PUC

Série: Monografias em Ciência da Computação
Nº 11/77

EFEITOS DE ERROS NO CÁLCULO DE ZEROS DE POLINÔMIOS

por

Emmanuel Piseces Lopes Passos

Departamento de Informática

Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de São Vicente 225 — ZC 19
Rio de Janeiro — Brasil

Série: Monografias em Ciência da Computação
Nº 11/77

EFEITOS DE ERROS NO CÁLCULOS DE ZEROS DE POLINÔMIOS

por

Emmanuel Piseces Lopes Passos*

Editor da Série: Michael F. Challis

Agosto 1977

* Professor do Laboratório de Projetos em Computação do CTC-PUC/RJ.

Para obter cópias dirija-se a:

Rosane T. L. Castilho
Chefe, Setor de Documentação e Informação
Depto. de Informática - PUC/RJ
Rua Marquês de São Vicente, 209 - Gávea
20.000 - Rio de Janeiro - RJ - BRASIL

RESUMO:

Esse trabalho propõe-se a realizar um estudo sintetizado sobre erros de arredondamento em Computação e seus efeitos no Cálculo de Zeros de Polinômio, em Computadores Digitais.

PALAVRAS-CHAVE:

Erros de arredondamento, computação em ponto-fixo, computação em ponto flutuante, arredondamento com acumuladores de precisão simples.

ABSTRACT:

The purpose of this paper is to make a study about Rounding Erros in Computation and its effects in the zeros finds polinomials in Digital Computation.

KEYWORDS:

Rounding erros, fixed-point computation, floating-point computation, round-off single-precision accumulator, effect of rouding errors on zero find polynomials.

SUMÁRIO

I. INTRODUÇÃO

Esse trabalho propõe-se a realizar um estudo sintetizado sobre erros de arredondamento em computação e seus efeitos no cálculo de Zeros de Polinômio, em Computadores Digitais.

A primeira parte será dedicada a um estudo resumido sobre erros de arredondamento nas operações aritméticas fundamentais. A segunda parte será um estudo das condições de um Polinômio com respeito a computação de seus ZEROS, baseado nos resultados da primeira parte.

II. ESTUDO SOBRE ERROS DE ARREDONDAMENTO NAS OPERAÇÕES ARITMÉTICAS FUNDAMENTAIS

Existem duas modalidades de operação, comumente usadas em computadores digitais: operação em ponto fixo e operação em ponto flutuante.

Estamos interessados em estudar os erros de arredondamento ocorridos nas operações aritmética fundamentais, como ferramenta ao leitor para estudo dos erros nas operações aritméticas decorrentes.

Consideramos, em nosso estudo, números expressos em bases binária e decimal, estando sua representação relacionada, portanto, a um número fixado t, de dígitos binários ou decimais, consideramos ainda computadores que trabalham com palavras de t dígitos e analisamos a precissão obtida nos resultados, sob dois aspectos:

- precisão simples, quando é possível a utilização de acumuladores de uma palavra;

- precisão múltipla, quando for necessário a utilização de acumuladores múltiplos de t, para se conseguir melhores resultados.

II.1. Computações em Ponto Fixo

Em ponto fixo, operamos com números referidos a um formato interno no qual a precisão, ou seja, o número de dígitos é fixo.

Podemos operar em ponto fixo, sempre que conseguirmos predefinir qual a amplitude e a precisão necessária de nossos números.

Nesse tipo de operação, cada número x operado é representado por um número t fixo, de dígitos binários ou decimais e deve satisfazer à seguinte condição:

$$-1 \leq x < 1$$

Essa condição é devido a que estamos trabalhando com a aritmética fracionária módulo 1, cuja representação (n, m)

- n º de bits após o ponto
- nº de bits antes do ponto (o 1º bit é de sinal, 1 sinal negativo, 0 sinal positivo)

Na aritmética fracionária, não temos nehum bit antes do ponto, logo todos estão depois do ponto. (1, t-1), isso nos informa que essa é uma representação binária de um número menor que 1 e maior que -1, isto é, (1, t-1) representa $-1 \leq x < 1$.

Sabemos que qualquer número pode ser reduzido a esse intervalo através de transformações de formato. Por exemplo, se temos um formato $(n-j_1, j_1)$ que representa um número maior ou igual a um $(y \geq 1)$ podemos representá-lo no formato (1, n-1) que é a nossa aritmética, através da fórmula de transformação $(n-j_1, j_1) = 2^{n-1-j_1}.(1,n-1)$, onde j_1 é inteiro positivo.

II.1.1. Erros de representação de números no computador

Quando representamos um número decimal em binário, usamos rotinas de transformações de BASE. O erro que irá ocorrer na representação desse nº decimal no computador binário, dependerá de quão bem feitas são essas rotinas.

decimal → binário → decimal

Essas rotinas estão documentadas nos manuais da linguagem em cada computador. A partir do estudo dessas rotinas, nós sabemos que erro cometemos, já, na representação do número no computador. O algoritmo em geral é tirado da fórmula que converte um nº na base 10 para um número na base 2.

II.1.2. Erros de arredondamento nas operações aritméticas fundamentais

Consideramos dois números x_1 e x_2, representados por t dígitos e definidos no intervalo $[-1, 1)$; o resultado das operações é representado por t-dígitos.

Adição e Subtração não envolvem arredondamento, em aritmética de ponto fixo, o que pode ocorrer é um overflow.

Equação matemática: $x_3 = x_1 \pm x_2$

Equação computacional: $f_i\ (x_1 \pm x_2) = x_1 \pm x_2$

A soma ou subtração computada podem não estar definidas no intervalo permitido.

Na multiplicação o produto exato $x_1 x_2$ é um número que requer 2t dígitos para sua representação.

O produto arredondado é obtido somando-se $\frac{1}{2}\,2^{-t}$ ou $\frac{1}{2}\,10^{-t}$ ao produto exato e desprezando-se os dígitos de t+1 a 2t.

Equação matemática: $x_3 = x_1 x_2$

Equação computacional: $f_i\ (x_1 x_2) \equiv x_1 x_2 + \varepsilon$

$|\varepsilon| \leq \frac{1}{2}.2^{-t}$ ou $|\varepsilon| \leq \frac{1}{2}.10^{-t}$

O produto exato e o produto arredondado são, sempre, definidos no intervalo permitido.

Exemplo:

$x_1 = 0{,}4325$ e $x_2 = 0{,}6235$ produto exato $x_1 x_2 = 0{,}26966375$

produto arredondado: $x_1 x_2 + \varepsilon = 0{,}2697$

erro absoluto $\varepsilon_a = 0{,}2697 - 0{,}26966375 = 0{,}00003625$

$\varepsilon_a \leq \frac{1}{2}.10^{-4}$

Na divisão o quociente exato x_1/x_2 é um número para o qual, em geral, não se pode definir o número de dígitos necessários à representação.

O quociente arredondado é obtido somando-se $\frac{1}{2}.2^{-t}$ ou $\frac{1}{2}.10^{-t}$ ao quociente exato e retendo-se, apenas, os dígitos de 1 a t. Para tal cálculo, precisamos considerar, somente, os t+1 dígitos do quociente exato.

Equação matemática: $x_3 = x_1/x_2$

Equação computacional: $f_i(x/x) = \frac{x_1}{x_2} + \varepsilon$

$|\varepsilon| \leq \frac{1}{2} 2^{-t}$ ou $|\varepsilon| \leq \frac{1}{2}.10^{-t}$

O quociente x_1/x_2 não estará definido no intervalo permitido, a menos que $|x_2| \geq |x_1|$.

Exemplo:

$x_1 = 0,2278$ e $x_2 = 0,8743$ $\quad |x_2| \geq |x_1|$

quociente exato: $\frac{x_1}{x_2} = 0,260551...$

quociente arredondado: $\frac{x_1}{x_2} + \varepsilon = 0,2606$

erro absoluto: $\varepsilon_a = 0,2606 - 0,260551... = 0,000048...$, $\varepsilon_a < \frac{1}{2}.10^{-4}$

II.1.3. Erros de arredondamento em acumulação de produtos internos

O produto de dois números representados por t-dígitos é exatamente representado por 2t-dígitos e muitos computadores se utilizam deste fato, para obtenção de resultados mais precisos, fazendo uso de acumuladores internos com 2t-dígitos.

Equação matemática do produto interno:

$$S = \sum_{1}^{n} x_i y_i$$

Equação computacional:

$$S = \sum_{1}^{n} x_i y_i + \varepsilon$$

Se o arredondamento for efetuado, a cada produto intermediário, teremos como limite para ε

$|\varepsilon| \leq \frac{1}{2}\, n.2^{-t}$ ou $|\varepsilon| \leq \frac{1}{2}\, n.10^{-t}$, esse é o caso normal.

Se, no entanto, cada produto intermediário for calculado com 2t-dígitos, sendo efetuado apenas um arredondamento ao final, teremos:

$$|\varepsilon| \leq \frac{1}{2}\, .2^{-t} \quad \text{ou} \quad |\varepsilon| \leq \frac{1}{2}\, .10^{-t}$$

Acumuladores de 2t-dígitos podem ser utilizados para representar o dividendo, em uma divisão, propiciando quociente corretamente arredondado, em t-dígitos. Quando o dividendo é um número com t-dígitos, deve ser expandido para 2t-dígitos, adicionando-se t zeros.

Consideremos a equação matemática:

$$W = \left(\sum_{1}^{n} x_i y_i\right) / Z$$

Se utilizarmos as facilidades descritas para o produto e a divisão, teremos como equação computacional:

$$f_i\left(\left(\sum_1^n x_i y_i\right)/Z\right) \equiv \left(\sum_1^n x_i y_i\right)/Z + \varepsilon \qquad \text{com} \quad |\varepsilon| \leq \frac{1}{2}.2^{-t} \quad \text{ou}$$

$$|\varepsilon| \leq \frac{1}{2}.10^{-t}$$

Exemplo:

$x_1 = 0,4124$; $y_1 = 0,6835$;

$x_1 = 0,5823$; $y_1 = 0,6141$;

$Z = 0,8467$;

$W = (0,4124x0,6835+0,5823x0,6141)/0,8467 =$

$= (0,28187540+0,35759043)/0,8467 = 0,63946583/0,8467=$

$= 0,75524...$

O valor computado de W é 0,7552 com erro de 0,00004...

Se o arredondamento for efetuado, a cada produto intermediário, teremos como dividendo: 0,2819+0,3576 = 0,6395

Logo:

$W = 0,6395/0,8467 = 0,75528...$

O erro encontrado, neste caso, é de 0,00008... em relação ao valor de W, anteriormente, computado.

O valor computado de W, desta forma, é 0,7553.

II.2. Computações de Ponto Flutuante

Utiliza-se a representação interna de ponto flutuante, em cálculos nos quais um mesmo número pode variar num intervalo muito amplo, assumindo valores muito grandes ou muito pequenos e também quando a sua amplitude não é conhecida.

Na operação de ponto flutuante, cada número x operado, é representado por um par a, b ordenado, na forma (a, b) = x = 2^b (a), x binário ou (a, b) = x = 10^b (a) para x decimal, onde b é o expoente ou índice e a é a mantissa ou parte fracionária.

Nessa representação, considera-se b inteiro, positivo ou negativo, e a um número que satisfaça à: $-\frac{1}{2} > a \geq -1$ ou $\frac{1}{2} \leq a < 1$ x binário

$-\frac{1}{2} \geq a > -1$ ou $\frac{1}{10} \leq a < 1$ x decimal

A mantissa de um número em ponto flutuante é representada por um número de t dígitos binários ou decimais.

Um número (a,b) é dito ser normalizado se o dígito mais significativo da representação de a (parte fracionária) é difererente de zero. Isto é, igual a condição de acima.

Exemplo: Número de Avogrado N = (+ .60225,23)

ou um exemplo em decimal:

nº negativo 1 0 .....

nº positivo 0 1 .....

II.2.1. Erros de representação de números em ponto flutuante no computador

Como no caso de ponto fixo, depende do método de converter decimal a binário e vice-versa, para sabermos que tipo de erro cometemos ao representar o nº no computador.

II.2.2. Erros de arredondamento nas operações aritméticas fundamentais

Tratamos as operações em ponto flutuante, levando em conta acumuladores de precisão dupla e simples respectivamente. Consideramos os operandos x_1 e x_2 representados por $x = 2^{b_1}.a_1$ ou $10^{b_1}.a_1$ e $x_2 = 2^{b_2}.a_2$ ou $10^{b_2}.a_2$ sendo a_1 e a_2 representados por t-dígitos e definidos no intervalo permitido para a mantissa.

O que estudaremos agora, são os limites dos erros nessas operações aritméticas lembrando que esses estudos são feitos do ponto de vista matemático, isto é, considerados erros somente nos resultados das operações, e não na representação dos operandos.

Por exemplo: $\overline{a + b}$ soma correta com arredondamento no final

$\bar{a} \oplus \bar{b}$ soma com erro; operandos com erro. $(\bar{a} = a(1 + \varepsilon_a))$

Portanto numa soma $\overline{a + b} = \bar{a} \oplus \bar{b} = (\bar{a} + \bar{b})\ (1 + \varepsilon_t) = \ldots.$

$\ldots = [a + b]\ (1 + \varepsilon_t) + \underbrace{(a.\varepsilon_a + b.\varepsilon_b)\ (1 + \varepsilon_t)}.$

não levaremos em conta esse erro

caso (i) de II.2.1.2.

II.2.2.1. Precisão dupla

Adição e Subtração

Supondo $|x_1| > |x_2|$, a soma é obtida do seguinte modo:

i) se $b_1 - b_2 > t$ então:

- x_2 é muito pequeno e não influi, enquanto considerarmos os t primeiros dígitos significativos, a soma é então, idêntica a x_1 e exige t dígitos para sua representação.

ii) se $b_1 - b_2 \leq t$ então:

- a_2 é dividido por $2^{b_1-b_2}$ ou por $10^{b_1-b_2}$, deslocando-se $b_1 - b_2$ posições para a direita;

- a soma exata $a_1 + 2^{b_2-b_1} . a_2$ ou $a_1 + 10^{b_2-b_1} . a_2$ é calculada e requer, no máximo, 2t dígitos para sua representação.

- se necessário, este resultado deve ser, então, normalizado, de modo que a mantissa fique definida no intervalo permitido.

- a mantissa é então, arredonda para t dígitos, obtende-se a soma arredondada.

Se a soma exata normalizada é $2^{b_3} . a_3$ ou $10^{b_3} . a_3$ então o erro absoluto é limitado por $2^{b_3} . \frac{1}{2} . 2^{-t}$ ou por $10^{b_3} . \frac{1}{2} . 10^{-t}$

Mais frequentemente, utilizaremos a fórmula do erro relativo, que podemos deduzir, considerando que:

- os módulos do ERRO é limitado por:
  $\frac{1}{2} . 2^{-t} \times 2^{b_3}$ ou $\frac{1}{2} . 10^{-t} \times 10^{b_3}$ (erro absoluto)

- erro relativo = $\frac{\text{erro absoluto}}{\text{valor exato}}$ (usaremos mais frequentemente)

- o maior valor do erro relativo é obtido considerando-se o menor valor exato;

$$[\varepsilon] \leq \frac{2^{b_3} \cdot \frac{1}{2} \cdot 2^{-t}}{\frac{1}{2} \cdot 2^{b_3}} \rightarrow \underline{[\varepsilon] \leq 2^{-t}} \quad \text{ou} \quad [\varepsilon] \leq \frac{10^{b_3} \cdot \frac{1}{2} \cdot 10^{-t}}{\frac{1}{2} \cdot 10^{b_3}} \rightarrow [\varepsilon] \leq \frac{1}{2} \cdot 10^{1-t}$$

Equação computacional: $f_1 (x_1 + x_2) \equiv (x_1 + x_2)(1 + \varepsilon)$ para algum ε satisfazendo a $[\varepsilon] \leq 2^{-t}$ ou $[\varepsilon] \leq \frac{1}{2} \cdot 10^{1-t}$

Os resultados da subtração são análogos aos da adição.

Exemplos:

i) $x_1 = 10^{-8}\ (0{,}4735)$ e $x_2 = 10^{3}\ (0{,}8318)$

$f_1 (x_1 + x_2) = 10^3\ (0{,}8318)$

ii) $x_1 = 10^{6}\ (0{,}4785)$ e $x_2 = 10^{3}\ (0{,}7839)$

soma exata: $10^6 \times 0{,}47850000$

$+ \underline{10^6 \times 0{,}00078390}$

$10^6 \times 0{,}47928390$

soma arredondada: $10^6 \times 0{,}4793$

$$\varepsilon_a = 10^6\ (0{,}4793) - 10^6\ (0{,}47928390) = 10^6 \times 0{,}00001610 < 10^6 \times \frac{1}{2} \cdot 10^{-4}$$

$$\varepsilon_r = \frac{10^6 \times 0{,}00001610}{10^6 \times 0{,}47928390} = \frac{10^2 \times 0{,}1610}{10^6 \times 0{,}47928390} = 10^{-4} \times 0{,}33590650 < \frac{1}{2} \cdot 10^{-3}$$

iii) $x_1 = 10^3(0,9732)$ e $x_2 = 10^2(0,4827)$

soma exata: $10^3 \times 0,07320000$ / $10^3 \times 0,04827000$ / $10^3 \times 1.02147000$

soma normalizada e arredondada $10^2 \times 0,1021$

$$\varepsilon_a = 10^2 \times 0,1021 - 10^3 \times 10214700 = 10^2 \times 0,00004700 < 10^2 \times \frac{1}{2}\, 10^{-4}$$

Multiplicação

O produto $x_1 x_2$ representado por $2^{b_3}.a_3$ ou $10^{b_3}.a_3$ é obtido de seguinte modo:

- $b_3 = b_1 + b_2$

- o produto exato $a_1 . a_2$ é calculado e requer 2t dígitos para sua representação;

- se necessário esse produto deve ser normalizado;

- a mantissa é arredondada para t dígitos, obtendo-se o produto arredondado.

Equação Computacional

$f_1(x_1 x_1) = x_1 x_2 (1+\varepsilon)$ e analogamente, à soma e subtração, temos:

$$[\varepsilon] \leq 2^{-t} \quad \text{ou} \quad [\varepsilon] \leq \frac{1}{2}.10^{1-t}$$

Exemplo:

$x_1 = 10^2(0,1375)$ e $x_2 = 10^3(0,1482)$

Produto exato: $(10^2 \times 0,1375) \times (10^3 \times 0,1482) = 10^4 \times 0,02037750$

Produto normalizado: $10^4 \times 0,20377500$

Produto arredondado: $10^4 \times 0,2038$

$\varepsilon_a = 10^4 \times 0{,}2038 - 10^5 \times 0{,}02037750 = 10^4 \times 0{,}2038 - 10^4 \times 0{,}20377500$

$\varepsilon_a = 10^4 \times 0{,}00002500 < \frac{1}{2}\, 10^{-3}$

Divisão

O quociente x_1/x_2, representado por $2^{b_3}.a_3$ ou $10^{b_3}.a_3$ é obtido do seguinte modo:

- $b_3 = b_1 - b_2$; a é colocado nos t dígitos mais significativos (mais alta ordem) do acumulador, preenchendo-se com zero, os t dígitos restantes;

- se $|a_1| > |a_2|$, o número no acumulador é deslocado de uma posição para a direta e b_3 é incrementado de 1;

- o número no acumulador é dividido por a_2 e o resultado é obtido na forma normalizada;

- a mantissa é arredondada para t dígitos, obtendo-se o quociente arredondado.

Equação computacional: $f_1(x_1/x_2) \equiv (x_1/x_2)(1+\varepsilon)$ e, analogamente às operações anteriores: $[\varepsilon] \leq 2^{-t}$ ou $[\varepsilon] \leq \frac{1}{2}\, 10^{1-t}$

Exemplo:

$x_1 = 10^3(0{,}8714)$ e $x_2 = 10^3(0{,}1718)$

quociente exato normalizado: 10x0,08714000/0,1718 = 10x0,50721....

quociente arredondado: 10x0,5072

$\varepsilon_a = 10 \times 0{,}5072 - 10 \times 0{,}50721... = 10 \times 0{,}00001...$

$\varepsilon_r = 10 \times 0{,}00001.../10 \times 0{,}50721...$

$\varepsilon_a < 10 \times \frac{1}{2} \cdot 10^{-4}$ $\qquad E_\eta < \frac{1}{2} \cdot 10^{-3}$

II.2.2.2. Precisão simples

As operações efetuadas com precisão simples obedecem, basicamente, à mesma sequencia definida para as operações com precisão dupla, com a diferença de que se utilizam de acumuladores com t dígitos. As operações deste tipo, apresentam, em geral, um erro relativo maior.

Adição e Subtração

A majoração para o erro relativo obedece a duas condições básicas:

i) quando é cometido, apenas, um erro de arredondamento considere o exemplo ii da adição e subtração de precisão dupla, o resultado nesse caso é o mesmo. Mas no exemplo:

$x_1 = 10^3(0,1024)$ e $x_2 = 10(-0,8933)$

Precisão dupla	Precisão simples
soma exata $= 10^3 \times 0,09346700$	soma exata $= 10^3 \times 0,0935$
soma normalizada $= 10^2 \times 0,93467000$	soma normalizada $= 10^2 \times 0,9350$
soma arredondada $= 10^2 \times 0,9347$	o erro cometido é como podemos ver, maior que o obtido com operação de precisão dupla
$\varepsilon_a = 10^2 \times 0,00003 < 10^2 \times \frac{1}{2} \times 10^{-4}$	
$\varepsilon_r < \frac{1}{2} .10^{-3}$	

ii) quando são cometidos dois erros de arrendodamento η_1 e η_2. η_1 é o erro de arredondamento cometido em um dos operandos, η_2 é o erro cometido no resultado. Nesse caso teremos

Equação computacional $f_1(x_1 + x_2) \equiv (x_1 + x_2)(1+\varepsilon_+) + \varepsilon_0$ onde

$[\varepsilon_+] \leq (1. \frac{1}{2}).2^{-t}$ ou $[\varepsilon_+] \leq \frac{1}{2} . 10^{1-t}$ e

$\varepsilon_0 = (x_1 . \varepsilon x_1 + x_2 . \varepsilon x_2)(1+\varepsilon_+)$

Exemplo:

$x_1 = 10^3(0,9732)$ e $x_2 = 10^2(0,4827)$

Soma:

$10^3 \times 0,9732$ soma normalizada e arredondada

$\underline{10^3 \times 0,0483}$ $10^4 \times 0,1022$

$10^3 \times 10215$

Multiplicação e Divisão

A multiplicação e a divisão são afetadas em menor grau que a adição e a subtração, com a precisão simples.

Equação computacional:

$$f_1(x_1 . x_2) \equiv x_1 . x_2 (1+\varepsilon)$$

$$f_1(x_1/x_2) \equiv (x_1/x_2).(1+\varepsilon) \qquad |\varepsilon| \leq 2^{1-t}$$

II.3. Considerações finais sobre erros de arredondamento em computação

Nós, efetivamente, começamos nossas operações com parâmetros aproximados $\hat{a}_i$ satisfazendo a $|\hat{a}_i - a_i| \leq \frac{1}{2} \cdot 2^{-t}$ (ponto fixo); $\left|\frac{\hat{a}_i - a_i}{a_i}\right| \leq 2^{-t}$ ponto flutuante. Sempre que os nossos operandos são representados exatamente por números com t-dígitos. Assim, se nos restringirmos rigidamente, ao uso de números com t-dígitos, limitaremos nossa precisão, mesmo antes de começarmos a operar, isto é, começaremos com uma aproximação de t-dígitos para o verdadeiro problema. Por exemplo, se considerarmos o polinômio $\sum_{i=0}^{n} a_i x_i = 0$, e se os parâmetros a_i, são arredondados, os efeitos nos cálculos dos zeros podem ser desastrosos, como veremos nesse texto a seguir. Esses problemas, em que mudanças relativamente pequenas (arredondamento) provocam comparativamente, grandes erros relativos nas soluções, chamamos problema mal condicionado.

O problema central na análise do erro é estabelecer limites para o efeito acumulado dos erros de arredondamento nas soluções, considerando-se que em geral, os erros de arredondamento são efetuados, a cada etapa do processo de obtenção da solução. É com essa diretriz que continuaremos nosso trabalho, estudando o efeito desses erros de arredondamento no cálculo dos zeros de um polinomio.

III. CONDIÇÃO DE UM POLINÔMIO COM RESPEITO À COMPUTAÇÃO DE SEUS ZEROS

III.1. Estado da arte no cálculo de zero de polinômios (Algoritmos e Programas)

Nos últimos anos tem aparecido muitos algoritmos para solução numérica de equações polinomiais. Muitos deles são soluções matemáticas interessantes, mas apenas poucos são de convergência rápida e confiáveis como "software" de aplicação (TRAUB-JENKINS-1974. "Principles for Testing Polynomial Zerofinding Programs").

III.1.1. Visão Geral do Problema de encontrar Zeros de Polinômios

Esse problema é computacionalmente difícil. Muitos algoritmos que trabalham com eficiência em exemplos de polinômios com zeros simples bem separados (uma raiz "distante" de outra) falham em exemplos mais difíceis.

Nós assumiremos aqui que discutiremos programas que calculam zeros de polinômios quaisquer, com coeficientes reais ou complexos.

Classificaremos os algoritmos naqueles que acham um único zero ou um par de zeros de cada vez ou naqueles que determinam todos os zeros simultaneamente. Precisaremos de usar alguma técnica de deflação explícita ou implicita (WILKINSON-1963-Rounding Erros in Algebraic Processes) para remover os zeros já determinados. A maioria dos algoritmos hoje existentes usam para esse problema um dos seguintes caminhos (ou combinações deles):

i) função de iteração - uma fração de iteração gera uma seqüência as quais convergem para um zero. Como exemplo temos: O método de Newton, de Bairstow, Laguerre, Muller etc.

ii) separação de zeros por potenciação - um algoritmo produz uma seqüência cujos elementos são coeficientes de um polinômio cujas raízes são potência das raízes do polinômio original que por sua vez é fácil calcular seus zeros. Exemplos temos método do Bernaulli e variações, método Graffe

ou Rott-squaring e o algoritmo QD.

iii) métodos topológicos - o plano complexo é dividido em regiões e fazem-se testes para determinar que região contem ou excluem zeros. As regiões que contem zeros são subdivididas em regiões cada vez menores é se obter o ZERO. Algumas dessas técnicas isolam um único zero (Lehmer 1961) outras acham todos os zeros simultaneamente (Henrici e Gargantini 1967).

iv) técnicas de minimização - utilizando as propriedades dos polinômios [Ostrowski 1967].

v) técnicas de fatorização - o polinômio é fatorado em dois ou mais fatores por algum critério. Um caminho é usar o algoritmo de EUCLIDES para "quebrar" o polinômio em fatores cada um dos quais tem zeros simples (Dunaway 1974).

É interessante ressaltar que a resolução de uma equação polinomial deve ser feita, utilizando-se operações de ponto-flutuante, porque os números utilizados nos cálculos podem variar em um intervalo muito amplo, com valores muitos grandes ou muito pequenos, e muitas vezes a sua amplitude não é conhecida. (Wilkinson, 1963).

Em quaisquer dos quatro métodos citados anteriormente, os coeficientes do polinômio obtido do polinômio original através de operações as mais diversas, possuem erros de arredondamento, os quais vão ser propagados e influirão na precisão das raízes obtidas em seguida. Esse problema pode complicar muito se a equação é mal condicionada, isto é, pequenas mudanças nos coeficientes implicam grandes mudanças nos ZEROS.

Na parte seguinte do trabalho será tratado o problema de polinômios mal condicionados.

III.2. Condicionamento de Polinômios

Quando pequenas mudanças nos coeficientes de um polinômio produzem grandes erros relativos nas raízes, dizemos que o polinômio é mal condicionado. O condicionamento de um polinômio é determinado pelo grau de sensibilidade da solução às pertubações ocorridas nos coeficientes.

No exemplo dado por Wilkinson (1963)

$$P(Z) = (Z-1)(Z-2)(Z-3)\ldots(Z-19)(Z-20) = Z^{20} - 210Z^{19} + \ldots + 20!$$

i) se substituirmos $a_1 = -210$ por $-210-2^{-23} = -210,0000001192$

$Z_1 = 1,000000000$
$Z_2 = 2,000000000$
$Z_3 = 3,000000000$
$Z_4 = 4,000000000$
$Z_5 = 5,999999928$
$Z_6 = 6,000006944$
$Z_7 = 6,999697234$
$Z_8 = 8,007267603$
$Z_9 = 8,917250249$

$Z_{10,11} = 10,095266145 \pm 0,643500904\ i$

$Z_{12,13} = 11,793633881 \pm 1,652329728\ i$

$Z_{14,15} = 13,992358137 \pm 2,518830070\ i$

$Z_{16,17} = 16,730737466 \pm 2,812624894\ i$

$Z_{18,19} = 19,502439400 \pm 1,940030347\ i$

$Z_{20} = 20,846908101$

* Uma mudança no coeficiente de Z^{19} fez com que encontrassemos 5 pares de números complexos conjugados e todos os zeros do polinômio original são reais.

ii) se substituirmos

$a_1 = -210$ por $210-2^{-55}$ as raízes serão

1.	10.999999999
2.	12.000000006
3.	12.999999999
4.	14.000000037
5.	14.999999983
6.	16.000000067
7.	16.999999947
8.	18.000000028
9.	18.999999991
10.	20.000000001

III.2.1. Condição do Problema

Consideramos o problema do cálculo dos n zeros de um polinômio.

(2-1) $$f(Z) = a_n Z^n + a_{n-1} Z^{n-1} + \ldots + a_1 Z + a_0$$

cujos dados iniciais são os coeficientes dos polinômio.

O objetivo dessa parte do trabalho é fazer um rápido estudo sobre a sensibilidade das raízes em relação as mudanças sofridas pelos coeficientes .

Consideremos então o polinômio

(2.2) $$P(Z) = f(Z) + g(Z) \quad \text{onde} \quad g(Z) = b_n Z^n + b_{n-1} Z^{n-1} + \ldots + b_0$$

$$f(Z) = a_n Z^n + \ldots + a_0 .$$

I. Caso do ZERO simples

Teorema:

"Se Z_r é um zero isolado de f(Z), então um zero $Z_r(\varepsilon)$ de P(Z) é dado por:

$$(2.3) \qquad Z_r(\varepsilon) = Z_r + \sum_{1}^{\infty} p_k \varepsilon^k = Z_r + h$$

onde a série em ε é convergente para um ε suficientemente pequeno".

Da relação (2.3) podemos escrever $Z_r(\varepsilon) - Z_r =$

$$= p_1\varepsilon^1 + \sum_{1}^{\infty} p_k\varepsilon^k \rightarrow \left| \sum_{k=2}^{\infty} p_k\varepsilon^k \right| = \varepsilon^2 . \sum_{k=2}^{\infty} p_k . \varepsilon^{k-2} \Big| \leq |Z_r(\varepsilon)-(Z_r+p_1\varepsilon)| \leq \varepsilon^2 K \ ,$$

$\varepsilon \downarrow 0$

portanto

$$(2.4) \qquad Z_r(\varepsilon) - Z_r \approx p_1\varepsilon$$

Então vemos que a pertubação sofrida pela raiz é aproximadamente igual a $p_1.\varepsilon$.

Para facilitar a notação façamos $Z_r(\varepsilon)-Z_r = \delta Z_r$ como a mudança sofrida por um coeficiente a_k é aproximadamente igual a ε , se chamarmos essa pertubação do coeficiente a_k de δa_k , podemos escrever $\delta.Z_r \approx p_1.\delta a_k$, logo p_1 é aproximadamente igual a condição do problema.

Cálculo de p_1

Se $Z_r(\varepsilon)$ é uma raiz de P(Z) então $P(Z_r(\varepsilon)) = 0$, logo de (2.2) e (2.3) podemos escrever na forma

$$\sum_{i=1}^{n} a_i (Z_r+h)^i + \varepsilon \sum_{i=1}^{n} b_i (Z_r+h)^i = 0$$

ou então $\sum_{0}^{n} C_k h^k + \varepsilon \sum_{0}^{n} d_k h^k = 0$

onde: (A) $= C_k = \frac{1}{k!} f^{(k)}(Z_r)$ e $d_k = \frac{1}{k!} g^{(k)}(Z_r)$

Substituindo o valor de h dado por (2.3) em (2.5) obtemos:

$$(2.6) \qquad \sum_{k=0}^{n} C_k (p_1\varepsilon^1 + p_2\varepsilon^2 + \ldots)^k + \varepsilon \sum_{k=0}^{n} d_k (p_1\varepsilon + p_2\varepsilon^2 + \ldots)^k = 0$$

como a série de potência em h é convergente para ε suficientemente pequeno, a série em (2.6) deve ser identicamente nula.

Verificamos em (2.5) que $C_0 = 0 = f(Z_r)$. (Z_r,raiz). Então igualando o coeficiente de ε a zero, obtemos:

$c_1 p_1 + d_0 = 0$ e de (A)

$$(2.7) \qquad \boxed{p_1 = -\frac{g(Z_r)}{f'(Z_r)}}$$

$$(2.8) \qquad \boxed{\delta Z_r = \frac{-g(Z_r)}{f'(Z_r)} \cdot \delta a_k} \qquad \text{pois } \delta Z_r = p_1 \delta a_k.$$

De (2.8) verificamos que se $f'(Z_r)$ é suficientemente pequeno, o coeficiente de δa_k pode ser bastante grande e o problema será mal condicionado.

II. Caso do zero múltiplo

Teorema:

"Se Z_r é um zero de multiplicidade m de f(Z), então existem m zeros de P(Z) definidos por

(2.9) $$Z_r(\varepsilon) = Z_r + \sum_{k=1}^{\infty} p_k \cdot \varepsilon^{k/m} = Z_r + h$$ (da teoria das funções algébricas). Cada zero corredpondendo a uma interpretação de $\varepsilon^{1/m}$ e onde a série em ε é convergente para um ε suficientemente pequeno.

- Condição do problema: de (2.9) podemos escrever
$|Z_r(\varepsilon) - (Z_r + p_1 \cdot \varepsilon^{1/m})| < k.e^{2/m}$ para ε suficientemente pequeno, então:

(2.10) $$Z_r(\varepsilon) - Z_r \approx p_1 \cdot \varepsilon^{1/m}$$

Ou seja $\delta Z_r \approx p_1 (\delta a_r)^{1/m}$ quanto maior for o m mais lentamente $(\delta a_r)^{1/m} \to 0$.

(temos $\sqrt[m]{\delta a_r}$, com $m=\infty$ $\delta a_r \to 1$).

Cálculo de p_1

Usando o mesmo procedimento anterior, e observando que

$f(Z_r) = f'_r(Z_r) = f''_r(Z_r) = \ldots = f^{m-1}(Z_r) = 0$ temos

(m raízes múltiplas)

$$\sum_{n}^{n} C_k h^k + \varepsilon \sum_{1} d_k h^k = 0$$

Substituindo h por $\sum_{1}^{\infty} p_k \varepsilon^{k/m}$ e igualando o coeficiente de $(\varepsilon^{1/m})^m$ a zero, temos $C_m p_1^m + do = 0 \rightarrow p_1 = \left| -\frac{do}{C_m} \right|^{1/m}$

ou $$p_1 = \frac{-m!\,g(Z_r)}{f^{(m)}(Z_r)}$$

III. Caso particular: raiz dupla

Se a raiz é dupla, então m=2 , e portanto temos:

$$p_1 = \frac{-2.g(Z_r)}{f''(Z_r)}$$

Os zeros pertubados podem ser escritos na forma:

$$\alpha = Z_r + p_1 \varepsilon^{1/2} + p_2 \varepsilon + \ldots.$$

$$\beta = Z_r - p_1 \varepsilon^{1/2} + p_2 \varepsilon + \ldots.$$

O fator quadrático correspondente é:

$$Z^2 - 2(Z_r + 2p_2\varepsilon + \ldots)Z + (Z_r^2 + (2p_2 Z_r - p_1^2)\varepsilon + \ldots) = 0$$

Verificamos que a pertubação nos coeficientes do fator quadrático é de ordem ε, e portanto o fato de raízes duplas serem mal condicionadas não implica que o fator quadrático correspondente seja mal condicionado.

III.2.2. Distribuição Linear dos ZEROS

1º exemplo: Consideremos a distribuição 1,2,...,20

$$f(Z) = (Z-1)(Z-2)\ldots(Z-20)$$

Seja ε a pertubação no coeficiente a_k. Aplicando (2.8) obtemos:

$$(2.12) \qquad \delta Z_r = -\varepsilon \frac{(Z_r)^k}{f'(Z_r)} = \pm\varepsilon \frac{r^k}{(20-r)!(r-1)!} = \pm\varepsilon . A$$

A é máximo quando $K = 19$. Fixando K = 19, teremos;

Para r=16, A tem seu valor máximo aproximadamente igual a

$$0{,}24 \times 10^{10} = \frac{16^{19}}{4!15!} .$$

Para r=1, A tem seu valor mínimo aproximadamente igual a

$$0{,}82 \times 10^{-17} = \frac{1}{19!} .$$

Então para uma mudança no coeficiente de Z^{19} as raízes maiores são bastante sensíveis, enquanto as raízes menores são praticamente insensíveis. Veja exemplo i) de III.2.

Fazendo dessa maneira, nós temos considerado o efeito do mesmo erro em cada coeficiente. Do ponto de vista de aplicações práticas isso é irrealístico. O mais próximo do real é o caso em que os coeficientes tem o mesmo erro relativo. Se computarmos o polinomio usando FLOATING-POINT, então o erro de arredondamento em um coeficiente a_k estará limitado por $\pm 2^{-t}.a_k$ (t dígitos binários).

Aplicando (2-12) temos $\delta Z_r \approx 2^{-t} a_k \cdot \frac{r^k}{(20-r)!(r-1)!}$

Para avaliarmos esta expressão precisamos conhecer a ordem de grandeza dos coeficientes que é a seguinte:

$$Z^{20} + 10^3 Z^{19} + 10^5 Z^{18} + 10^7 Z^{17} + 10^8 Z^{16} + 10^{10} Z^{15} + 10^{11} Z^{14} + 10^{12} Z^{13} + 10^3 Z^{12}$$
$$10^{14} Z^{11} + 10^{16} Z^{10} + 10^{17} Z^9 + 10^{17} Z^8 + 10^{18} Z^7 + 10^{19} Z^6 + 10^{19} Z^5 + 10^{19} Z^4 +$$
$$10^{20} Z^3 + 10^{20} Z^2 + 10^{19} Z + 10^{19}$$

O máximo valor de Z_r ocorre quando $r = 16$ e $k = 15$, sendo a ordem de magnitude de $a_{15} \cong 10^{10}$

$$\delta Z_{16} \approx 2^{-t} \times (3{,}7 \times 10^{14})$$

Para $t = 48$ detivemos $\delta Z_r \approx 1{,}3$ o que é uma grande pertubação.

Ilustrações

1ª) Veja exemplo i) de III.2 e ii) de III.2.

2ª) $f(Z) = (Z-K-1)(Z-K-2)....(Z-K-20)$

$$\delta Z_r = \pm \frac{(k+r)^5}{(20-r)!(r-1)!} \cdot \delta a_s \qquad \text{onde} \qquad \begin{matrix} k = \pm 1, \pm 2, \pm 3, \ldots \\ k \text{ inteiro} \end{matrix}$$

Temos o pior caso quando $k = 20$

$$\frac{\delta Z_{15}}{\delta a_{19}} = 0{,}21 \times 10^7 \quad \text{(mal condicionado)}$$

Para $k = -20$, temos: $\dfrac{\delta Z_{18}}{\delta a_{19}} = 0{,}20 \times 10^3$ (melhor condicionado)

3º) $f(Z) = (Z - 2^{-1})(Z-2^{-2})....(Z-2^{-20}) = Z^{20} + Z^{19} + 2^{-1}Z^{18} + ... + Z^{-184}Z + 2^{-209}$

Os coeficientes deste polinomio variam muito em grandeza. Aplicando (2.8) obtemos

$$\delta Z_r \approx -\delta a_k \frac{2^{k.r}}{P.Q}$$

onde $P = (2^{-r}-2^{-1})(2^{-r}-2^{-2})\ldots(2^{-r}-2^{-r+1})$

$$Q = (2^{-r}-2^{-r-1})(2^{-r}-2^{-r-2})\ldots(2^{-r}-2^{-20})$$

Efetuando os cálculos (pág. 44-Wilkinson) temos

$$\left|\frac{\delta Z_r}{Z_r}\right| < 64 \cdot \left|\frac{\delta a_k}{a_k}\right|$$

Concluimos que pequenas mudanças relativas em a_k não nos levam a grandes mudanças em Z_r.

No caso $(Z-2^{-1})(Z-2^{-2})\ldots(Z-2^{-20})-2^{-31}Z^{19}$, calculando os zeros com 9 decimais somente 3 deles são mudados, em 1 ou 2 dígitos.

Se em lugar de considerarmos mudanças relativas considerarmos mudanças absolutas de 2^{-t} nos coeficientes, alguns zeros são extremamente sensíveis.

Para exemplicar, consideremos o caso em que substituirmos o último coeficiente do polinômio acima que é 2^{-210} por

$2^{-210} + 2^{-48}$, obteremos então:

$Z_1^1.Z_2^1\ldots Z_{20}^1 = 2^{-210} + 2^{-48} = Z_1.Z_2\ldots Z_{20}(1 + 2^{162})$ e pelo menos um desses zeros satisfaz a condição:

$$\left|\frac{Z'_r}{Z_r}\right| \geq \sqrt[20]{1+2^{162}} \geq 2^8$$ O problema é mal condicionado

4º) $z^{20}-1$

Os zeros são da forma $e^{\frac{-2i\ r}{20}}$ $(r = 0,\ldots,19)$ e estão uniformemente distribuídos sobre o círculo unitário. Aplicando (2-8) chegamos a

$$|\delta z_r| \approx \frac{|\delta a_k|}{20}$$

Verificamos que o problema é bem condicionado.

III.2.3. Conclusões sobre condicionamento de polinômios

Com os exemplos acima, acentuamos bastante a variação na condição do problema, de calcular os zeros de um polinômio. O mau condicionamento pode ocorrer mesmo para polinômios que tenham zeros que não pareçam extremamente próximos, e bom condicionamento pode ocorrer com polinômios com zeros extremamente próximos, contrariando a regra geral.

O terceiro exemplo mostra que a razão entre 2 zeros vizinhos é que é o fator decisivo, e não a distancia absoluta entre eles. Essa razão estando próxima de um sempre indica mau condicionamento.

III.2.3.1. O significado do condicionamento de zeros de polinômio

O que mais frequentemente ocorre é a resolução de problemas reduzirem-se ao cálculo de zeros de polinômio, não sendo porém os coeficientes do polinômio os dados iniciais.

Como exemplo, consideremos a equação $A\dot{X} = BX$, isto é, a solução de um conjunto de equações diferenciais simultânea de 1º ordem, onde A e B são matrizes (nxn). A solução desse problema é da forma $x = Q.e^{\lambda t}$, onde λ são as raízes de $\det(A\lambda-B) = 0$. Pode ocorrer que os zeros dessa expressão sejam bastante insensíveis às mudanças de A e B, e contudo, os zeros do polinômio correspondente sejam sensíveis as mudanças de seus coeficientes.

III.3. DETERMINAÇÃO DOS ZEROS

Até agora estudamos o efeito de erros nos coeficientes do polinômio, problema de condicionamento. Agora consideramos o efeito dos erros de arredondamento durante o processo de calcularmos os zeros. Não vamos considerar todos os processos para computar os zeros e portanto nos concentramos nos métodos iterativos.

III.3.1. Método de Newton

A limitação de exatidão que pode ser atingida com a computação sob uma dada precisão é maior para alguns dos métodos iterativos nos quais a avaliação de um polinômio para uma sequencia de aproximação de um zero, definem uma regra dominante. Um caso típico é o processo de Newton:

Para uma aproximação inicial $Z_k + h_0$ de Z_k, a sequencia das aproximações $Z_k + h_r$ é determinada pela relação

$$Z_k + h_{r+1} = (Z_k + h_r) - \frac{f(Z_k+h_r)}{f'(Z_k+h_r)} = \frac{Z_k + \left|\frac{1}{2} h_r^2 f''(Z_k)+\ldots\right|}{|f'(Z_k)+\ldots.|} ,$$

onde são omitidos termos no numerador e denominador que envolvem potências da mais alta ordem de h_r. Se Z_k é um zero simples então $f'(Z_k) \neq 0$, então temos

$$h_{r+1} \sim \frac{f''(Z_k)}{2f'(Z_k)} \cdot h_r^2 . (h_r \to 0).$$

Observe que o coeficiente de h_r^2 pode ser muito grande, se Z_k é um zero mal condicionado, e h_r pode então, ser muito pequeno antes da relação acima.

Consideremos agora o erro de arredondamento na determinação do zero Z_k. Admitindo que partindo de um valor inicial para o qual o processo de NEWTON converge para Z_k, chega-se finalmente que para todos os valores de Z que foram usados, o erro relativo de $f'(Z)$ é bem menor que o erro relativo de $f(Z)$, pois $f(Z) \to 0$.

Usaremos $Z_k + \bar{h}_r$ para denotar sucessivas aproximações computadas e $Z_k + h_{r+1}$ para denotar as aproximações que poderia ser obtida usando o método de Newton exatamente com $Z_k + \bar{h}_r$. Chamando os valores computados $f(Z_k + \bar{h}_r)$ e $f'(Z_k + \bar{h}_r)$ de $\bar{f}_r$ e $\bar{f}'_r$ e os valores reais $f(Z_k+\bar{h}_r)$ e $f'(Z_k+\bar{h}_r)$ de f_r e f'_r, levando em $h_{r+1} \sim \frac{f''(Z_k)}{2f'(Z_k)} . h_r$ com $h_r \neq 0$. Para valores suficientemente pequenos, temos de $\bar{h}_r$ $|h_{r+1}| < \bar{A}h_r^2$.

Para alcançar a precisão limitada por Z_k e $\bar{h}_r$, obtem-se uma sequencia de aproximações $Z_k + \bar{h}_m$ com a mesma precisão. As aproximações podem de fato serem representadas por $Z_k + C.h_r$.

Para polinômios mal condicionados de grau pequeno, é comum o valor computado do polinômio ser um zero no intervalo em que está contido o zero.

Exemplo, o polinômio $Z^2-2,0288888.Z + 1,0287690$ tem zeros $Z_1 = 1,0325673...$ e $Z_2 = 0,99632146$. Computado em ponto flutuante com 8 dígitos decimais, obtem-se Z no intervalo $1,0325660 \leq Z \leq 1,0325687$, enquanto que para o polinômio Z^2-2Z+1 o zero computado estará no intervalo $0,9999293 \leq Z \leq 1,0000707$.

Quando se inicia com uma aproximação para um zero, que é consideravelmente mais precisa do que a limitação de precisão, uma simples iteração poderá danificar aquilo que seria uma boa aproximação e produzir um erro típico da limitação da precisão. Isso não será verdade se o polinômio tem coeficientes que tem representação exata na máquina e um zero que requeira poucos dígitos para sua representação. Se este zero exato é usado como uma aproximação, então, geralmente nenhum erro de arredondamento será produzido durante a avaliação do polinômio.

Para determinação do erro na fórmula de NEWTON.

$x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)})}{f'(x^{(n)})}$ fazendo o erro no passo n por e_n então:

$$e_n = x - x^{(n)} \qquad e$$

$$e_{n+1} = x - x^{(n+1)}$$

subtraindo na formula de Newton na linha de cima ambos o termos de x.

$$x - x^{(n+1)} = x - x^{(n)} + \frac{f(x^{(n)})}{f'(x^{(n)})} \qquad \text{ou em termos de erro}$$

$$e_{n+1} = e_n + \frac{f(x-e_n)}{f'(x-e_n)}$$

expandindo na série de TAYLOR:

$$e_{n+1} = e_n + \frac{f(x)-e_n f'(x)+(e^2_{n/2}).f''(x)+ \ldots\ldots}{f'(x)-e_n f''(x)+(e^2_{n/2})f'''(x)+ \ldots.}$$

como $f(x) = 0$ e dividindo pelo denominador

$$e_{n+1} = e_n - e_n - \frac{e_n^2 f''(x)}{2f'(x)} + \text{o termo de maior ordem}$$

ou $$e_{n+1} \approx - \frac{e_n^2 f''(x)}{2f'(x)}$$

A equação acima determina que o erro absoluto até o passo n+1 é proporcional ao quadrado do erro absoluto até o passo n. Assim se tem uma resposta correta para uma casa decimal colocada no passo um, teria uma precisão para 2 casos decimais no próximo passo, quatro para o seguinte e assim sucessivamente.

III.3.2. Problema da Deflação

Em métodos iterativos, o valor inicial usado é mais ou menos arbitrário e, depois de algumas iterações um valor próximo dos zeros é encontrado. Quando um zero aproximado α é determinado, o polinômio é dividido por $(Z-\alpha)$ e a iteração continua com o polinômio quociente. Procedendo dessa maneira, todos os ZEROS são finalmente determinados e o perigo de convergir duas vezes para o mesmo zero é evitado. Entretanto existe o perigo de que os zeros do polinômio quociente passam divergir gradualmente daquele do polinômio original, determinando os zeros encontrados com muita imprecisão.

Por exemplo um polinômio do 3º grau com zeros 1,1, e 2. Usando t-dígitos aritméticos, a convergência para Z=1 determinaria quando (Z-1) for da ordem de $z^{-t/2}$ ($Z_0 = 1 + \delta.2^{-t/2}$; $Z_0 - 1 = \delta.2^{-t/2}$ Z_0-1 é da ordem $2^{-t/2}$. Se tal aproximação for aceita então todos os coeficientes do polinômio deflacionado serão diferentes do polinômio deflacionado verdadeiro, $(Z-1)(Z-2)$ até $\frac{t}{2}$ algarismos. Para Z=2 é um zero bem condicionado.

Na iteração do polinômio original, para uma dada precisão de computação a eficiência de determinação do zero é limitada pela sua condição de condicionamento. A acumulação de erros de arredondamento atinge pequena parte na limitação de eficiência. Um zero mal condicionado será determinado ineficientemente mesmo quando se faz a iteração com o polinômio original.

Exemplos de deflação

Considerando primeiro o polinômio

$x^4 - 6,7980x + 2,9948x^3 - 0,043686x^2 + 0,000089248$ cujos zeros são: 0,0024532....; 0,012576....; 0,45732....; 6,3256..... Esse polinômio tem zeros bem condicionados com respeito a pequenas trocas relativas nos coeficientes, mas não a respeito de pequenas trocas absolutas.

Aceitando-se o zero 0,0024532 e deflacionando usando a aritmética do ponto flutuante com 5 dígitos decimais escreve-se:

(Método Horner)

1	-6,7980000	2,9948	-0,043686	0,000089248
	0,0024532	-0,0166707206	+0,0073058	-0,0000892
1	-6,7955	2,9781	-0,03638	0,0000000=0
	(-6,7955468)	(2,9781293)	(0,0363802)	(0,000000048)

A computação foi executada com o polinomio pertubado $x^4 - 6,7979532x^3 + 2,9947707206x^2 - 0,0436858749x + 0,000089247416$. Mas como temos 5 dígitos decimais então entram para as contas o polinômio $x^4 - 6,7980x^3 + 2,9948x^2 - 0,043686x + 0,000089248$.

Entretanto este polinômio dividido por (x-0,0024532) tem resto zero e dá exatamente o quociente polinomial computado. Para o caso do zero 6,3256 ser aceito primeiro, a deflação procede assim:

1	-6,7980	2,9948	0,043686	0,000089248
ñ arred ↗	6,3256	2,98821344	0,04174896	0,122526872
arred.	6,3256	2,9882	0,041749	0,012252
1	-0,4724	0,00660	0,001937	

Cuja computação foi feita com o polinômio:

$x^4 - 6,7980x^3 + 2,994831344x^2 - 0,04368596x + 0,0122526872$

e este polinômio dá exatamente o quociente polinomial computado.

O polinômio pertubado está diferente do original em seu termo constante. Esse erro reside no fato de que os valores intermediários diferem do zero por uma quantidade que é muito maior que o termo constante, a dispeito que o zero aceito, está correto para 5 algarismos.

III.3.3. Refinamento usando o polinômio original

Quando todos os zeros foram obtidos por deflação podemos usar os valores computados como aproximações iniciais para a iteração com o polinômio original, obtendo-se o número de dígitos corretos que desejamos. Se os zeros foram encontrados estritamente, em ordem crescente de valores absolutos e o polinômio era de um tipo em que nenhuma deterioração de condição ocorreu durante a deflação, nenhum melhoramento é acrescido pelo processo de refinamento.

Entretanto o refinamento no polinômio original é útil em todos os casos. É uma maneira de determinar que o processo de Newton determina os zeros utilizando ordem crescente da menor à maior raiz, e o refinamento dá alguma proteção contra a aceitação de falsos valores resultantes de encontrar zeros com ordens erradas. O refinamento dá também algum melhoramento no caso em que a precisão cai consideravelmente abaixo dos limites de precisão, em conseqüência da deterioração da condição do condicionamento.

III.3.4. Outros métodos iterativos

Anteriormente concentramos a discussão do erro no método iterativo de Newton. Quase todos os resultados obtidos se aplicam aos outros métodos iterativos também. Dentre esses outros encontramos 2 com convergência cúbica cuja fórmula do zero é:

$$Z_{k+1} = Z_k - \frac{f(Z_k)}{f'(Z_k)} - \frac{\{f(Z_k)\}^2.\{f''(Z_k)\}}{2.\{f'(Z_k)\}^3}$$

e o segundo devido a Laguerre:

$$Z_{k+1} = Z_k - \frac{nf(Z_k)}{f'(Z_k) \pm \{H(Z_k)\}^{1/2}}$$

onde $H(Z_k) = (n-1)^2.\{f'(Z_k)\}^2 - n(n-1).f(Z_k).f''(Z_k)$. n, grau do polinômio

A limitação fundamental na precisão que se consegue com este métodos é determinada pela nossa habilidade de avaliar com precisão $1(Z_k)$ nas vizinhanças do zero. A precisão é quase a mesma daquela obtida por Newton. O método de Laguerre apresenta vantagem para o cálculo de zeros reais, porque partindo de $Z_0 = a$ e aplicando sucessivamente a fórmula acima, nos obtemos 2 sequências convergentes, para o próximo zero maior do que a, e para o próximo zero menor do que a. Devemos controlar a ordem que os zeros vão ocorrendo e avaliá-los em conexão com o processo de deflação.

O método de Bairstow pode também ser analisado por técnicas análogas. O objetivo desse método é o cálculo de fatores quadráticos reais $Z^2 - pZ - q$ de polinômios com coeficientes reais:

$$P_n(Z) = (Z^2 - pZ - q).(b_0 Z^{n-2} + \ldots + b_{n-2}) + R(pq).(Z-p) + S(p,q)$$

O polinômio $P_n(Z)$ tem o fator $(Z^2 - pZ - q)$ $\Longleftrightarrow$ $\begin{cases} R(p,q)=0 \\ S(p,q)=0 \end{cases}$

ou

$$P_n(Z) = \sum_{j=0}^{n} a_j x^{n-j} \quad \text{vem que}$$

$$\begin{cases} a_{n-1} + pb_{n-2}(p,q) + q.b_{n-3}(p,q) = 0 \\ a_n + pb_{n-1}(p,q) + q.b_{n-2}(p,q) = 0 \end{cases}$$

supondo as raízes complexas: $x = \xi \pm \eta i$

o fator será

$$[x - (\xi + \eta_i)].[x-(\xi-\eta_i)] = x^2 - 2\xi x + (\xi^2 + \eta^2)$$

III.3.5. CONCLUSÕES

O propósito desse trabalho foi realizar um estudo sintetizado sobre erros de arredondamento em computação e seus efeitos no cálculo de zeros de polinômios nos computadores digitais. Estudamos o erro de arredondamento nas operações aritméticas fundamentais, com isso pretendemos que o leitor conclua que se formos implementar, por exemplo aritmética em ponto flututante num determinado computador, temos de ter um formato interno maior que o formato externo, visível ao usuário (arredondando somente no final para os t dígitos) para que, quando for avaliar uma função, raiz quadrada por exemplo, em que usamos o método de Newton , $x_{u+1} = \frac{1}{2}(x_n + \frac{a}{x_u})$ onde $\{x_{n+1}\} \to \sqrt{a}$, os erros de arredondamento não se acumulem de tal forma a que a precisão da máquina, t dígitos não prejudique o resultado final. Quando a máquina tem a aritmética por hardware, isso já é feito e quando é implementado por software, é mais problemático, mas tem de ser feito. Em geral, seno, cosseno são feitos por software em precisão dupla, internamente quando o usuário quer em simples.

Outra conclusão é que ao avaliarmos os zeros de um polinômio , a limitação de t dígitos de um computador, já limita a precisão antes de iniciarmos a computação de seus zeros, portanto, os erros de truncamento e ou arredondamento vão se acumulando desde o início, além dos erros normais (teóricos) dos métodos. Isso tem uma implicação muito grande nos polinômios mal condicionados, onde pequenas variações dos coeficientes causam grandes transtornos no resultado final.

Portanto a conclusão principal dessa primeira parte e dar ferramentas para uma análise do erro para estabelecer-se limites para o efeito acumulado dos erros de arredondamento nas soluções, considerando-se que, em geral, os erros de arrendodamento são efetivados, a cada etapa do processo de obtenção da solução.

Da 2ª parte, estudo das condições de um polinômio com respeito a computação de seus zeros, gostaríamos de fazer alguns comentários:

Primeiro sobre as aproximações iniciais. Os métodos de ordem de convergência maior que 1 geralmente exigem melhores aproximações iniciais. Uma boa estratégia pode ser usar um método lento e seguro, (bisseção, algoritmo QD) para obter aproximações iniciais e depois usar um método de convergência rápido para obter resultados mais precisos.

Segundo sobre os erros introduzidos pela divisão por fatores já calculados.

Procedimentos para evitá-los:

- Determinar as raízes em ordem crescente de valores absolutos (geralmente não é trivial). Uma maneira de fazer seria usar o método de Laguerre com $x_0 = 0$, é bastante provável obter a raíz de menor valor absoluto.

- Calcular todas as raízes, efetuando-se as divisões, e depois fazer novas iterações com o polinômio original (refinamento).

- O método de Newton-Maehly aplica o método de Newton com dupla correção ao quociente de $P_n(x)$ pelos fatores já calculados, sem efetuar a divisão.

$$Q(x) = \frac{P_n(x)}{(x-u_1)(x-u_2)\ldots(x-u_k)}$$ onde $u_1, u_2, \ldots, u_k$ são as raizes já calculadas.

O terceiro comentário seria sobre o condicionamento das raízes.

Trata-se de saber o quanto uma raiz é sensível a variações (erros) nos valores dos coeficientes do polinômio, ver parte III.2.1.

$$\delta Z_r \approx p_1 \, \delta a_k$$

Finalmente uma conclusão devido a Wilkinson (1963, pág. 76):

"os métodos iterativos com deflação, seguido de refinamento tem obtido sucesso, mas para o problema em geral não parece ser fácil obter um algoritmo que assegure a convergência a partir de aproximações grosseiras. Além disso, se usarmos deflação, há a necessidade de averiguar se os zeros estão sendo determinados dentro do esperado".

A resolução de equações polinomiais é um problema não trivial em computação. Encontrar um algoritmo infalível para qualquer tipo de coeficientes aleatórios é difícil. Mas ainda assim, segundo Jenkins e Traub (1974, pág.20) "sentimos que o problema do cálculo de zeros de polinômios chegou a um nível de maturidade, existindo um bom software disponível e os critérios para discriminar programas bons e ruins são razoavelmente conhecidos."

Como sugestão para continuação desse estudo sugiro a leitura de [6] e [7]

BIBLIOGRAFIA

[1] WILKINSON,J.H. - Rounding Erros in Algebraic Process Englewood Cliffs: Prentice-Hall Inc. 1963.

[2] JENKINS,M.A. e TRAUB J.F. - Principles for Testing Polinomial Zerofinding Programs. Department of Computer Science - Carnegie Mellon University - Pittsburgh 1974.

[3] ALBRECHT,P. - Análise Numérica - Um Curso Moderno Livros Técnicos e Científicos Editora S.A.,1973

[4] RALSTON,A. - A first course in Numerical Analysis New York: MacGraw Hill Co. 1965

[5] PENNINGTON,R.H.- Introductory Computer Methods and Numerical Analysis New York: Collier - Macmillan Student Edition,1966.

[6] DUANE, A. ADAMS - A Stopping Criterion for Polynomial Root Finding CACM - outubro 1967, volume 10, nº 10

[7] KUT, H. e CODY, W.J. - A Statistical Study of the Accuracy of Floating Point Number Systems ACM, APRIL, 1973, volume 16, nº 4

[8] VIGNES, J.e.La PORTE, M - Error Analysis in Computing. Information Processing 74 - North-Holland Publishing Company (1974).