Erros em computações numéricas

Sérgio Galdino

7 de Setembro de 2009

Conteúdo

Capítulo 1

Erros em computações numéricas

1.1 Revolução da computação

A opinião geral de cientistas técnicos ou usuários de computadores é que estão diante de máquinas capazes de fazer operações aritméticas rapidamente e com grande exatidão. Em parte correta quando comparada ao cálculo manual. Por exemplo, taxas típicas para multiplicação manual de números são $1/20s$ enquanto num computador são tipicamente maiores que $10^6/s$ chegando a $10^{12}/s$ em supercomputadores. Portanto, cálculos computacionais superam em várias ordens de magnitude os cálculos manuais permitindo enfrentar verdadeiros desafios computacionais. Só para fomentar o efeito psicodélico, em 1 segundo um computador de $10^6/s$ pode fazer mais cálculos (e sem cometer erros grosseiros) que uma pessoa irá fazer manualmente em sua vida inteira. Desviando-se do glamour da computação, este capítulo dará bases pra mostrar que resultados de computações numéricas devem ser vistas com cautela.

1.2 Análise numérica versus análise matemática

Matemática (especialmente no sentido convencional) e análise numérica diferem muito quando são utilizadas. A matemática normalmente assume uma representação infinita para números e processos, enquanto computações numéricas são realizadas em maquinas finitas em tempo finito. A representação finita dos números em uma máquina produz erros de arredondamento, enquanto a representação finita de um processo produz erro de truncamento:

Exemplos:

Números:

$$\pi = 3.1415926535\ldots \tag{1.1}$$

$$\frac{1}{3} = 0.33333\ldots \tag{1.2}$$

Processos:

$$\int_a^b f(x)dx = \lim_{|\Delta x_i| \to 0} \sum_{i=1}^{n} f(x_i)\Delta x_i \tag{1.3}$$

$$\frac{df(x)}{dx} = \lim_{\Delta x \to 0} \left(\frac{\Delta f(x)}{\Delta x}\right) \tag{1.4}$$

Grande parte da matemática é sensível aos efeitos produzidos pelo arredondamento numérico; esta vulnerabilidade é que os estatísticos chamam de "não é robusto".

Esta diferença é fundamental, pois somente partes da matemática resistente aos erros de arredondamentos são úteis em matemática aplicada ao mundo real.

Significados diferentes surgem para as mesmas palavras. A expressão zeros de um polinômio pode ter vários significados em computação e um único significado em matemática (ver [1] sec 1.10).

A matemática lida preferencialmente com teoremas exatos e precisos; análise numérica usa muitos métodos heurísticos. A existência de um teorema não é suficiente em cálculos computacionais. Estas diferenças são muitas vezes sérias e conduzem a grandes desentendimentos.

1.3 Análise de erros

Resultados exatos dos cálculos são um supremo ideal em análise numérica. Quatro tipos de erro afetam a exatidão dos cálculos [2]: erros de modelo, erros de dados, erros de algoritmos e erros de arredondamento. A maioria da literatura em língua inglesa faz classificação diferente. Estes erros não são conseqüências de equívocos ou decisões precipitadas. Diferente, por exemplo, de erros de programação, eles são inevitáveis. Em muitos casos eles podem ser antecipados, e requerimentos de exatidão podem ser impostos, i.e., eles podem ser controlados para permanecerem abaixo de certos limites de erros. Os limites de erro são parte da especificação do problema numérico:

|Erros do modelo + erros dos dados + erros de algoritmos + erro de arredondamento| $\leq$ tolerância

Todos os erros relevantes têm que ser identificados e seus efeitos nos resultados numéricos devem ser avaliados. Nas seções seguintes os quatro tipos de erros são caracterizados.

1.3.1 Erros de modelo

Em qualquer processo de modelagem, várias grandezas são desprezadas. O modelo resultante é uma abstração da realidade e vários modelos podem ser utilizados. O desvio inevitável entre o modelo e o objeto modelado é denotado por erro de modelagem. É necessário estimar a magnitude dos efeitos dos erros de modelagem para garantir os requisitos de tolerância de erro. Normalmente tais estimativas não são obtidas pois os fatores envolvidos são desconhecidos e não quantificados.

1.3.2 Erros de dados

Geralmente modelos não são para uma aplicação específica, mas para uma classe de aplicações similares. Uma instância é identificada por valores de parâmetros do modelo. Por exemplo, o comprimento l, o deslocamento angular inicial θ e a constante gravitacional g (que depende da localização geográfica) são parâmetros do modelo matemático do pêndulo simples. Devido a medições inexatas e outros fatores, os valores usados para parâmetros do modelo diferem do verdadeiro valor (normalmente desconhecido); isto é chamado de erro de dados. Os impactos dos erros de dados são objetos de análise.

1.3.3 Erros de algoritmo

Quando um problema matemático não pode ser resolvido analiticamente usando manipulações algébricas, então pode ser tentada uma solução por algoritmo numérico. No desenvolvimento de algoritmos numéricos são feitas simplificações antes que uma formulação finita do problema possa ser obtida para que o esforço computacional requerido seja reduzido a um nível razoável. O desvio resultante dos resultados obtidos pelo algoritmo dos da solução do problema matemático é denotado por erro de algoritmo.

Exemplo:

A solução do sistema de equações

$$Ax = b, \qquad A \in \mathbb{R}^{nxn}, \qquad b, x \in \mathbb{R}^n$$

requer um esforço computacional proporcional a n^3. Se uma solução aproximada $\widetilde{x}$ satisfazendo

$$||A\widetilde{x} - b|| < \varepsilon$$

é suficiente, o custo computacional pode ser reduzido significativamente. Se A não possui estrutura especial, então somente

$$k \approx \sqrt{\kappa_2}\frac{ln(2/\varepsilon)}{2}, \qquad \kappa_2 := \|A\|_2 \left\|A^{-1}\right\|_2$$

multiplicações matriz-vetor são necessárias para solução iterativa [4]. O número κ_2 é o número condição euclidiano da matriz A (ver seção 13.8 [2]).

1.3.4 Erro de truncamento

Algoritmos numéricos implantados em um computador podem somente realizar uma seqüência finita de operações aritméticas (adição, subtração, multiplicação, divisão e lógicas). Para calcular funções predefinidas *sin*, *exp*, *ln*, ... somente uma seqüência finita de operações aritméticas são executados pelo computador. O erro devido a troca de um processo infinito por uma seqüência finita de operações aritméticas é chamado de erro de truncamento.

Exemplo:

A troca da série infinita da exponencial

$$e^x = \sum_{k=0}^{\infty} \frac{x^k}{k!} \quad com \quad P_n(x) = \sum_{k=0}^{n} \frac{x^k}{k!}$$

produz um erro de truncamento

$$e_{trunc}(x) := P_n(x) - cos(x)$$

que cresce com a distância entre x e zero.

1.3.5 Erros de discretizaração

O erro resultante de uma troca de informação contínua por informação discreta, num processo de amostragem, é referido como erro de discretização. Muitos autores estendem o termo erro de truncamento para incluir o erro de discretização.

Exemplo

O cálculo de uma integral definida

$$I = \int_a^b f(x)dx$$

são aproximados por somas finitas envolvendo uma malha de pontos $x_i \in [a,b]$, $i = 1,2,\cdots n$ que pertencem ao conjunto de números de pontos flutuantes e as correspondentes avaliações aproximadas das funções $\{f(x_1),\cdots f(x_n)\}$. Os erros de arredondamento comprometem a exatidão dos resultados quando tenta-se minimizar os erros de truncamento refinando-se a malha de discretização.

1.3.6 Erros de arredondamento

Um computador fornece somente um conjunto finito de números: inteiros, e número de ponto-flutuante com mantissa de comprimento fixo. Desta forma as operações feitas num programa de computador não são geralmente executadas exatamente. Cada passo mapeia seu resultado em um dos números de ponto-flutuante disponíveis, normalmente o mias próximo. A diferença entre o resultado exato e o resultado arredondado de uma operação á chamado de erro de arredondamento. O efeito dos erros de arredondamento acumulados sobre o resultado final do método de aproximação é chamado de efeito de erro de arredondamento.

1.3.7 Estudo de caso

É interessante ilustrar diferentes tipos de erro mais explicitamente. Suponha que nós queremos computar $f(x)$ para um dado x real. No cálculo prático x é aproximado por $\widehat{x}$ pois o computador tem uma palavra finita. A diferença entre $|\widehat{x}-x|$ é o *erro inicial*, enquanto $\varepsilon_1 = f(\widehat{x}) - f(x)$ é o *erro propagado* correspondente. Normalmente f é trocado por uma função mais simples f_1 (freqüentemente uma série de potência truncada). A diferença $\varepsilon_2 = f_1(\widehat{x}) - f(\widehat{x})$ é o *erro de truncamento*. Os cálculos são feitos por um computador, portanto não são exatos. Na realidade calculamos $f_2(\widehat{x})$ no lugar de $f_1(\widehat{x})$, o qual é um valor calculado errado de uma função errada com argumento errado. A diferença $\varepsilon_3 = f_2(\widehat{x}) - f_1(\widehat{x})$ é o *erro de arredondamento propagado*. O erro total é $\varepsilon = f_2(\widehat{x}) - f(x) = \varepsilon_1 + \varepsilon_2 + \varepsilon_3$.

Exemplo: Calcular $e^{1/3}$ fazendo todos os cálculos com 4 decimais. (**Obs.** Os cálculos foram feitas com uma calculadora)

$$\varepsilon_1 = e^{0.3333} - e^{1/3} = -0.000465196$$

$$e^x \approx f_1(x) = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!}$$

para $\widehat{x} = 0.3333$

$$\varepsilon_2 = f_1(\widehat{x}) - f(\widehat{x}) = -\left(\frac{0.3333^5}{5!} + \frac{0.3333^6}{6!} + \cdots\right) = -0.0000362750$$

$$f_2(\widehat{x}) = 1 + 0.3333 + 0.0555 + 0.0062 + 0.0005 = 1.3955$$

$$f_1(\widehat{x}) = 1.3955296304 \textit{ obtidos com } 10 \textit{ decimais}$$

$$\varepsilon_3 = -0.0000296304$$

$$\varepsilon = 1.3955 - e^{1/3} = \varepsilon_3 + \varepsilon_3 + \varepsilon_3 = -0.0001124250$$

1.4 Propagação de erros

Os erros são definidos como absoluto e relativo. Se x é o número exato e $\widehat{x}$ uma aproximação, então temos:

$$\textit{Erro absoluto}: \quad \varepsilon = |x - \widehat{x}|, \quad e$$

$$\textit{Erro relativo}: \quad \left|\frac{\varepsilon}{x}\right| = \left|x - \frac{\widehat{x}}{x}\right|,$$

Um número decimal é arredondado na posição *n* desprezando-se todos os dígitos à direita desta posição. O dígito na posição *n* é deixado inalterado ou acrescido de uma unidade se o dígito da posição $n+1$ é um número menor que 5 ou maior que 5. Se o número na posição $n+1$ for igual a 5, o digito na posição *n* é acrescido de uma unidade se ele for par e é deixado inalterado se for impar (a regra pode ser o contrário: o digito na posição *n* é acrescido de uma unidade se ele for impar e é deixado inalterado se for par) . Freqüentemente é feito o truncamento para *n* decimais onde todos os dígitos além da posição n são simplesmente desprezados.

Exemplo:

3.1415926535

Arredondamento:

3.14 (2d)

3.141 (3d)

3.1416 (4d)

3.1415927 (7d)

onde (nd) = número de casas decimais.

Nós podemos dizer de forma simplória que dígitos significativos são aqueles que têm informação sobre a dimensão do número sem contar com a parte exponencial. Naturalmente um dígito *d* localizado mais à esquerda tem mais informação do que um mais à direita. Quando um número é escrito com somente seus dígitos significativos estes formam uma cadeia de símbolos que começa com o primeiro dígito diferente de zero. Portanto se a parte fracionária termina com um ou vários zeros, eles são significativos por definição. Se o número é inteiro e termina com zeros, somente com o conhecimento da situação é que podemos decidir se eles são significativos ou não. Por exemplo, 8630574 escrito com 4 dígitos significativos é 8630000.

Em muitos casos nós estimamos o erro de uma função $f(x_1, x_2, \ldots, x_n)$ com erros individuais nas variáveis $(x_1, x_2, ..., x_n)$ conhecidos. Nós encontramos diretamente que

$$\Delta f = \frac{\partial f}{\partial x_1}\Delta x_1 + \frac{\partial f}{\partial x_2}\Delta x_2 + \cdots + \frac{\partial f}{\partial x_n}\Delta x_n$$

onde os termos de ordem superior foram desprezados. O erro máximo é dado por

$$|\Delta f| = \left|\frac{\partial f}{\partial x_1}\right||\Delta x_1| + \left|\frac{\partial f}{\partial x_2}\right||\Delta x_2| + \cdots + \left|\frac{\partial f}{\partial x_n}\right||\Delta x_n|$$

O limite superior do erro é geralmente bastante pessimista, em computações práticas os erros têm uma tendência a cancelar. Por exemplo, se 20000 números são arredondados com quatro casa decimais e adicionados, o erro máximo é

$$\frac{1}{2} \times 10^{-4} \times 20000 = 1$$

1.4.1 Aritmética intervalar

O cálculo de erro de arredondamento estimado por aproximação de primeira ordem, descrito no final da indrodução da seção 1.4 acima, são inviáveis para serem utilizados em métodos numéricos típicos onde o número de operações aritméticas é muito grande para permitir uma estimativa satisfatória do efeito total de todos os erros de arredondamentos.

A aritmética intervalar é uma alternativa para determinar limites para o erro absoluto de um algoritmo, considerando todos erros de dados e arredondamento. A aritmética intervalar faz cálculos sistemáticos através de intervalos $[x] = [\underline{x},\ \overline{x}]$ limitados de números de máquinas $\underline{x},\ \overline{x} \in \mathbb{F}$, em vez de números reais simples x. As operações aritméticas $+,\ -,\ \times,\ \div$ são definidas através de intervalos. Algoritmos intervalares são implementados em computadores produzindo resultados intervalares garantidos conterem a solução desejada.

A aritmética intervalar deve ser usada com bastante senso crítico, caso contrário os resultados confiáveis de limites de erro serão, na maioria das vezes, muito pessimísticos.

Exemplo:

Calcule $y = x^3 - 3x^2 + 3x$ para $[x] = [0.9, 1.1]$?

Pelo esquema de Horner

$y = ((x-3)x+3)x$

$[y] = [0.6209, 1.4191]$ $(muito\ largo)$

usando

$y = 1 + (x-3)^3$

$[y] = [0.9989, 1.0011]$ $(resultado\ ótimo)$

Para o sucesso da aplicação da aritmética intervalar é necessário desenvolver novos algoritmos que produzam limites de erros aceitáveis. Um tratamento mais profundo da aritmética intervalar pode ser visto em Moore [5], Alefeld [6] e Kulish [7] (para iclusão dos arredontadmenos direcionados da aritmética de ponto flutuante).

1.4.2 Estimativa estatística de erros de arredondamento

Para obtenção de estimativas estatísticas de erros de arredondamento relativo (ver Rademacher [8]) que é causado por uma operação elementar, seus resultados são considerados uma variável aleatória no intervalo $[-eps, eps]$ onde $|\varepsilon| \leq eps$. Alem disso assume-se que os erros de arredondamento atribuídos as operações diferentes são variáveis independentes. Por μ_ε e σ_ε^2 denota-se o valor esperado e variância da distribuição do arredondamento. Eles satisfazem as seguintes relações gerais

$$\sigma_\varepsilon^2 = E(\varepsilon - E(\varepsilon))^2 = E(\varepsilon^2) - ((E(\varepsilon))^2 = \mu_{\varepsilon^2} - \mu_\varepsilon^2$$

Assumindo uma distribuição uniforme no intervalo $[-eps, /eps]$, obtém-se

$$\mu_\varepsilon := E(\varepsilon) = 0, \quad \sigma_\varepsilon^2 = E(\varepsilon^2) = \frac{1}{2\,eps}\int_{-eps}^{eps} t^2 dt = \frac{1}{3}eps^2 =: \varepsilon^{-2} \tag{1.5}$$

Exames rigorosos mostram que a distribuição de arredondamento não é muito uniforme (ver Sterbenz [9]). Deve-se ter em mente que o padrão ideal do erro de arredondamento é somente uma aproximação do padrão observado em cálculos computacionais, assim os as estimativas de μ_ε e σ_ε^2 devem ser determinadas empiricamente. Os resultados x dos algoritmos estando sujeitos a erros de arredondamento aleatórios tornam-se variaríeis aleatórias com valor esperado μ_x e variância σ_x^2 satisfazendo a mesma relação básica

$$\sigma_x^2 = E(x - E(x))^2 = E(x^2) - ((E(x))^2 = \mu_{x^2} - \mu_x^2$$

Os efeitos da propagação de erros de arredondamento das operações elementares são descritas pelas seguintes fórmulas para variáveis aleatórias independentes x, y e constantes $\alpha, \beta \in \mathbb{R}$:

$$\begin{aligned} \mu_{\alpha x \pm \beta y} &= E(\alpha x \pm \beta y) = \alpha E(x) \pm \beta E(y) = \alpha \mu_x \pm \beta \mu_y \\ \sigma^2_{\alpha x \pm \beta y} &= E((\alpha x \pm \beta y)^2) - (E(\alpha x \pm \beta y))^2 \\ &= \alpha^2 E(x - E(x))^2 + \beta^2 E(y - E(y))^2 = \alpha^2 \sigma_x^2 + \beta^2 \sigma_y^2 \end{aligned} \tag{1.6}$$

A primeira fórmula acima resulta da lineridade do valor esperado. Ela é valida para variáveis aleatórias x, y. A segunda fórmula segue da relação $E(xy) = E(x) \cdot E(y)$, que é satisfeita se x e y são independentes. Assim

$$\begin{aligned} \mu_{x \times y} &= E(x \times y) = E(x)E(y) = \mu_x \mu_y \\ \sigma^2_{x \times y} &= E(x \times y - E(x)E(y))^2 = \mu_{x^2}\mu_{y^2} - \mu_x^2 \mu_y^2 \\ &= \sigma_x^2 \sigma_y^2 + \mu_x^2 \sigma_y^2 + \mu_y^2 \sigma_x^2 \end{aligned} \tag{1.7}$$

Alem disso os valores esperados μ_x são trocados por valores estimados x e variâncias relativas $\varepsilon_x^2 = \sigma_x^2/\mu_x^2 \approx \sigma_x^2/x^2$ são considerados, então de (1.6) e (1.7):

$$\begin{aligned} z = x + y \text{ ou } z = x - y &: \quad \varepsilon_z^2 \doteq (\tfrac{x}{z})^2 \varepsilon_x^2 + (\tfrac{y}{z})^2 \varepsilon_y^2 + \overline{\varepsilon}^2 \\ z = x \times y \text{ ou } z = x \div y &: \quad \varepsilon_z^2 \doteq \varepsilon_x^2 + \varepsilon_y^2 + \overline{\varepsilon}^2 \end{aligned}$$

Os limites de erros para o resultado final r de uma computação numérica são obtidos da variância relativa ε_r^2, assumindo que o erro final tem distribuição normal. Esta suposição é justificada uma vez as distribuições dos erros propagados tendem a ser normal se sujeitas as muitas operações elementares. Supondo o resultado final ser normal, o erro relativo do resultado final r é limitado com probabilidade 0.9 por $2\varepsilon_r$

1.5 Padronização do sistema de números de ponto flutuante

Padrões internacionais são desenvolvidos pelo ISO (International Standardization Organization) com suas organizações nacionais afiliadas de padronização (ABNT - Associação Brasileira de Normas Técnicas, Av. 13 de Maio, n° 13, 28° andar CEP 20003-900 - Rio de Janeiro-RJ- Brazil, E-mail: abnt@abnt.org.br, Web: http://www.abnt.org.br/ , Tel +55 11 30 17 36 00, Fax +55 11 30 17 36 33) . Um caso especial é o campo da engenharia elétrica e eletrônica, na qual os padrões são desenvolvidos pela IEC (International Electrotecnical Commission).

Nos anos 70 tentou-se desenvolver padrões para a aritmética de ponto flutuante binária para microcomputadores. Um dos principais objetivos era tornar programas mias portáveis para que técnicas de programação particulares usadas para tratar erros de arredondamento exceções (e.g. , overflow de expoentes) fossem efetivas pra arquiteturas de computadores diferentes .

Após lenta negociação, a sociedade americana de computadores IEEE (Institute of Electrical and Eletronics Engineers) adota o padrão IEE 754-1985, o padrão para a aritmética de ponto flutuante binária (abreviadamente IEEE-754). Em 1984 o IEC decide considerar este padrão nacional um padrão internacional , IEC 559:1989 da aritmética de ponto flutuante para sistemas de microcomputadores. O padrão relacionado IEEE 854-1987 generaliza 754 para cobrir tanto decimal como binário.

1.5.1 Parâmetros de um sistema de número de ponto-flutuante

Quatro parâmetros e um valor booleano:

1. Base: $b \geq 2$
2. Precisão: $p \geq 2$
3. menor expoente: $e_{min} < 0$
4. maior expoente: $e_{max} < 0$
5. indicador de normalização: $dnorm \in booleano$

caracterizam cada sistema de números IEEE/IEC

dnorm	= **true** caso o sitema contenha números denormalizados (sub-normal)
	= **false** para o sistema normalizado

A notação de um sistema de ponto flutuante é

$$\mathbb{F}(b, p, e_{min}, e_{max}, dnorm)$$

que satisfaz as seguintes relações

$$\mathbb{F}(b,p,e_{min},e_{max},true) = \mathbb{F}_N(b,p,e_{min},e_{max}) \cup \mathbb{F}_D(b,p,e_{min},e_{max})$$

$$\mathbb{F}(b,p,e_{min},e_{max},false) = \mathbb{F}_N(b,p,e_{min},e_{max}).$$

Exemplo (Intel) - De acordo com a norma IEC/IEEE os sistemas de numeração utilizados nos microprocessadores Intel são $\mathbb{F}(2,24,-125,128,true)$ e $\mathbb{F}(2,53,-1021,1024,true)$ para precisão simples e precisão dupla. Microprocessadores Intel têm precisão estendida $\mathbb{F}(2,64,-16381,16384,true)$.

Exemplo (IBM System/390) - Fornece três sistemas de numeração hexadecimal $\mathbb{F}(16,6,-64,63,false)$ para *short precision*, $\mathbb{F}(16,14,-64,63,false)$ *para long precision* e $\mathbb{F}(16,28,-64,63,false)$ para *extended precison.*

Exemplo (Cray) - Fornece dois sistemas de numeração $\mathbb{F}(2,48,-16384,8191,false)$ e $\mathbb{F}(2,96,-16384,8191,false)$.

Exemplo (Calculadoras) Calculadoras cientificas são normalmente fornecidas com um único sistema de numeração $\mathbb{F}(10,10,-98,100,false)$. Algumas trabalham com maior precisão interna enquanto exibem somente dez casas decimais.

1.6 Programas de precição múltipla

Em certas aplicações o sistema de ponto flutuante fornecido pelo computador tem precisão ou intervalo insuficiente. Se os cálculos são feitos em precisão simples, então se usa precisão dupla para tentar remediar dificuldades e, se ainda assim, a precisão dupla é insuficiente, tenta-se a precisão estendida. Contudo esta estratégia prejudica a portabilidade dos programas, pois poucos sistemas de computadores fornecem mais que dois níveis de precisão. Além disso, esta estratégia é de uso limitado: existem raramente quatro níveis, então o nível de precisão pode não ser adaptado aos requisitos de uma aplicação particular. Uma solução é o uso de um software de precisão múltipla (Maple, Matlab, etc.) , que permite aumentar a precisão independente da arquitetura.

Estes softwares permitem uma escolha flexível da precisão e/ou intervalo necessário dos números de ponto flutuante. A desvantagem mais séria das implementações de sistemas de numeração via software comparada a hardware é o tempo de processamento (crescimento em tempo de execução por um fator > 100 [10]

1.7 A influência da aritmética do ponto flutuante

Os problemas com o uso da aritmética de ponto flutuante são fartamente discutido na literatura, o artigo [11] é clássico. A seguir veremos alguns casos selecionados.

Para resolver um sistema de equações na linear, a função $f : \mathbb{R}^n \to \mathbb{R}^n$ deve ser implementada na forma de um (sub)programa. Então os zeros da implementação $\tilde{f} : \mathbb{F}^n \to \mathbb{F}^n$ são determinados, em vez da função matematicamente definida f.

O termo *zero* perde seu significado comum mesmo no caso de funções de uma variável $(n = 1)$ porque a equação

$$\tilde{f}(x) = 0 \quad com\ \tilde{f} : \mathbb{F} \to \mathbb{F}$$

pode ter várias soluções (zeros) ou nenhuma solução na vizinhança de um zero isolado da função original.

Exemplo (sem zeros [3], pag. 279)

A função

$$f(x) = 3x^2 + \frac{1}{\pi^4} ln(\pi - x)^2 + 1 \tag{1.8}$$

Tem duas raízes no intervalo $[3.14, 3.15]$; $\varepsilon_1 \in [3.14, \pi]$ *e* $\varepsilon_2 \in [\pi, 3.15]$. Estes zeros não podem ser determinados numericamente, desde que

$$f(x) > 0$$

Para todo $x \in \mathbb{F}(2, 53, -1021, 1024, true)$ para a qual 1.8 pode se calculado. Os dois zeros de f,

$$x_1^* \approx \pi - 10^{-647}\ e\ x_2^* \approx \pi + 10^{-647}$$

estão localizados muito próximos de π, e $f(x) \leq 0$ é valido somente no intervalo localizado entre dois números de máquina. A implementação $\tilde{f} : \mathbb{F} \to \mathbb{F}$ não possui zeros (ver figura 1.1).

Exemplo (grande número de zeros [3], pag. 279)

O polinômio

$$P_7(x) = x^7 - 7x^6 + 21x^5 - 35x^4 + 35x^3 - 21x^2 + 7x - 1 =$$

$$= ((((((x-7)+21)x-35)x+35)x-21)x+7)x-1 = (x-1)^7$$

tem somente um zero em $x^* = 1$. A implementação de $\tilde{P}_7 : \mathbb{F} \to \mathbb{F}$ usando o esquema de Horner tem milhares de zeros na vizinhança de $x = 1$ [cálculo com $\mathbb{F}(2, 24, -125, 128, true)$, i.e., com uma aritmética de precisão simples IEC/IEEE]; para $x > 1$ existem milhares de pontos com valores $\tilde{P}_7 < 0$, embora $\tilde{P}_7 > 0$ nesta região e, similarmente, para $x < 1$ existem milhares de pontos com valores $\tilde{P}_7 > 0$, embora $\tilde{P}_7 < 0$ nesta região. A fenômeno do *Caos* surge do cancelamento dos dígitos mais significativos (ver figura 4.10 e 4.11 [2], pag 145).

Exemplo (solução do sistema linear inspirado em [3], pag. 233)

$$\mathbf{A} = \begin{pmatrix} 8 & 3 & & & & & 0 \\ 7 & 8 & 3 & & & & \\ & 7 & 8 & 3 & & & \\ & & . & . & . & & \\ & & & . & . & . & \\ & & & & 7 & 8 & 3 \\ 0 & & & & & 7 & 8 \end{pmatrix} \quad e\ \mathbf{b} = \begin{pmatrix} 11 \\ 18 \\ 18 \\ . \\ . \\ 18 \\ 15 \end{pmatrix}$$

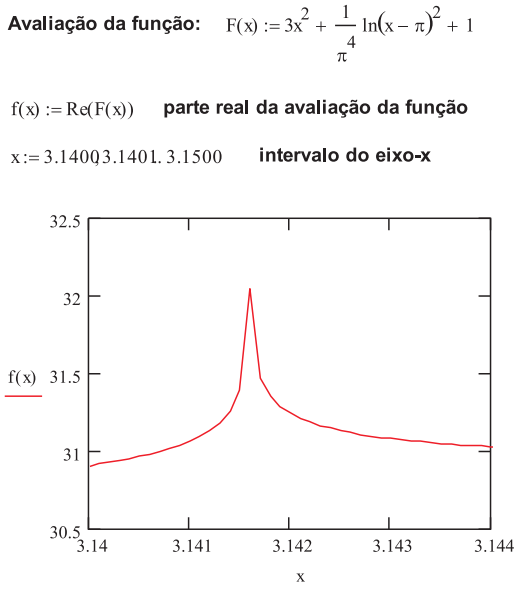

Figura 1.1: Planilha do software matemático Mathcad usada para traçar o gráfico da função $f(x) = 3x^2 + \frac{1}{\pi^4} ln(\pi - x)^2 + 1$. A função não tem zeros quando numericamente calculado com $\mathbb{F}(2,53,-1021,1024,true)$ (analiticamente tem dois)

A figura 1.2 mostra uma planilha Mathcad para resolução do sistema linear $\mathbf{A} \cdot \mathbf{x} = \mathbf{b}$. Embora o residual seja praticamente zero para todos as componentes do vetor solução, temos um resultado numericamente insatisfatório:

$$\mathbf{x} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ \cdot \\ \cdot \\ 1 \\ 1 \end{pmatrix} = \text{solução EXATA !}$$

Obs. A função ***verifylss*** do **Matlab Intlab toolbox** produziu os resultados de acordo.

Exemplo (avaliação de função)

Abaixo vamos avaliar a função $f(x,y) = 333.75 * y^6 + x^2 * (11 * x^2 * y^2 - y^6 - 121 * y^4 - 2) + 5.5 * y^8 + x/(2 * y)$ usando o Maple com precisão variada. A figura 1.3 mostra a planilha Maple onde só a partir de 37 dígitos é que os resultados apresentados são numericamente válidos.

Solução do sistema usando a função lsolve :

$x := \text{lsolve}(A, b)$

$x =$

	0
0	1
1	1
2	1
3	1
4	1
5	1
6	1
7	1
8	1
9	1
10	1
11	1
12	1
13	1
14	1
15	1

$x =$

	0
68	1
69	1
70	1
71	1
72	1.001
73	0.999
74	1.001
75	1
76	0.999
77	1.004
78	0.991
79	1.013
80	0.985
81	1.008
82	1.012
83	0.948

$x =$

	0
84	1.11
85	0.828
86	1.202
87	0.862
88	0.896
89	1.598
90	-0.353
91	3.213
92	-1.744
93	3.154
94	1.66
95	-5.786
96	17.555
97	-27.313
98	37.873
99	-31.264

Resíduo:

$A \cdot x - b =$

	0
84	0
85	0
86	0
87	0
88	0
89	0
90	0
91	0
92	0
93	0
94	0
95	0
96	0
97	$-3.553 \cdot 10^{-15}$
98	$7.105 \cdot 10^{-15}$
99	$-2.132 \cdot 10^{-14}$

Figura 1.2: Software matemático Mathcad usado na solução do sistema linear

```
> f:=(x,y)->333.75*y^6+x^2*(11*x^2*y^2-y^6-121*y^4-2)+5.5*y^8+x/(2*y):
> y:=33096:  x:=77617:
> Digits=10:
> f(x,y);
                    -298.827396059946821368141165095479 82

> Digits:=20:
> f(x,y);
                    0.10000000000000000117 10^18

> Digits:=30:
> f(x,y);
                    0.10000001172603940053178631858 8 10^8

> Digits:=40:
> f(x,y);
                    -0.827396059946821368141165095479816291999

> Digits:=37:
f(x,y);
                    -0.82739605994682136814116509547981629 2

> Digits:=36:
f(x,y);
                    21.17260394005317863185883490452018 37

> Digits:=35:
f(x,y);
                    -298.82739605994682136814116509547 982
```

Figura 1.3: Software matemático Maple usado na avaliação da função $f(x,y) = 333.75 * y^6 + x^2 * (11 * x^2 * y^2 - y^6 - 121 * y^4 - 2) + 5.5 * y^8 + x/(2 * y)$

Bibliografia

[1] R.W. Hamming, E.A. Feigenbaum Introduction to applied numerical analysis. McGraw-Hill, Inc New York (1971)

[2] C.W. Ueberhuber: Numerical Computation: Methods, software and analysis. Springer Berlin Heidelberg (1997) Vol. 1 474 pages

[3] C.W. Ueberhuber: Numerical Computation: Methods, software and analysis. Springer Berlin Heidelberg (1997) Vol. 2 495 pages

[4] J.F. Traub, H. Wozniakowski: On the Optimal Solution of Large Linear Systems. J. Assoc. Comp. Mach. 31 (1984), pp. 545-559.

[5] R.E. Moore: Interval Analysis. Prentice Hall, Englewood Clifs, NJ, USA (1966)

[6] G. Alefeld, J. Herzberger: Introduction to Interval Computations. Academic Press, (1983)

[7] U.W. Kulish and W.L. Miranker: The Arithmetic of the Digital Computers: A New Approach. SIAM Review **28**, 1 (1986)

[8] Rademacher, H, A.: On the accumulation of errors in processes of integration on high-speed calculating machines. Procedings of a symposium on large-scale digital calculating machinary. Annals Comp. Labor. Havard Univ. **16** (1948) pp 176-185

[9] Sterbenz, P.H.: Floating Point Computation. Prentice Hall, Englewood Clifs, NJ, USA (1974)

[10] Bailey, D.H.: MPFUN - A portable High Performance Multiprecision Package. NASA Ames Tech. Report RUR-90-022, (1990)

[11] Goldeberg, D: What Every Computer Scientist Should Know About Floating-Point Arithimethic. ACM Computing Surveys, **23** (1991) pp 5-48