\frac{\hat{y} - y}{y} = \left(\frac{x f'(x)}{f(x)} \right) \frac{\Delta x}{x} + \mathrm{O}(\Delta x^2)$$

Definindo:

$$C(x) = \left(\frac{x f'(x)}{f(x)} \right)$$

Para um Δx pequeno, *C(x)* mede a perturbação relativa na saída, para uma perturbação relativa na entrada. Este valor é chamado de número de condição ou número de condicionamento. Para um problema mal condicionado, mesmo uma pequena perturbação na entrada, produz uma grande perturbação na saída.

Exemplo: $f(x) = \ln(x)$



Para a função dada:

$$C(x) = \frac{1}{\ln(x)}$$

Para um problema em que $x \approx 1$, verifica-se que $C(x)$ é muito grande. Para pequenas perturbações em x. produz grandes alterações em f(x). Significa que o problema é muito mal-condicionado para x próximo a 1. Esta conclusão também pode ser verificada no gráfico da função *f(x)* acima. A questão de condicionamento de matrizes será vista quando da solução de sistemas lineares.

1.8 Desenvolvendo Algoritmos Estáveis

Não existe receita simples para desenvolver algoritmos estáveis. Um procedimento importante é saber da necessidade da estabilidade numérica, quando do desenvolvimento de um algorítmo e não se concentrar somente em outros itens, tais como custo computacional e rapidez de solução.

Alguns itens podem ser seguidos para o desenvolvimento de algoritmos estáveis:

- Tentar evitar subtrações com quantidades contaminadas por erros.
- Minimizar o tamanho de quantidades intermediárias, relativo à solução final.
- Procurar diferentes formulações para a computação que são matematicamente, mas não numericamente equivalentes.
- É vantajoso expre4ssar atualizações do tipo: valor novo= valor velho + pequena correção, se pequenas correções podem ser computadas com muitos dígitos significativos.
- Tome precauções para evitar underflow e overflow.