Artigo Técnico

Revista de
Informática
Teórica e Aplicada

Um Algoritmo Algébrico para Isolar Zeros Polinomiais Complexos

Maria Angelica de Oliveira Camargo
Universidade Estadual de Londrina
Vilmar Trevisan
Dalcidio Moraes Claudio
Universidade Federal do Rio Grande do Sul

Abstract

O presente trabalho trata do problema de isolar zeros de polinômios complexos. Muitos algoritmos calculam zeros polinomiais, a partir de regiões iniciais disjuntas, cada uma contendo um único zero. Entretanto o problema de obter tais regiões ainda é alvo de estudo, uma vez que as soluções propostas ainda não são satisfatórias. A obtenção de regiões disjuntas, denominada de isolamento de raízes está diretamente relacionada com a contagem (enumeração) do número de raízes numa determinada região do plano complexo. Algoritmos para enumerar e isolar raízes de polinômios complexos são analisados, desenvolvidos e implementados. A proposta de uma modificação no método numérico de Wilf é realizada, na qual se usa basicamente sequências de Sturm e o princípio do argumento da análise complexa. Um enfoque algébrico é dado para o algoritmo, visando enumerar zeros de forma exata dentro de um retângulo. Diversas melhorias foram introduzidas, principalmente no tratamento da presença de zeros nas fronteiras de um retângulo alvo de pesquisa. O desempenho do algoritmo proposto é avaliado através de testes experimentais. A abrangência do algoritmo também é verificada, através da realização de testes com polinômios mal condicionados. Uma comparação deste algoritmo com um recente trabalho é também realizada, mostrando a adequação deles de acordo com o tipo de polinômio.

Palavras-chave: Zeros polinomiais, sequências de Sturm, algoritmos algébricos.

1. Introdução

Existem diversos métodos numéricos para calcular zeros de polinômios complexos, que podem ser eficientes, mas estão sujeitos aos erros inerentes de qualquer sistema numérico. Soluções alternativas têm sido propostas, como os métodos de inclusão [?]. Entretanto, a maioria dos métodos de aproximação de zeros polinomiais requer o conhecimento prévio de regiões isoladas para os zeros procurados. Isto significa que se necessita obter um conjunto de regiões disjuntas (retângulos por exemplo), cada um contendo uma raiz simples do polinômio em questão. Entretanto, poucos trabalhos preocupam-se com a obtenção destas regiões disjuntas, que é o processo de isolar as raízes. Este processo está diretamente relacionado com o problema de enumerar zeros polinomiais. A maioria dos algoritmos para isolar zeros polinomiais utiliza um processo para contar o número de raízes numa certa região, a qual deve ser recursivamente particionada até que todos os zeros estejam isolados. O resultado deste processo, denominado enumeração, deve ser essencialmente um número inteiro e exato. Não é admissível um valor aproximado.

Alguns trabalhos que tratam do assunto valem ser brevemente comentados: Pinkert [?] parece ter sido o pioneiro numa proposta de um algoritmo algébrico para isolar zeros polinomiais complexos. Seu algoritmo está baseado no Teorema de Routh, que determina o número de zeros existentes no semiplano superior. Através da construção de faixas horizontais e verticais, a determinação do número de zeros dentro de um retângulo pode ser feita, usando para isso sequências de Sturm, juntamente com transformações polinomiais apropriadas (como rotação e raiz quadrada). Outro método foi proposto por [?] , para isolar zeros num retângulo, usando o princípio do argumento e sequências de Sturm. Um problema típico que aparece com o uso deste procedimento de enumeração diz respeito à restrição do retângulo não conter zeros na fronteira. Se estamos pesquisando zeros polinomiais em subretângulos arbritrários, como detectar esta condição? E ainda, como proceder em caso afirmativo? A solução proposta por Wilf não é satisfatória, uma vez que trabalha sobre possíveis erros devido a utilização de uma aritmética

de ponto flutuante. Recentemente, Collins & Krandick [?] propuseram um algoritmo também baseado no princípio do argumento, porém a contagem dos zeros é feita através da contagem do número de troca de quadrantes que ocorrem, considerando um polinômio $p(z)$ e z percorrendo a fronteira de um retângulo no plano complexo. Este método evita a utilização de sequências de Sturm, porém requer um procedimento para isolar raí zes reais de polinômios transformados.

Neste trabalho, apresentamos um algoritmo baseado no método numérico de Wilf [?], porém usando uma abordagem algébrica. Além disto, diversas melhorias foram adicionadas ao algoritmo original. Por exemplo, o problema da presença de zeros na fronteira do retângulo é tratada de uma forma exata. Otimizações foram realizadas, para evitar buscas dispensáveis, assim como utilização de subalgoritmos apropriados para classes especiais de polinômios (com coeficientes reais, com raízes somente nos eixos, etc).

A implementação do algoritmo proposto foi realizada usando o sistema de computação algébrica Macsyma, e diversos testes foram feitos em estações de trabalho Sun.

O restante do texto está organizado da seguinte forma: Os fundamentos matemáticos necessários para o entendimento do algoritmo são apresentados na seção 2. Os detalhes de implementação do algoritmo, bem como estrutura de dados e otimizações são descritos na seção 3. A Seção 4 trata da avaliação de desempenho do algoritmo proposto, através de testes experimentais. As conclusões estão na seção 5

2. Fundamentos Matemáticos

Uma das formas clássicas de se contar zeros de uma função dentro de uma região fechada é através do princípio do Argumento. Dado um n'umero complexo $z = a + bi$, com $r = |z|$, tem-se que o argumento (o ângulo θ) é pode ser obtido pelas seguintes relações:

$$a = rcos\theta$$

$$b = rsen\theta$$

Seja $p(z)$ um polinômio da seguinte forma:

$$p(z) = a_n\ (z - z_1)...(z - z_p)...(z - z_n)$$

A função $argp(z)$ é dada por:

$$arga_n + \sum_{i=1} parg(z - z_i) + \sum_{i=(p+1)} narg(z - z_i)$$

Teorema 2..1 (Princípio do Argumento) *Seja R uma região delimitada por uma curva simples, fechada com fronteira δR, e $p(z)$ um polinômio diferente de zero sobre δR. Seja $\Delta_{\delta R} \arg p(z)$ a mudança na função contínua $\arg p(z)$, quando z percorre δR no sentido antihorário.*

$$N = \frac{1}{2\pi} \Delta_{\delta R} \arg p(z)$$

Veja [?].

O valor de N é um número inteiro e representa o número de voltas que a imagem de $p(z)$ dá ao redor da origem quando z percorre δ_R no sentido antihorário.

Para qualquer função argumento, $\tan \arg p(z) = \frac{Im\, p(z)}{Re\, p(z)}$, onde $Im\, p(z)$ é a parte imaginária e $Re\, p(z)$ é a parte real de $p(z)$. As mudanças em $\arg p(z)$ podem ser obtidas, contando os saltos na função $\frac{Imp}{Rep}$ quando z percorre δR. Quando $p(z)$ cruza o eixo imaginário no sentido antihorário, *tan arg* $p(z)$ salta de $+\infty$ para $-\infty$, e se este cruzamento for no sentido horário , o salto é de $-\infty$ para $+\infty$. A contagem destes saltos pode ser feita usando o conceito de índice de Cauchy.

Definição 2..1 (Indice de Cauchy) *Seja $\frac{p}{q}$ uma função real racional e $[\alpha, \beta]$ um intervalo real. O Indice de Cauchy, denotado por $I_\alpha^\beta \frac{p}{q}$ é dado pela diferença entre o número de pontos $\in [\alpha, \beta]$ onde $\frac{p}{q}$ salta de $-\infty$ para $+\infty$, e o número de pontos nos quais $\frac{p}{q}$ salta de $+\infty$ para $-\infty$.*

A ligação entre Indice de Cauchy e o Princípio do Argumento pode ser estabelecida pelo seguinte teorema:

Teorema 2..2 *Se $\delta R : z = z(x), \alpha \leq x \leq \beta$ é uma curva simples fechada e p é um polinômio definido sobre δR, então:*

$$\Delta_{\delta R} \arg p(z) = -\pi I_\alpha^\beta \frac{Im\, p}{Re\, p}$$

Veja [?].

Se δR é um polígono, cada lado j de δR pode ser mapeado no eixo x, através de uma transformação adequada t_j, tal que $p_j = p(t_j(z))$. Assim, pode-se escrever:

$$\Delta_{\delta R} \arg p(z) = -\pi \sum_{i=1}^{m} I_l^0 \frac{Im\, p_j}{Re\, p_j} \tag{1}$$

Onde m é o número de lados de δR e l é o comprimento do lado j.

Usando os teoremas ?? e ??, pode-se concluir que o número de zeros polinomiais dentro de um retângulo pode então ser obtido por:

$$N = -\frac{1}{2} \sum_{i=1}^{4} I_0^l \frac{Im\, p_j}{Re\, p_j} \tag{2}$$

O cálculo do índice de Cauchy pode ser efetuado usando sequências de Sturm.

Definição 2..2 (Sequência de Sturm) *Uma sequência $f_0, f_1, ..., f_m$ de polinômios reais forma uma sequência de Sturm para um intervalo real $[a, b]$ se satisfaz as seguintes condições:*

i) f_m tem sinal constante em $[a, b]$;

ii) Para cada k $(1 \leq k \leq m-1)$ e cada zero $x^* \in [a, b]$ de $f_k(x)$, tem-se $f_{k+1}(x^*), f_{k-1}(x^*) < 0$.

O teorema a seguir estabelece como uma sequência de Sturm pode ser gerada.

Teorema 2..3 *Dados dois polinômios reais p_0 e p_1 não nulos, e os polinômios p_2, p_3, ... p_m definidos pelo algoritmo euclidiano (cálculo do máximo divisor comum - m.d.c.)*

$$p_{k+1} : -resto(\frac{p_{k-1}}{p_k})$$

Os polinômios $f_k := p_k/p_m$, $(k = 0, 1, ..., m)$ formam uma sequência de Sturm para qualquer intervalo $[a, b]$ tal que $p_0(a)p_0(b) \neq 0$.

Veja [?].

O teorema a seguir mostra como sequências de Sturm e Indice de Cauchy estão relacionados.

Teorema 2..4 (Teorema de Sturm) *Seja* $f_0, f_1, \ldots, f_m(x)$ *uma sequência de Sturm para* $[a, b]$ *e seja* $V(x)$ *o número de variação de sinais ao longo desta sequência no ponto* x. *Então:*

$$I_a^b(\frac{f_1(x)}{f_0(x)}) = V(a) - V(b) \tag{3}$$

Veja [?].

Combinando as expressões ?? e ??, tem-se:

$$N = \frac{1}{2} \sum_{1}^{4} (V(l_j) - V(0)) \tag{4}$$

[?].

3. O desenvolvimento de um algoritmo exato para isolar zeros polinomiais

Baseado num algoritmo numérico de Wilf, [?], desenvolveu-se um algoritmo de abordagem algébrica. No processo de isolamento de raízes polinomiais, pode-se dizer que a principal tarefa é o processo de enumerar as raízes numa determinada região. O algoritmo na forma como é proposto possibilita que o resultado do processo de enumeração seja sempre um n'umero inteiro, obtido de forma exata.

Inicialmente descrevemos o algoritmo para contar zeros polinomiais num retângulo, a partir da teoria estabelecida previamente.

Dispondo de um retângulo inicial contendo todas as raízes do polinômio, procede-se da seguinte forma para cada lateral:

i) Obtem-se o polinômio transformado correspondente à lateral mapeada no eixo x com seu ponto inicial na origem (0, 0). Isto é feito pela expansão de P(z) no vértice inicial do segmento de reta em questão. Assim, tem-se um polinômio mapeado, que será denotado por $pm_j(z)$, onde $pm_j(z) = p(t_j(z))$, sendo t_j a transformação necessária para a lateral j. Considerando Q_1 o vértice inferior esquerdo do retângulo, tem-se:

$t_1 = Q_1 + z$, $t_2 = Q_2 + iz$, $t_3 = Q_3 - z$, $t_4 = Q_4 - iz$,

ii) Separa-se a parte real e parte imaginária de pm_j.

$pm_j = \alpha_j(z) + i\beta_j(z)$.

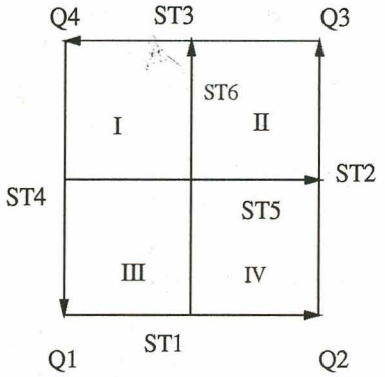

Figura 1: "Retângulo Particionado

iii) Gera-se a sequência de Sturm associada a lateral j, iniciando com $f_1 = \alpha_j$, $f_2 = \beta_j$ e $f_{k+1} = -resto(\frac{f_{k-1}}{f_k})$

iv) Calcula-se o número de variação de sinais para aquelas sequências no intervalo $[o, e]$, onde o é a origem deslocada e e é o ponto final do intervalo.

A região inicial usada neste trabalho é obtida através do cálculo da quota superior para o módulo de raízes definida em [?].

Dispondo-se das 4 sequências de Sturm para as 4 laterais, o seguinte procedimento é repetido até que se consiga todos retângulos isolados do tamanho desejado:

i) Obtém-se o centro do retângulo (zc)

ii) Obtém-se 2 novos polinômios transformados que equivalem ao mapeamento da linha horizontal central, e da linha vertical central no eixo x. Isto é obtido pelas expansões p(z+zc) e p(iz+zc) respectivamente. iii) Gera-se 2 novas sequências de Sturm a partir dos 2 novos polinômios. iv) Obtém-se os extremos de cada intervalo de acordo com a sequência que está sendo utilizada, podendo assim determinar o número de zeros no retângulo sendo pesquisado.

A figura 1 mostra o retângulo bisseccionado, com os 4 subretângulos (I, II, III e IV), e os e vértices (Q1, Q2, Q3 e Q4). Cada lado está associado com uma sequência de Sturm. As 4 sequências de Sturm básicas correspondem aos 4 lados do retângulo pesquisado e são denotadas respectivamente por ST1, ST2, ST3 e ST4. As duas sequências de Sturm adicionais são associadas com as linhas internas que compõem os 4 subretângulos. Estas sequências são denotadas por ST5 (para linha horizontal) e ST6 (para linha vertical).

De acordo com a numeração de subretângulos da figura, pode-se verificar que para cada subretângulo, ao ser percorrido no sentido antihorário, utiliza-se uma combinação de 4 sequências de Sturm, a partir de um conjunto de 6 sequências possíveis.

Assim, para I, usa-se ST5, ST6, ST3 e ST4;
para II, usa-se ST5, ST2, ST3 e ST6;
para III, usa-se ST1, ST6, ST5 e ST4 e
para IV, usa-se ST1, ST2, ST5 e ST6.

A determinação do número de zeros para cada subretângulo I, II, III e IV é feita pela utilização da fórmula **??**, deslocando-se devidamente as origens. Esta tarefa, apesar de simples, requer que se analise cuidadosamente cada situação possível da localização de um subretângulo em relação ao seu ascendente (retângulo pai). O maior problema encontrado foi como identificar quais as situações e como deslocar as origens de acordo com cada caso.

Um programa para obter o número de zeros num retângulo usando esse raciocínio foi implementado num ambiente de computação simbólica (Sistema Macsyma). Alguns problemas tiveram que ser contornados, especialmente na geração das sequências de Sturm. Conforme o teorema **??**, o algoritmo de Euclides pode ser usado para gerar uma sequência de Sturm. Entretanto, a aplicação direta deste algoritmo pode gerar expressões intermediárias muito grandes. (Veja por exemplo, [?], pag. 68).

Uma sequência de restos polinomiais negativos pode ser gerada usando o algoritmo de Euclides. Seja R_k o k-ésimo polinômio de uma Sequência de Sturm $R_k = \sum(a_i x^i)$.

$$f_{k+1} = R_k/cont(R_k)secont(R_k) < 0 - R_k/cont(R_k)casocontrário$$

onde cont(R_k)=mdc(a_i).

Esta sequência continua sendo uma sequência de Sturm.

Desta forma, o tamanho dos coeficientes gerados ficam menores, além de se ter a garantia que estes são sempre números inteiros. Como exemplo, considere uma sequência de Sturm, iniciando com os polinômios:

$$f_0 = 27z^4 - 719z^2 + 462$$

$$f_1 = 231z^3 - 961z$$

e os termos seguintes sendo gerados de forma direta, tem-se:

$$f_2 = \frac{46714z^2 - 35574}{77}$$

$$f_3 = \frac{18337280z}{23357}$$

$$f_4 = 462$$

Se usarmos a parte primitiva, a sequência fica então

$$f_0 = 27z^4 - 719z^2 + 462$$

$$f_1 = 231z^3 - 961z$$

$$f_2 = 23357z^2 - 17787$$

$$f_3 = z$$

$$f_4 = 1$$

Apesar do trabalho adicional requerido para se obter a parte primitiva, o custo total da geração da sequência fica reduzido, pois a aritmética com números inteiros requer menos trabalho computacional do que a aritmética com números racionais.

Um problema típico que aparece na utilização deste procedimento é a presença de zeros nas linhas que dividem um retângulo. Quando for necessário enumerar os zeros dos subretângulos , é preciso que não haja zeros nas linhas de fronteira que os compõem, para a aplicação da fórmula ?? . Como então identificar esta situação? E ainda, como proceder em caso afirmativo. A solução proposta por Wilf é estabelecida da seguinte forma: O lado esquerdo da fórmula ?? é um inteiro, logo o termo resultante da soma do lado direito deve ser um número par. Caso isto não se verifique, pode ter ocorrido perda de dígitos significativos nos cálculos. Em cada nível de pesquisa, os 4 subretângulos devem ser checados quanto à paridade. Se um ou mais subretângulo apresentar a soma com número ímpar, assume-se que houve perda de precisão, possivelmente pela presença de um zero de $p(z)$ sobre ou próximo das linhas de fronteira que compõem estes subretângulos. O centro do retângulo ascendente é então deslocado e examina-se novamente as 4 novas somas, repetindo este processo até 3 vezes, e em caso de insucesso, retorna-se o retângulo com o número de zeros obtidos.

É possível perceber que esta estratégia, apesar de interessante, pode gerar um certo conflito com zeros que realmente estão na fronteira com zeros que estão muito próximos de uma fronteira.

Assim, uma outra estratégia é adotada no algoritmo desenvolvido neste trabalho para solucionar este tipo de problema, garantindo a exatidão dos resultados obtidos. Procede-se da seguinte forma:

Seja $p(z)$ um polinômio de coeficientes complexos e livre de quadrados. Através da determinação de uma quota superior para o módulo das raí zes (por exemplo $b = 2 * \max_{1 \leq i \leq n} \frac{|a_{n-i}|}{q} a^{n^{\frac{1}{i}}}$), define-se um quadrado centrado na origem e com lateral de comprimento $2b$, onde b é o valor da quota superior calculada. Como esta quota gera valores irracionais, utilizou-se $\lfloor (b) \rfloor + 1$ para obter um valor inteiro e não muito maior que a quota real obtida. Desta forma, o quadrado resultante não contém zeros na fronteira.

Com relaç ão ao problema da presença ou não de zeros nas fronteiras dos retângulos pesquisados, o tratamento deve ser diferenciado para os diferentes casos. A fórmula ?? só pode ser usada quando não existem zeros na fronteira da região sendo pesquisada.

Para detectar a existência de zeros na fronteira, procede-se da seguinte forma:

Para cada lateral j, tem-se uma sequência de Sturm associada. O último termo desta sequência é , a menos do sinal, o máximo divisor comum (mdc) dos dois termos iniciais (f_0 e f_1), que denota-se por h_j. Se h_j for constante, então não existem zeros sobre a lateral j. Caso contrário, pode haver zeros sobre esta lateral, mas não necessariamente. Se o polinômio original $p(z)$ tiver algum zero sobre a lateral j, implica que o poli nômio transformado Pm_j tem zeros reais no intervalo $[o, e]$, onde o é a origem deslocada. Se Pm_j é complexo, significa que um zero de $p(z)$ deve ser um zero comum de f_0 e f_1 . Como neste caso h_j não é constante, se z_0 é um zero de h_j então é um zero comum de (α_j, β_j). Para verificar se h_j tem zeros reais, basta utilizar o teorema de Sturm [?], iniciando a sequência com $f_0 = h_j$ e $f_1 = h'_j$. Se h_j não contém zeros reais, o procedimento continua como no caso em que h_j é constante. Caso contrário, o lado associado é assinalado que contém zeros e o número de zeros desta lateral é armazenado (b_j).

Como é necessário uma região com fronteira sem zeros para aplicar a fórmula ??, obtém-se Pm_j/h_j, para eliminar estes zeros. Feito isto a sequência de Sturm para a lateral j deve iniciar com $f_0 = \alpha_j/h_j$ e $f_1 = \beta_j/h_j$.

Desta forma trabalha-se com cada lateral do retângulo sendo pesquisado livre de zeros. A fórmula ?? fica então modificada para

$$N = \frac{1}{2}\left(\sum_{j=1}^{4}(V(l_j) - V(0)) - \sum_{j=1}^{4}(b_j)\right) \quad (5)$$

3.1 Estrutura de Dados

A estrutura básica usada são listas. A entrada do procedimento de busca é dada pelos seguintes ítens:

- O canto noroeste do retângulo R;
- O tamanho do lado de R;
- Uma lista com as seis sequências de Sturm referente a R, da qual se seleciona as 4 sequências necessárias de acordo com o retângulo sendo pesquisado. Para cada sequência de Sturm deve-se acrescentar: seu nível, seu ponto de origem e uma sublista para o caso de se ter zeros na fronteira. Neste caso, esta sublista armazena a sequência de Sturm associada à fronteira.

A saída do procedimento de busca é dado por uma lista de retângulos isolados e uma lista de intervalos isolados (para os zeros localizados sobre as fronteiras dos retângulos pesquisados).

O procedimento de busca é feito em profundidade. Cada retângulo com número de zeros $N > 1$ deve ser bisseccionado e o processo de enumeração deve ser repetido recursivamente para cada subretângulo. Os subretângulos sem zeros são desprezados e aqueles com um único zero são armazenados numa lista de retângulos isolados.

3.2 Otimizações adicionais

Algumas modificações básicas foram introduzidas visando minimizar o número de regiões pesquisadas. Além disto, sugere-se uma verificação prévia de algumas propriedades algébricas sobre as raízes do polinômio alvo, visando aplicar algoritmos especializados para certas classes especiais de polinômios.

No algoritmo proposto, para cada retângulo bisseccionado, quando seu número total de zeros for encontrado, a pesquisa é encerrada (naquele nível).

Por exemplo, se um quadrado tem 4 zeros (2 no primeiro subquadrado e 2 no segundo subquadrado) o procedimento pesquisa somente os 2 primeiros quadrados, ao invés de todos eles.

Para cada nível da pesquisa, o último retângulo apresenta uma vantagem, pois é possível concluir previamente que o número total de zeros dentro dele é a diferença entre o total de zeros do retângulo ascendente e o total de zeros já encontrado. Este retângulo só precisa ser pesquisado se tiver um total de zeros maior que 1 no seu interior.

Quando o polinômio tem coeficientes reais, o algoritmo pode ser simplificado, observando-se diversas propriedades sobre esta classe especial de polinômios, conforme pode ser visto adiante.

- Inicialmente é possível verificar se todos os zeros de um polinômio são reais. Para isto, gera-se uma sequência de Sturm iniciando com $p(z)$ e $p'(z)$. De acordo com [?], se todos os coeficientes da sequência de um polinômio mônico forem positivos, então todos os zeros são reais. Neste caso, basta aplicar o algoritmo de Sturm para isolar os zeros reais.

- Outra propriedade interessante que pode ser verificada é a estabilidade de um polinômio: Um polinômio é estável (também chamado polinômio de Hurwitz) quando todos os zeros têm a parte real negativa. Um algoritmo eficiente para verificar tal condição é apresentado em [?]. Para polinômios estáveis basta pesquisar o lado direito do eixo y, reduzindo assim a região a ser pesquisada.

- Se o polinômio com coeficientes reais tem raízes reais complexas, estas ocorrem em pares conjugados. Então a pesquisa só precisa ser realizada na parte superior do eixo x.

- Para identificar a existência de zeros nos eixos, verifica-se $V(-\infty) - V(+\infty)$ $V(\infty)$ = número de variação de sinal entre os coeficientes de mais alta potência da sequência de Sturm gerada. $V(-\infty)$= número de variação de sinal entre os coeficientes de mais alta potência, trocando-se os sinais destes coeficientes de grau ímpar [?].

  Desta forma, identifica-se o número de zeros no eixo x, evitando-se o uso de avaliações, reduzindo assim o custo computacional. Processo análogo pode ser feito para o eixo y, bastando usar $p(iz)$.

Para qualquer tipo de coeficiente polinomial (real ou complexo), é possível verificar previamente a existência de zeros sobre os eixos x e y. Entretanto, para este caso não foi comprovado se existe uma boa relação de custo e benefício. Inicialmente, obtém-se $p(z) = p(x+yi) = Pr(x+yi)+iPc(x+yi)$, onde pr é a parte real e Pc é a parte imaginária de $p(x+yi)$.

Para verificar a existência de zeros sobre o eixo x, é necessário obter $Pr(x)$ e $Pc(x)$, pois neste caso $y = 0$. Os zeros de $p(z)$ sobre o eixo x são os zeros comuns de $Pr(x)$ e $Pc(x)$. Usando um algoritmo para isolar raízes reais, não é necessário obter $Pr(x)$ e $Pc(x)$, uma vez que $Pr(x)$ é o polinômio original $p(z)$ e $Pc(X)$ é identicamente nulo.

Da mesma forma, os zeros sobre o eixo y podem ser identificados, obtendo-se $Pr(iy)$ e $Pc(iy)$, pois neste caso, $x = 0$. Os zeros de $p(z)$ sobre o eixo y são os zeros comuns de $Pr(iy)$ e $Pc(iy)$.

Quando se aplica um algoritmo para isolar raízes reais, pode ocorrer que alguns intervalos resultantes sejam na verdade um ponto (no caso uma raiz de $p(z)$, que denota-se por r). Neste caso, é possível deflacionar seguramente o polinômio, obtendo-se $p/(z-r)$ de forma exata e então aplicar o algoritmo genérico de busca a este novo polinômio. Os intervalos não pontuais, mas que isolam uma raiz, fornecem subsídio que é usado na pesquisa de um retângulo, evitando futuras avaliações para contar o número de zeros sobre estas fronteiras.

4. Testes experimentais

Os algoritmos aqui analisados foram implementados no sistema de Computação Algébrica Macsyma, num ambiente Unix, em estações de trabalho Sun. Os programas foram escritos usando a linguagem do Macsyma, que por sua vez roda sobre a linguagem Lisp e permite que esta seja utilizada dentro dos programas escritos em Macsyma. Os programas não foram compilados, pois seria necessário que toda a codificação fosse escrita em Lisp. Isto provavelmente justifica alguns tempos de execução elevados que foram observados nos testes. Como estas medidas foram usadas para efeitos comparativos, este é um fator que pode ser desconsiderado. Entretanto, vale lembrar que futuramente alterações devem ser realizadas para melhorar o desempenho dos algoritmos desenvolvidos.

4.1 Seleção dos polinômios para teste

Os polinômios utilizados para testar a funcionalidade dos algoritmos foram:

- um conjunto de polinômios com coeficientes gerados aleatoriamente
- uma coletânea de polinômios usados em trabalhos realizados nesta área de pesquisa. (Consulte [?], [?],[?], [?] [?], [?]) e outros.
- Polinômios construídos especificamente para testes.

Os polinômios que apresentavam seus coeficientes em números de ponto flutuante foram substituídos pela sua representação racional para atender os requisitos dos algoritmos testados. Além disto, os polinômios com coeficientes não inteiros foram adequadamente transformados para satisfazer tal condição

Cabe lembrar que esta não é uma limitação para aplicação dos algoritmos, mas sim uma forma conveniente de colocar o problema a ser resolvido. Conforme já citado anteriormente, a aritmética exata realizada sobre números racionais requer maior esforço computacional do que a aritmética exata realizada sobre números inteiros. Ainda, os sistemas de computação simbólica dispõem de uma série de algoritmos de apoio eficientemente implementados sobre o domínio dos números inteiros (Z) ou dos inteiros Gaussianos, o que facilita a implementação de novos algoritmos.

Para se analisar o tempo médio gasto pelos algoritmos, (de acordo com a variação do grau), utilizou-se somente os polinômios gerados aleatoriamente, uma vez que estes apresentam comportamento similares.

O grau dos polinômios testados variou entre 2 e 30. Para os polinômios gerados aleatoriamente utilizou-se coeficientes com 10 dígitos binários. Foram considerados separadamente polinômios com coeficientes complexos e reais. Para obtenção dos tempos de execução, utilizou-se uma função interna do Macsyma, e foram realizadas em torno de 15 execuções para cada caso, para obter-se um tempo médio. Os valores expressos são sempre em segundos.

Com respeito a polinômios com raízes múltiplas, foram realizados alguns testes, mas não para efeito de verificação de desempenho, mas sim para verificação de funcionalidade. O problema das raízes múltiplas é resolvido por se realizar previamente uma fatoração livre de quadrados, para o qual já existem algoritmos eficientes.

4.2 Desempenho dos algoritmos testados

Tendo em vista que um recente algoritmo algébrico para isolar zeros polinomiais foi proposto por Collins-Krandick [?], o algoritmo desenvolvido neste trabalho é comparado com o de Collins-Krandick.

Considerando-se que o maior tempo dispendido pelos algoritmos encontra-se na enumeração dos zeros polinomiais, os testes realizados comparam o desempenho dos algoritmos nesta fase.

Como o algoritmo aqui desenvolvido está baseado no uso de sequências de Sturm, os tempos observados com relação a ele serão referenciados abreviadamente como t.Sturm.

O algoritmo de Collins-Krandick, baseado na troca de quadrantes terá seus tempos observados referenciados por t.Collins. A tabela 1 mostra os tempos observados na fase de enumeração para ambos os algoritmos.

No processo de enumeração usando troca de quadrantes (Collins-Krandick), após o isolamento das raízes reais dos polinômios especializados, há a necessidade de se refinar (bisseccionar) os intervalos isolados de polinômios distintos que não são disjuntos. Tendo em vista que não se tem informação sobre a relação das raízes reais dos polinômios especializados e as raízes do polinômio original, foram observados para o conjunto de polinômios testados, o número de bissecções necessárias após isolar as raí zes reais dos polinômios especializados no algoritmo de Collins-Krandick.

Apesar do processo de isolar as raízes ser diferente para os 2 algoritmos analisados, observa-se que as seguintes operações são necessárias a cada nível de particionamento do retângulo pesquisado:

. Para o algoritmo proposto:

Fazer 2 transformações para mapear as laterais para o eixo x. Gerar 2 sequências de Sturm, avaliar sequências de Sturm nos diferentes pontos necessários para enumerar os zeros dentro dos subretângulos.

. Para o algoritmo de Collins-Krandick:

Uma especialização = 2 avaliações. Isolar raízes de 2 polinômios especializados, refinar os intervalos isolados de diferentes polinômios que não são disjuntos.

No algoritmo proposto, tem-se que a maior parte do tempo é gasta na geração da sequência de Sturm, enquanto que no algoritmo de Collins-Krandick, o maior gasto de tempo fica por conta do processo de isolar as raízes reais. Os resultados obtidos, comparando-se os dois algoritmos mostram-

grau	Coefic. Reais			Coefic. Complexos		
	t.Sturm	t.Collins	dif. relat.	t.Sturm	t.Collins	dif. relat.
2	0,73	*		1,02	2,98	1,92
3	1,17	3,45	1,95	1,98	5,13	1,59
4	1,67	5,62	2,36	2,27	6,58	1,89
5	2,28	7,18	2,15	3,92	11,85	2,02
6	4,12	11,42	1,77	5,88	16,30	1,77
7	5,43	15,70	1,89	9,08	23,32	1,56
8	7,52	23,58	2,13	13,53	34,43	1,54
9	11,62	29,70	1,55	20,05	48,22	1,40
10	15,23	39,37	1,58	28,25	60,78	1,15
11	22,75	49,62	1,18	36,00	78,25	1,17
12	28,17	65,48	1,32	59,42	98,08	0,65
13	43,65	84,67	0,94	82,6	123,52	0,49
14	65,28	102,70	0,57	110,9	152,27	0,37
15	76,12	127,58	0,67	148,12	189,22	0,28
16	123,48	158,22	0,28	164,20	235,77	0,43
17	132,95	193,88	0,46	261,13	289,27	0,10
18	174,30	238,10	0,36	330,80	356,52	0,07
19	224,57	283,22	0,26	423,10	428,88	0,01
20	275,40	357,08	0,29	536,68	504,43	-0,06
21	352,75	404,88	0,15	658,97	607,33	-0,07
22	434,38	478,75	0,10	846,58	695,35	-0,18
23	547,85	568,73	0,04	1063,97	812,50	-0,23
25	773,50	780,75	-0,03	1257,65	939,20	-0,29
30	1804,15	1520,60	-0,16	-	-	-

labeltab-erc

se compatíveis com os observados na enumeração, indicando que para polinômios de grau abaixo de 20, é mais conveniente usar o algoritmo proposto, e para polinômios de grau maior, o de Collins-Krandick é mais apropriado.

No algoritmo de Collins-Krandick, as bissecções ocorrem de maneira alternada, ora horizontal, ora verticalmente. No algoritmo aqui desenvolvido, as bissecções na verdade dividem o retângulo em quatro, e os quatro subretângulos são analisados sempre na mesma ordem.

A questão de quantos níveis são necessários aprofundar-se para conseguir isolar as raízes é altamente dependente da distribuição destas. Entretanto, parece que o número de pesquisas necessárias tende a ser o mesmo para os dois algoritmos analisados.

Outro aspecto que foi analisado é a questão do comportamento do algoritmo considerando-se o tamanho dos coeficientes. Neste sentido, foram realizados testes com polinômios de grau fixo (10), variando-se o tamanho dos coeficientes gerados aleatoriamente (3, 6, 9, 12 e 15 dígitos decimais). Observou-se que para polinômios com coeficientes de até 10 dígitos decimais, o algoritmo proposto é mais conveniente, e para polinômios com coeficientes grandes, o de Collins-Krandick é mais adequado. Além disto, o algoritmo proposto é mais sensível ao tamanho dos coeficientes do polinômio do que o algoritmo de Collins-Krandick.

Outros polinômios foram testados, conforme descrito anteriormente. Os polinômios abaixo têm seus resultados na tabela 3. $p1 : 1000z^{10} - 2500z^9 - 460800z^8 - 9133400z^7 - 50761800z^6 - 88653100z^5 - 53510400z^4 - 37313000z^3 - 197170000z^2 - 364800000z - 198000000;$
$p2 : (z - (1 + \frac{9}{10}i)) * (z - (1 + \frac{99}{100}i)) * (z - (1 + \frac{999}{1000}i)) * (z - (1 + \frac{9999}{10000}i))$
$p3 : z^4 - (5 + 3i)z^3 + (6 + 7i)z^2 - (54 - 22i)z + (120 + 90i)$
$p4 : 470832z^3 - 665857(i + 1)z^2 - 1883328iz - 2663428(1 - i)$
$p5 : z^6 - 1,2z^5 + 0,04z^4 - 0,132z^3 - 0,7955z^2 - 1,43576z + 0,89496$
$p6 : z^9 - 1,1z^8 + 1,12z^7 - 1,232z^6 + 2,084z^5 - 2,2924z^4 + 1,075648z^3 - 1,1832128z^2 + 1.07910544z - 1.187015984$
$p7 : 13652098z^4 - (38613965 + 38613965i)z^3 + (2i)z^2 + (-154455860 + 154455860i)z + 218433576$

Outras classes de polinômios também foram testadas como os polinômios ciclotômicos que são definidos da seguinte forma:

Tabela 1: Tempos observados para para enumerar zeros dos polinômios

Polinômio	t.Sturm	t.Collins
p1	19,93	39,95
p2	2,47	6,67
p3	1,72	6,6
p4	1,94	4,12
p5	4,2	11,84
p6	22,5	32,85
p7	2,9	5,88

$$P_n(z) = \sum_{j=0}^{n-1}(z^j),\ \textit{para } n \geq 2 \textit{ e } n \textit{ primo}$$

cujas raízes são: $\alpha_k = e^{\frac{2\pi k_i}{n}}$.

No caso complexo, tem-se:

$$P^*(z) = P(z+i)$$

cujas raízes são: $\alpha_k^* = \alpha_k - i$

Pode-se observar o mesmo comportamento previamente descrito para os polinômios gerados aleatoriamente, ou seja para graus mais elevados (acima de 20), os tempos observados para o algoritmo de Collins-Krandick tendem a ser menor do que para o algoritmo proposto, usando sequências de Sturm.

5. Conclusões

Com os estudos desenvolvidos neste trabalho foi possível sugerir novos critérios para utilização de algoritmos apropriados para enumeração e isolamento de raízes polinomiais. Para a obtenção de resultados exatos (que são imprescindíveis nestas fases), optou-se por uma abordagem algébrica no desenvolvimento dos algoritmos.

Pode-se confirmar que a mera adaptação de algoritmos numéricos para uma versão algébrica não resolve por completo o problema, uma vez que os algoritmos resultantes tornam-se muito ineficientes.

O algoritmo para isolar raízes desenvolvido neste trabalho permitiu resolveu problemas identificados na aplicação do princípio do argumento, como por exemplo a presença de zeros na fronteira de um retângulo.

A verificação prévia de algumas propriedades algébricas dos polinômios permitiu a utilização de algoritmos especializados para casos específicos de polinômios, como por exemplo polinômios que só possuem raízes reais, ou cujas raízes estão sobre os eixos, etc.

Os testes experimentais realizados permitiram uma avaliação de desempenho do algoritmo aqui desenvolvido, mostrando que este é competitivo com o algoritmo recentemente proposto por Collins-Krandick.

De acordo com os resultados obtidos, observou-se que para graus relativamente baixos (em torno de 20) o algoritmo proposto tem um desempenho melhor.

Na fase de enumeração, o algoritmo de Collins mostrou-se mais eficiente a partir de grau 25 para polinômios de coeficientes reais e a partir de grau 20 para polinômios de coeficientes complexos. Já para o isolamento, analisando-se as tarefas necessárias, observou-se que até grau 20, o aumento de tempo em relação ao grau é similar para ambos os algoritmos. Para graus maiores, o aumento de tempo ocorre bem mais rápido para o algoritmo proposto, tornando inadequado o uso do algoritmo proposto para polinômios de graus elevados.

Já com relação ao tamanho dos coeficientes, observou-se que para coeficientes com tamanho menor do que dez dígitos decimais, o algoritmo proposto apresenta um desempenho melhor, invertendo-se a situação para números maiores. Observou-se também que o algoritmo proposto é mais sensível ao tamanho dos coeficientes do que o de Collins. (O aumento de tempo em relação ao tamanho dos coeficientes cresce bem mais rápido).

Referências

[1] BARBEAU, E. J. Polynomials. New York:Springer-Verlag, 1989. 441p. (Problem Books in Mathematics).

[2] CAMARGO-BRUNETTO, M. A. O. Algoritmos algébricos para enumerar e isolar zeros polinomiais complexos. Tese submetida para obtenção do grau de doutoramento em Ciência da Computação - Universidade Federal do Rio Grande do Sul, 130 p. 1994.

[3] CLAUDIO, D. M. e RUMP, S. M. Inclusion methods for real and complex functions in one variable. *Berichte des Forschungsschwerpunktes Informations- und Kommunikationstechnik*, Bericht 92.5, TUHH, Hamburg, 1992, 16 p.

[4] COLLINS, G. E. and KRANDICK, W. An Efficient Algorithm for Infallible Polynomial Complex Root Isolation. *Proceedings* . : ISSAC, USA, 1992. p. 189-194.

[5] COLLINS, G. E. Infallible Calculation of Polynomial Zeros to Specified Precision. G.E. Collins in: Mathematical Software edited by J. Rice, 1977.

[6] DAVENPORT, J. H.; SIRET, Y. ; Tournier, E. Computer Algebra: Systems and algorithms for algebraic computation. London, San Diego: Academic Press 1988.

[7] GATES, G. ; WANG, P. A LISP-based Ratfor Code Generator. In: MACSYMA USERS CONFERENCE, 1984. Proceedings..., 1984, p.319-329.

[8] GEORG, S. Two Methods for the verified inclusion of zeros of complex polynomials. Contributions to computer arithmetic and Self-Validating numerical methods. In: 12 IMACS WORLD CONGRESS ON SCIENTIFIC COMPUTATION, 1988, França. Proceedings..., Basel, J.C. Baltzer AG, 1990, 526p., p.229-244.

[9] GHADERPANAH, S. ; KLASA, S. Polynomial Scaling. SIAM J. NUMER. ANAL., v.27, n.1, p.117-135, 1990.

[10] HENRICI, P. Applied and computational complex analysis. New Jersey: John Willey & Sons, 1974, v.1.

[11] KNUTH, D. "Seminumerical Algorithms", in *The Art of Computer Programming*, Vol. 2, Addison-Wesley, 1969.

[12] MARDEN, M. The Geometry of the zeros of a polynomial in a complex variable. American Mathematical Society Mathematical Surveys III, 1949.

[13] NODA, Matu-Tarow Approximate GCD and its application to ill-conditioned algebraic equations. Journal of Computational and Applied Mathematics, Antwerp, Bélgica, v.38, p.335-351, 1991.

[14] PETKOVIC, M. S. ; HERZBERGER, J. Hybrid inclusion algorithms for polynomial multiple complex zeros in rectangular arithmetic. Applied Numerical Mathematics, v.7, p.241-262, 1991.

[15] PETKOVIC, M. S. A class of simultaneous methods for the zeros of analytic functions. Computers & Mathematics With Applications, Oxford, v.22, n.10, p.79-87, 1991.

[16] PINKERT, J. R. An Exact Method for finding the roots of a complex polynomial. Acm Transactions on Mathematical Software, New York, v.2, n.4, p.351-363, 1976.

[17] RUMP, S. M. Solving Algebraic Problems with High Accuracy. In: Kulisch, U.; Miranker, W. L. (eds). A new Approach to Scientific Computation (eds. U. Kulisch, W. L. Miranker). Academic Press, 1983. p.51-120.

[18] TREVISAN, V. Recognition of Hurwitz Polynomials. SIGSAM Bulletin, New York, v.24, n.4, p.26-32, 1990.

[19] WILF, H. S. A Global Bisection algorithm for computing the zeros of polynomials in the complex plane. Journal of the ACM, New York, v.25, n.3, p.415-420, 1978.