UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Algoritmos algébricos para enumerar e isolar zeros polinomiais complexos

por

Maria Angelica de Oliveira Camargo Brunetto

Tese submetida como requisito parcial
para a obtenção do grau de
Doutor em Ciência da Computação

Prof. Dalcídio Moraes Claudio
Orientador

Prof. Vilmar Trevisan
Co-orientador

Porto Alegre, julho de 1994.

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Camargo Brunetto, Maria Angelica de Oliveira

Algoritmos algébricos para enumerar e isolar zeros polinomiais complexos / Maria Angelica de Oliveira Camargo Brunetto.—Porto Alegre: CPGCC da UFRGS, 1994.
131 p.: il.

Tese - (Doutorado)—Universidade Federal do Rio Grande do Sul, Curso de Pós-Graduação em Ciência da Computação, Porto Alegre, 1994. Orientador: Claudio, Dalcídio Moraes; Co-orientador: Trevisan, Vilmar

Tese: Computação Simbólica
Zeros polinomiais, algoritmos algébricos, Sequências de Sturm

Ao meu esposo Fernando e as minhas filhas Claudia, Bruna e Gabriela, com amor e agradecimento.

AGRADECIMENTOS

Nesta longa aventura, foram muitas as pessoas que de uma forma ou de outra, contribuiram para que ela chegasse ao fim. (Com sucesso, espero).

Ao prof. Dr. Dalcídio Moraes Claudio, meu orientador pelo seu esforço dedicação, críticas e sugestões para conclusão deste trabalho. Também pela força naquelas horas difíceis que deu vontade de desistir.

Ao prof. Dr. Vilmar Trevisan, meu co-orientador pelo grande auxílio que me proporcionou acompanhando de perto todos os passos do meu trabalho, discutindo , questionando e sugerindo idéias.

A profa. Dra Laira V. Toscani, e ao prof. Dr. Valdir Roque, pelas sugestões dadas.

Aos meus colegas do pós, que compartilharam as ansiedades e alegrias de cada obstáculo avançado, em especial: Jussara, Bendedito, Flavinho, Tiaraju e Carmem e Wanderley.

Ao Grupo de Matemática Computacional, por assistirem aos seminários do trabalho que estava realizando, com críticas e sugestões, em especial a Beatriz.

À amiga Marcilia da UFPE, pela verdadeira amizade e pelo incentivo dado para que este trabalho fosse concluído.

Ao pessoal da biblioteca, pelo suporte oferecido.

Aos funcionários do laboratório de software, em especial Camilo, Mauro e Luis Otavio, pela paciência e dedicação em nos atender.

À família Parizzi, que me deu todo apoio e hospitalidade quando estava distante de minha família.

Ao prof. Ramon Poisl, pela confiança em mim depositada.

A professora Eliza Alves de Lima (NPD- Universidade Estadual de Londrina), pela amizade e pela coragem de se responsabilizar pelo meu retorno a UEL.

Ao pessoal da CPCD-CRH-UEL, em especial Jonas, pela dedicação em cuidar de todos assuntos pertinentes a minha licença.

Ao departamento de Matemática Aplicada da UEL, pelo apoio em conceder e renovar minha licença que possibilitou a conclusão deste trabalho.

À CAPES, pelo apoio financeiro em forma de bolsa que me foi concedido durante o período de estudo.

Ao meu esposo Fernando, pela paciência , compreensão e incentivo dado nas horas mais difíceis, e por me acompanhar nesta aventura. Também pela ajuda na confecção final desta tese.

Às minhas filhas (Claudia, Bruna e Gabriela), pela compreensão demonstrada e apoio na minha ausência.

Aos meus pais e sogros, pela ajuda inestimável no cuidado das crianças, por ocasião de férias e mudanças.

A Deus, que me deu forças para realizar este trabalho com sucesso.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

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 tanto nos aspectos teórico como prático, através da determinação da complexidade teórica e 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.

TITLE: ALGEBRAIC ALGORITHMS FOR ENUMERATE AND ISOLATE COMPLEX POLYNOMIAL ZEROS.

ABSTRACT

In this thesis, the problem of isolating polynomial complex zeros is treated. There are many algorithms to calculate polynomial zeros, having previously isolated regions, each containning only one zero. Despite of this, the problem of obtainning such regions is still unsatisfactory. This problem, called root isolation, requires number of root in a given region of the complex plane. Algorithms to enumerate and isolate complex polynomial roots are analised, developed and implemented. A modified Wilf method is given, in wich Sturm Sequences and the principle of argument is used. An algebraic approach is given, with the aim to enumerate zeros inside a rectangle in an exact way. Several improvements are introduced, mainly to treat zeros on the boundary of the rectangle. The performance of this new algorithm is evaluated theoretical as well as practice point of view, by means experimental tests. The robustness of the algorithm is verified by means of tests with ill-conditioned polynomials. The algorithm proposed is compared with a recent paper, presenting the performance of both, according different polynomial classes.

KEYWORDS:

Polynomial Zeros, Sturm Sequences, Algebraic Algorithms.

1 INTRODUÇÃO

1.1 Relevância do estudo

Dentro da área de computação científica, o uso de equações polinomiais é bastante amplo. A determinação de zeros polinomiais é de grande importância em tópicos como teoria de sistemas de controle (desenvolvidos para executar controle automático de operações diversas, como controle de pressão, temperatura, umidade, etc) [OGA 82], estabilidade de sistemas, circuitos não lineares, análise de funções de transferência, modelos matemáticos diversos, equações diferenciais, problemas de autovalores e outros.

Uma das grandes preocupações em relação à polinômios diz respeito a resolver problemas do tipo: determinar todas as raízes do polinômio (reais ou complexas), ou identificar o número de raízes em uma região do plano, isolar as raízes de um polinômio, ou ainda identificar condições de estabilidade de um polinômio.

Muitos estudos têm sido desenvolvidos nesta direção, com ênfases distintas. Uma vasta teoria matemática foi desenvolvida visando dar subsídios para resolver os problemas citados. Neste sentido, são de grande contribuição os trabalhos de [MAR 49] e [HEN 74].

A rápida evolução tecnológica fez com que surgissem computadores cada vez mais potentes, possibilitando que problemas, antes intratáveis, tornassem exequíveis, mesmo com antigos algoritmos. A disponibilidade dos recursos computacionais, fez com que surgissem novos "softwares" que se adequassem a capacidade de tais máquinas. Tal fato contribuiu de forma decisiva para o desenvolvimento de novos algoritmos para a resolução dos problemas em questão.

Entretanto, o uso de computadores trouxe novos problemas. A existência de um método matematicamente correto não implica na existência de um algoritmo eficiente. Uma das razões pode ser o esforço computacional demasiado para implementar um determinado método, tornando-o inviável na prática. Outra é a diferença que existe entre o Universo Matemático e o Computacional, que podem gerar resultados não confiáveis. Entre estas diferenças, pode-se citar a representação não contínua dos números reais nas máquinas, a não validade de propriedades básicas da aritmética matemática, como a associatividade aditiva, problemas com equivalência de expressões, problemas com arredondamentos em operações sobre entidades do tipo vetores e matrizes, etc.

Medidas de trabalho computacional necessário (teórico e prático) são imprescindíveis para utilização de novos algoritmos. O trabalho computacional teórico pode ser definido através do número de operações necessárias para se realizar determinado algoritmo. Este tipo de medida é importante no sentido de poder avaliar a viabilidade de se implementar o algoritmo nos computadores disponíveis. Por outro lado, sendo viável executar-se as computações envolvidas no algoritmo, sabe-se que eventualmente o trabalho necessário na prática pode ser diferente do que o estabelecido teoricamente. A questão do tempo consumido é bastante discutível, pois é uma característica que varia de acordo com a máquina que se utiliza, bem como as diferentes formas de se implementar um mesmo algoritmo. Cabe salientar entretanto, que os padrões de qualidade de tempo são dinâmicos: o que era inconcebível há anos atrás, hoje é perfeitamente viável, devido ao aumento de velocidade dos processadores e outros recursos adicionais.

Num contexto mais amplo, é inegável o crescente uso da computação científica em diversas áreas que até anos atrás pouco uso fazia desse valioso recurso. Áreas como Biologia, Química e outras fazem parte desta nova gama de usuários. Isso fez com que se aumentasse o nível de exigência com respeito à qualidade e a especificidade do "software" desenvolvido. Outra questão que deve ser considerada é o grande aumento das diversas formas alternativas pelas quais o problema pode ser

explorado: ambientes diversos para desenvolvimento de software científico, formas distintas de aritmética computacional, desenvolvimento de novas teorias, etc.

Com relação ao problema de aproximar zeros polinomiais, existe um grande número de métodos numéricos que tratam desta questão (Veja [CAM 90] e [HEN 74]), os quais podem ser eficientes, mas estão sujeitos aos erros inerentes de qualquer sistema numérico. Existem basicamente duas razões que justificam tamanha diversidade [PIN 76] : uma delas é a necessidade de se controlar os erros computacionais. Outra razão é a dificuldade de se tratar com raízes múltiplas ou raízes aglomeradas, que leva a uma constante busca de um algoritmo melhor. Certamente os erros computacionais variam conforme a equação e a aplicação. Se uma equação é mal condicionada e a aplicação requer resultados com alta exatidão, os erros devem ser controlados rigorosamente para não invalidar os resultados obtidos.

Muitos algoritmos propostos para aproximar zeros requerem condições de aplicabilidade que nem sempre são simples de se conseguir. Por exemplo: é necessário que se tenha regiões disjuntas, cada uma contendo uma única raiz; ou é necessário que todas as raízes sejam reais, ou ainda todas raízes sejam simples. A identificação prévia das características de um polinômio pode exigir um grande esforço computacional, além de que a obtenção dos requisitos necessários pode não ser trivialmente resolvidos. Uma grande questão está em se usar o algoritmo adequado para cada tipo de problema. É preciso identificar o "ponto forte" de um algoritmo para usá-lo adequadamente. Mais ainda, às vezes é preciso saber como colocar o problema numa forma mais adequada para aplicar determinados algoritmos que requerem algumas exigências.

Em contrapartida, poucos trabalhos se preocupam com o problema de isolar as raízes de um polinômio, que consiste em determinar um conjunto de regiões disjuntas, cada uma contendo um único zero do polinômio. Certamente existem algoritmos desenvolvidos para tal fim [HEN 74], porém alguns inconvenientes surgem quando estes algoritmos são transportados para um ambiente computacional. Isto acontece porque o processo de isolamento requer que se conheça o número de raízes

numa determinada região do plano. Para tal, é preciso que se tenha disponível um algoritmo para contar (enumerar) as raízes de um polinômio na região desejada. O resultado deste processo de enumeração deve ser necessariamente um número inteiro e exato, não podendo pois ser um valor aproximado que gere dúvida. Qualquer erro nesta fase pode fazer com que as regiões isoladas não sejam confiáveis.

Muitos algoritmos numéricos que utilizam teoremas clássicos da álgebra (como Sturm, Routh, Vincent) para enumerar zeros (num intervalo por exemplo), podem produzir resultados totalmente errados, devido às aproximações realizadas nas operações envolvidas (transformações polinomiais, avaliações, etc).

Observa-se pelos trabalhos desenvolvidos nesta área (e outros que fazem uso de computação científica), que o uso de formas alternativas e mais avançadas têm sido incentivadas com o objetivo de se reduzir a um nível mínimo o problema de erros causados pela não continuidade dos números, inerentes a qualquer computador. Soluções alternativas que estão sendo utilizadas para minimizar os erros podem ser através do uso de aritmética exata (computação simbólica), ou pelo uso de aritmética de alta exatidão (através da análise intervalar).

Nesta busca contínua por algoritmos que forneçam resultados mais próximos do real, um fator a ser considerado é o custo computacional. É sabido que o uso de aritmética exata (via computação simbólica) ou aritmética de alta exatidão (usando intervalos) têm em princípio um custo computacional maior que a forma convencional num ambiente de computação numérica. Assim, torna-se necessário que os algoritmos sejam desenvolvidos cuidadosamente, de forma a minimizar operações dispendiosas. Cabe lembrar que um grande auxílio no desenvolvimento de novas implementações é a disponibilidade de bibliotecas com rotinas básicas otimizadas nos sistemas de desenvolvimento de software científico. Além disto, o crescente desenvolvimento de ambientes de computação paralela motiva o desenvolvimento futuro de versões paralelas dos algoritmos aqui propostos.

Perante essa evolução, não se pode ignorar a necessidade de se buscar soluções mais adequadas e eficientes para problemas de grande aplicabilidade nas áreas científicas, como é o caso de determinação de zeros polinomiais. Parece pois perceptível que para se determinar raízes de polinômios, é necessário que problemas adjacentes sejam eficientemente resolvidos, como enumerar e isolar as raízes, obter as soluções triviais, de forma a simplificar o problema. Só assim, será possível a aplicação de um método eficiente para determinação das raízes procuradas.

1.2 Estado da arte

Com a evolução tecnológica crescente, acompanhada do desenvolvimento de novos recursos de software, surgiu a necessidade da criação de novos algoritmos para resolver problemas que anteriormente eram considerados intratáveis computacionalmente. A tendência cresce no sentido de se achar soluções para problemas mais abrangentes e complexos.

Basicamente o que se busca são alternativas que permitam obter resultados que melhor se aproximem de uma solução real. Isto leva ao aparecimento de formas diversas de contornar o problema de erro na obtenção de uma solução. Com o objetivo de melhor colocação do problema em relação ao estado da arte, alguns trabalhos que são de interesse na problemática dos zeros polinomiais serão brevemente comentados a seguir.

Num contexto de algoritmos para calcular zeros polinomiais, observa-se que existe um esforço de pesquisa contínuo no sentido de se obter aproximações melhores tanto quanto possível.

Diversos trabalhos tratam da determinação simultânea de raízes polinomiais. Formas alternativas para minimizar os erros das aproximações obtidas têm sido desenvolvidas. Como exemplo, pode-se citar algoritmos usando aritmética

intervalar complexa que objetivam fornecer uma aproximação com os limites de erro já obtidos no processo iterativo. Tais algoritmos caracterizam-se por produzir iterações intervalares, onde cada intervalo resultante é sempre menor que o da iteração anterior, garantindo que a aproximação final é consistente com o intervalo inicial utilizado. Estes algoritmos também são conhecidos como métodos de inclusão (por exemplo [FRA 89]. Um problema que eles podem apresentar é o alto custo computacional. Em [PET 91a], Petkovic propõe um algoritmo híbrido, onde se inicia o processo com o centro de uma região que possui um único zero, aplica um método iterativo pontual, que é convergente localmente. Em seguida aplica uma única iteração intervalar, obtendo um intervalo a partir da última iteração pontual, e a região inicial. Esta combinação de método pontual e intervalar possibilitou um algoritmo mais eficiente e manteve a característica da determinação automática dos limites de erro da solução aproximada (fornecida pelo próprio intervalo resultante).

Considerando-se o problema do tamanho dos intervalos resultantes (que pode muitas vezes ser relativamente grande), Claudio e Rump [CLA 92] propuseram um algoritmo cujos intervalos resultantes para as raízes aproximadas são bem menores se comparados com outros métodos, sem exigir maior esforço computacional para isto.

Observa-se entretanto, que tanto nestes trabalhos, como em outros que se preocupam com o cálculo de boas aproximações das raízes, utiliza-se como ponto inicial, o centro de regiões disjuntas (que possuem uma única raiz).

Georg, [GEO 90], em seu trabalho, propõe um algoritmo baseado na verificação de inclusões para zeros múltiplos e complexos. Neste caso, existe uma preocupação no processo de se obter aproximações iniciais que pertençam a regiões disjuntas. Previamente, um processo para enumerar raízes num círculo é aplicado, e sucessivas divisões são realizadas até obter-se regiões disjuntas, ou a exatidão desejada tenha sido atingida, ou ainda o raio dos círculos não pode ser melhorado. Esta última situação acontece quando o raio do novo disco não é menor que o raio do disco anterior. Isto pode ser devido a existência de zeros muito próximos

entre si ou próximos da fronteira do círculo. Verifica-se então que o processo de isolamento pode não fornecer as aproximações desejadas para se calcular os zeros. Apesar da utilização da aritmética de alta exatidão, nas transformações necessárias ainda ocorre um arredondamento, o que pode resultar em valores não exatos, que é imprescindível no processo de enumeração. Outros problemas sobre este algoritmo serão abordados na seção 4.1.

Sakurai, [SAK 91] coloca a idéia de combinar diferentes técnicas visando obter melhores resultados. Neste caso, a combinação diz respeito a forma de se determinar as raízes. Elas podem ser determinadas uma por vez, ou simultaneamente. A partir de aproximações iniciais dadas para cada raiz, calcula-se a primeira iteração usando um método de determinação individual. As iterações seguintes são dadas por um método de determinação simultânea de raízes, usando os resultados da primeira iteração como valores iniciais. A superioridade deste método em relação a outros conhecidos como Durand-Kerner, Farmer-Loizou é mostrada através de testes experimentais. Observa-se novamente a utilização de valores iniciais suficientemente próximos das raízes procuradas .

Schroder, [SCH 90] combina técnicas de computação simbólica e verificação de inclusões. O enfoque algébrico é utilizado somente na fase inicial, para determinar as raízes múltiplas, através de uma fatoração livre de quadrados do polinômio [DAV 88]. O isolamento das raízes é feito usando um procedimento que a partir de um retângulo inicial contendo todas as raízes, vai bisseccionando e enumerando as raízes de cada subretângulo até que se consiga isolá-las. O centro de cada retângulo isolado é usado então como uma aproximação inicial para uma rotina do sistema ACRITH [IBM 86], que calcula uma inclusão para esta raiz, através de um método iterativo intervalar. Problemas que são encontrados neste algoritmo ocorrem exatamente no procedimento de isolar as raízes: a linha que divide um retângulo não deve conter zeros do polinômio. Se isto ocorrer, um deslocamento da linha é realizado. Neste caso surge uma questão: quanto se deve deslocar, de forma que não se corra o risco de perder alguma raiz deste novo subretângulo? Outra dificuldade

que aparece é que em alguns casos, a rotina pode convergir para outra raiz, o que significa que a aproximação inicial não está na região de convergência do algoritmo iterativo intervalar. Assim, para cada inclusão calculada deve-se verificar se esta está dentro do retângulo inicial. Caso esteja fora, continua-se subdividindo para se obter um novo valor inicial.

Deren [DER 91] preocupa-se em produzir regiões iniciais que garantam a convergência do método de determinação simultânea de zeros polinomiais proposto por Durand-Kerner. Através da utilização de um subalgoritmo (Kuhn), obtém-se discos cujos raios não precisam ser tão pequenos para que o método de Durand-Kerner convirja.

Fazendo uso de técnicas de computação gráfica, Oliveira, [OLI 92] desenvolveu uma ferramenta para localização de zeros de polinômios complexos. A ferramenta auxilia o usuário no sentido de visualizar a distribuição das raízes, bem com suas multiplicidades. A aproximação das raízes entretanto é feita através de um procedimento numérico. A vantagem encontrada nesta proposta é o alto grau de interação com o usuário, que se concentra somente na região ou regiões que lhe interessam.

Dentro da área de computação simbólica, a problemática de zeros polinomiais complexos teve um trabalho pioneiro de Pinkert, [PIN 76] e por um longo período poucos trabalhos trataram especificamente deste assunto. Recentemente, Collins e Krandick [COL 92] propuseram um novo algoritmo para isolar raízes de polinômios complexos, usando basicamente o princípio do argumento e um procedimento para isolar raízes reais. Este algoritmo será detalhadamente discutido e será comparado com outra alternativa explorada neste trabalho.

Outros trabalhos que não estão diretamente relacionados com os objetivos desta pesquisa, mas que de alguma forma podem ser de interesse são os seguintes:

- Aplicação do cálculo do máximo divisor comum aproximado em equações algébricas mal condicionadas [NOD 91].
- O problema do aglomeração de raízes e intersecção de raízes [PAL 91].
- Questão dos coeficientes de um polinômio como funções bem condicionadas de suas raízes [UHL 92].
- O problema de determinação do número de zeros de uma função qualquer e sua multiplicidade através do uso de integrais [HOE 92].
- Tratamento de algoritmos paralelos para cálculo de zeros polinomiais [FRE 91]

1.3 Objetivos e Contribuições

Nesta seção serão levantados problemas que estão em aberto na aproximação exata de zeros polinomiais e as soluções propostas.

Apesar de ser uma abordagem clássica dividir o problema de resolver equações polinomiais em três fases distintas (enumeração, isolamento e aproximação), deve-se lembrar que cada uma destas fases por si só é um problema, assim como a integração entre elas.

Com respeito à enumeração, que consiste em identificar o número de raízes que o polinômio tem em uma determinada região, surgem questões do tipo: existem algoritmos satisfatórios que forneçam o resultado exato do número de zeros numa determinada região do plano? Quais as implicações que surgem quando da presença de zeros em linhas críticas como na fronteira, ou muito próximas da fronteira de uma região? Quais as diferenças encontradas no processo de enumeração em diferentes regiões do plano (por exemplo, círculos e retângulos)?

O processo de isolamento por sua vez, está diretamente relacionado com o problema da enumeração de raízes. Isto porque o processo de isolamento deve produzir um conjunto de regiões disjuntas, cada uma contendo um único zero do polinômio. Normalmente o procedimento para se obter estas regiões implica na contagem de zeros numa determinada região, a qual deve ser bisseccionada no caso de ter mais de um zero em seu interior. A aplicação recursiva deste processo deve resultar em regiões isoladas (disjuntas). Surgem então questões como: Que tipo de região é mais adequada para o processo de isolamento dos zeros polinomiais? É possível estabelecer um algoritmo eficiente para isolar de forma exata estes zeros?

Neste trabalho, buscou-se resolver algumas destas questões, tendo-se estabelecido os seguintes objetivos:

- Estabelecer as formas apropriadas de tratar cada uma das diferentes fases para aproximação exata de zeros polinomiais.
- Analisar, desenvolver e implementar algoritmos para enumerar e isolar de forma exata zeros de polinômios complexos em uma variável;
- Incorporar a utilização de subalgoritmos mais simples, através do conhecimento de propriedades do polinômio e suas raízes;
- Avaliar a adequação e desempenho dos algoritmos desenvolvidos;
- Identificar pontos deste algoritmo que permitam introduzir o uso de paralelismo.

Mediante os estudos e experiências realizadas, as seguintes contribuições foram obtidas.

Tendo-se identificado a necessidade de procedimentos de enumeração que forneçam resultados inteiros e exatos, a única alternativa segura nesta fase é a utilização de aritmética exata na implementação dos algoritmos, através da computação simbólica.

Foram analisados algoritmos para enumerar zeros em diferentes regiões do plano: círculos e retângulos. Um algoritmo algébrico, baseado no resultado de Schur-Cohn para determinação de zeros num círculo foi desenvolvido e implementado. Melhoramentos foram introduzidos, como a utilização da parte primitiva na sequência de transformações, o que mostrou um significativo ganho de eficiência (em termos de tempo de execução), comparando-se com uma simples versão algébrica do método de Schur-Cohn. Além disto, foi possível verificar a confiabilidade do algoritmo para polinômios com raízes muito próximas da fronteira ou muito próximas entre si.

O algoritmo de enumerar zeros num círculo (usando os resultados de Schur-Cohn) foi comparado com o de enumerar zeros num retângulo (usando o princípio do argumento e sequências de Sturm). A partir da implementação dos algoritmos, e observando-se o tempo necessário para a enumeração de zeros de um mesmo polinômio num círculo e num retângulo, foi possível identificar que o tempo gasto para se enumerar zeros num círculo é menor do que para enumerar zeros num retângulo (considerando-se o raio do círculo igual a semidiagonal do retângulo), e polinômios de grau até 25.

Por outro lado, as regiões circulares não se mostraram favoráveis para o processo de bissecção que é necessário para a etapa seguinte do isolamento das raízes, devido ao aumento das zonas de intersecção resultantes da bissecção, o que implica na contagem de zeros num maior número de regiões.

Desta forma, optou-se por trabalhar com regiões retangulares para desenvolver um algoritmo para isolar raízes. Baseado num algoritmo numérico de Wilf, foi desenvolvido e implementado um algoritmo modificado para o isolamento de zeros polinomiais. Um enfoque algébrico é dado ao algoritmo, visando enumerar zeros dentro de um retângulo de forma exata. O algoritmo desenvolvido não é uma mera versão algébrica do algoritmo original. Diversas melhorias foram introduzidas, principalmente no tratamento da presença de zeros nas fronteiras de um retângulo,

bem como verificação prévia do tipo de raízes e possível aplicação de algoritmos especializados.

Como um recente método para isolar zeros polinomiais foi proposto por Collins e Krandick [COL 92], o mesmo foi implementado parcialmente e foi possível estabelecer algumas comparações com o algoritmo desenvolvido neste trabalho. Testes experimentais mostraram que na fase de enumeração para polinômios reais de grau abaixo de 25, o algoritmo modificado mostrou-se mais eficiente do que o de Collins, invertendo-se a situação para graus mais elevados. Para o caso de polinômios complexos, esta mudança observou-se próxima do grau 20. Isto pode nos fornecer um indicativo do algoritmo mais adequado, de acordo com o grau do polinômio.

Na fase de isolamento, analisou-se os tempos gastos para cada atividade necessária em um nível de bissecção, para ambos os algoritmos. Os resultados mostraram-se consistentes com os obtidos na fase de enumeração.

Particularmente no algoritmo de Collins, existe a necessidade de diversos refinamentos de intervalos isolados resultantes de um processo de isolar raizes reais de 2 polinômios distintos. A quantificação do número de bissecções necessárias foi estabelecida de forma teórica e prática.

1.4 Organização do texto

Para que se tome conhecimento das etapas desenvolvidas para atingir os objetivos propostos, são apresentados a seguir os conteúdos desenvolvidos em cada capítulo deste trabalho.

O capítulo 2 apresenta diferentes ambientes de desenvolvimento de "software" científico, com suas potencialidades e limitações. Objetiva-se com este tópico introduzir as diferentes abordagens que podem ser dadas na resolução de problemas matemáticos, e identificar as implicações que se tem ao transportar um método

matemático para um programa de computador. Serão caracterizados aspectos da computação numérica e da computação simbólica, bem como aspectos de integração entre estas duas abordagens, que deverá servir para estudos futuros. Problemas encontrados na transformação de um método matemático para um programa computacional serão apresentados, como a representação não contínua dos números reais nas máquinas, a não validade de algumas propriedades aritméticas, a não equivalência de expressões, etc.

O capítulo 3 trata do problema da enumeração de zeros polinomiais em diferentes regiões do plano (círculos e retângulos). Estabelece-se os problemas com os métodos existentes e identifica-se a necessidade de uma abordagem essencialmente algébrica na fase de enumeração. Apresenta-se algoritmos para enumerar de forma exata zeros em círculos e retângulos, verificando-se o trabalho computacional para cada uma destas regiões através de testes experimentais. Toda a fundamentação matemática necessária para o desenvolvimento dos algoritmos é apresentada.

O problema de isolamento dos zeros polinomiais é tratado no capítulo 4. A escolha por regiões retangulares para o processo de bissecção é justificado. Problemas detectados em algoritmos existentes são identificados e soluções propostas são apresentadas. Um destaque especial é dado a seção que trata do algoritmo desenvolvido neste trabalho para isolar zeros polinomiais, usando sequências de Sturm. Também merece destaque a seção que trata do algoritmo recentemente proposto por Collins, pois este servirá como parâmetro de comparação com o anterior.

O capítulo 5 trata da análise dos algoritmos estudados, estabelecendo-se a complexidade teórica, bem como apresenta os resultados obtidos através de testes experimentais com polinômios diversos. O algoritmo proposto é comparado com um recente trabalho de [COL 92], estabelecendo-se a adequação deles, conforme o polinômio.

As conclusões bem como sugestões de trabalho futuro são apresentadas no capítulo 6.

2 AMBIENTES DE DESENVOLVIMENTO DE SOFTWARE CIENTíFICO

Este capítulo tem por objetivo apresentar as diferentes estratégias utilizadas para o tratamento computacional de problemas matemáticos. Duas abordagens distintas são tratadas: computação numérica e computação simbólica. O uso integrado destas duas abordagens também é discutido, pois é de grande importância para a continuidade deste trabalho. Os principais problemas encontrados no tratamento computacional de entidades matemáticas são apresentados, bem como as alternativas que se utilizam atualmente para contorná-los.

A expressão "software" científico tem um significado bastante amplo. Basicamente um software científico é um conjunto de programas devidamente organizado na forma de um sistema, servindo de suporte na resolução de problemas em áreas diversas da ciência: física, matemática, computação, química, engenharia, etc.

Tais sistemas se caracterizam pelo uso intenso de computações (simples ou complexas). Isso faz com que alguns cuidados devam ser tomados, para garantir a maior exatidão possível dos resultados fornecidos, bem como obter melhor aproveitamento dos recursos da máquina (como por exemplo memória e tempo de processamento).

Dentre os principais usuários de software científico é possível identificar 2 classes características: Primeiro, o tipo de usuário que está interessado nos resultados que o sistema pode lhe fornecer, sem se preocupar na forma como o sistema resolve os problemas; um segundo tipo é o usuário que está preocupado em desenvolver suas próprias soluções e saber quais procedimentos podem ou devem ser usados na resolução dos problemas.

Em tempos passados, os únicos recursos disponíveis eram as linguagens de programação, que praticamente só atendiam à segunda classe de usuários. Além

disso, o tipo de computação realizado era quase que exclusivamente numérica. Com a evolução da tecnologia e das técnicas de computação, novos avanços foram conseguidos: desenvolvimento de novas linguagens com abordagens mais adequadas para tratamento de problemas matemáticos; bibliotecas com rotinas matemáticas para se usar numa linguagem de programação; sistemas de resolução de problemas, e finalmente os sistemas de desenvolvimento de software científico, que combinam linguagem de programação específica com módulos para resolver problemas diversos. Estes tópicos serão descritos brevemente a seguir.

2.1 Computação Numérica e Computação Simbólica

Na área de computação científica, o tipo de abordagem que mais se usava inicialmente era a computação numérica, ou seja os algoritmos eram desenvolvidos buscando-se soluções aproximadas para os modelos em estudo. Manipulações algébricas eram pouco incentivadas de se realizar por computadores devido a grande necessidade de memória e excessivo consumo de tempo. Apesar disto, desde a década de 50/60 já se iniciava alguns estudos numa área denominada de computação algébrica (ou simbólica) [HEA 90].

Nos dias de hoje, verifica-se que ambas as áreas continuam se desenvolvendo, buscando soluções para os problemas de minimizar os possíveis erros introduzidos pela utilização de computadores. As principais características destas abordagens serão brevemente descritas a seguir, bem como o que se tem desenvolvido no sentido de se integrar estas duas abordagens.

Na computação numérica, um algoritmo deve resultar numa aproximação de um resultado exato, normalmente fazendo uso da aritmética de ponto flutuante. Operações em ponto flutuante são afetadas por erros de arredondamento, ou seja, o

resultado de uma simples operação é aproximadamente igual ao resultado exato da operação (apresentando um certo erro).

Existem muitos exemplos de computações de ponto flutuante produzindo aproximações distantes da solução exata ou mesmo exemplos onde soluções são calculadas onde na verdade elas não existem (veja [RUM 87] , [RUM 83] ,[KUL 86] e [TRE 91]).

A computação simbólica tem como objetivo automatizar uma grande variedade de computação envolvida na solução de problemas matemáticos. Sua ênfase está na computação discreta sobre símbolos que representam objetos matemáticos, tais como números inteiros ou racionais, polinômios, funções racionais, números algébricos, etc. As computações realizadas simbolicamente se caracterizam por serem exatas.

Muitos termos têm sido usados na literatura para denominar esta área: computação simbólica, álgebra computacional, computação seminumérica, etc.

. Nas últimas décadas houve um crescente progresso, tanto na área de desenvolvimento de sistemas de computação algébrica como algoritmos para resolução de problemas. Como exemplos, podem ser citados algoritmos rápidos para fatorar polinômios e calcular máximo divisor comum de polinômios [TRE 92].

A pesquisa em computação simbólica tem se destacado pelo desenvolvimento de "softwares" sofisticados. Tais "softwares" frequentemente incluem capacidade gráfica, interfaces com o usuário, software numérico e algoritmos usando técnicas de computação algébrica.

O uso de tais "softwares" cresceu significativamente nos últimos anos. Tal crescimento se justifica pelo melhoramento em "hardware" tanto em velocidade de processamento, como capacidade de armazenamento, refletindo diretamente no custo e na efetividade dos sistemas de computação simbólica. Exemplos de sistemas

de computação simbólica: REDUCE, MAPLE, MACSYMA, DERIVE, MATHEMATICA e outros.

2.1.1 A integração da computação numérica e computação simbólica

Cálculos exatos e aproximados podem ser usados de forma combinada. Computação simbólica pode ser útil para automatizar os passos para solucionar o problema matemático que precede a avaliação numérica dos modelos. A computação numérica por sua vez, automatiza o último passo da solução de um problema matemático. Usadas juntas, computação numérica e simbólica podem se completar [HEA 90]. Trabalhos que tratam desta temática podem ser encontrados em [DAV 87], [HEN 90], [NIC 91], [VAS 87].

Em termos práticos, os "softwares" (numéricos ou simbólicos) pouco oferecem em termos de interação entre si, o que seria uma característica bastante desejável. Entretanto, devido a grande demanda de computação algébrica e numérica por muitos usuários, diversos esquemas já foram propostos e implementados:

- IRENA (Interface REduce with NAg)
- GENTRAN : Facilita a geração de program FORTRAN, a partir de resultados simbólicos gerados por um sistema de computação simbólica. Tal estratégia é usada pelos sistemas REDUCE E MACSYMA [GAT 84].
- GENCRAY: Um programa escrito em linguagem C, independente do sistema de computação simbólica e tem capacidade de gerar código para fortran (77) ou Cray Fortran [WAN 89].
- Implementação de sistemas algébricos em uma linguagem que permite computações numéricas e simbólicas facilmente. Tal estratégia foi adotada pelos sistemas SMP e MAPLE, escritos em linguagem C.

- Geração de código C ou Fortran, gerado a partir de programas escritos no sistema MATHEMATICA.

- J. Wolff [WOL 90] descreve uma uma extensão da linguagem Pascal-SC que propicia formas de manipulação simbólica de expressões juntamente com aritmética de alta exatidão.

- SUI -Um sistema para integrar diferentes ambientes de computação científica [DOL 90].

Pouco se conhece sobre uma integração que seja efetiva, no sentido de propiciar troca de informações nos dois sentidos, ou seja (simbólico-numérico e numérico-simbólico). No trabalho de [SUZ 87], o autor propõe uma abordagem híbrida (ANS) que basicamente usa sistemas de linguagem convencional o máximo possível, ligando-as através de um mecanismo simples. Um dos problemas de se ligar programas escritos em linguagens diferentes é como ajustar as representações de dados que não são as mesmas em diferentes sistemas de linguagem. Como exemplo, pode-se citar inteiros de precisão fixa, que são diferentemente representados internamente em LISP e em FORTRAN. Os esquemas de integração existentes evitam este problema construindo o sistema todo numa única linguagem. A proposta do ANS converte a representação de dados quando um dado é enviado de um sistema para outro. O ANS está implementado preliminarmente sob o UNIX/VAX11-750 na linguagem C. Esta versão liga C e REDUCE.

2.2 Linguagens usadas em computação científica

Nesta seção serão abordadas somente linguagens de programação propriamente dita. Sistemas mais amplos serão discutidos na seção 2.4.

Desde os primórdios da computação científica, a linguagem FORTRAN foi a mais utilizada. Foi projetada para fins de computação numérica. Outras linguagens de uso geral surgiram posteriormente, como Algol, Pascal, Basic, Ada, C, sendo aplicáveis à computação científica.

Um destaque especial tem sido dado à linguagem C, devido às suas características de portabilidade e flexibilidade de programação. Muitos sistemas vem sendo desenvolvidos nesta linguagem. Apesar do surgimento destas novas linguagens, o Fortran, que foi pioneiro, não está fadado a desaparecer, como muitos poderiam pensar. Com o surgimento das máquinas de arquiteturas vetoriais e/ou paralelas, o Fortran volta a ser intensamente usado e consequentemente, novos compiladores foram desenvolvidos aumentando a potencialidade desta linguagem.

Independente da linguagem que está se usando, muitos cuidados devem ser tomados no desenvolvimento de programas para aplicação científica. Surpresas desagradáveis podem ocorrer no caso de se usar algumas operações elementares inadequadamente. Maiores detalhes sobre aritmética computacional serão discutidos posteriormente na seção 2.5.

Uma boa documentação sobre capacidades e limitações de linguagens aplicadas a programação científica pode ser encontrada em [PRE 90], abordando a linguagem C. O mesmo autor trata em outros trabalhos as linguagens Pascal e Fortran sob o mesmo enfoque.

Algumas extensões de linguagens foram propostas, de forma a propiciar melhores recursos adequados para a computação científica. Como exemplo, pode-se citar a linguagem Pascal-SC (Scientific Pascal) [BOH 87], Fortran-SC [MET 90]. Alguns pré-compiladores que auxiliam a integração de linguagens padrões com bibliotecas de rotinas matemáticas: Pascal TPX [HUS 91], Modula-SC [KOR 90], Pascal-XSC, M++, etc.

Uma outra maneira de incorporar recursos específicos para desenvolvimento de software científico, é através da criação de bibliotecas de rotinas matemáticas, que são de uso constante em aplicações diversas. Como exemplo, pode-se citar: manipulação com vetores e matrizes, aritmética com números complexos e intervalares, e mais uma infinidade de operações. Existem bibliotecas que são usadas intensamente, como IMSL (International Mathematical and Statistical Library), NAG (Numerical Algorithm Group), SLATEC, HARWELL, etc. Tais bibliotecas se caracterizam normalmente por conterem rotinas otimizadas e testadas exaustivamente. Apesar dessas bibliotecas resolverem problemas complexos, o acesso às respectivas rotinas normalmente é feito através de uma linguagem de programação, ou seja, um usuário final que não tem conhecimentos de programação teria acesso limitado a tais recursos.

2.3 Sistemas de resolução de problemas

Tipicamente problemas nas áreas científicas são modelados através de formulações matemáticas, como por exemplo equações algébricas, equações diferenciais, sistemas de equações, problemas com matrizes, etc. Muitos algoritmos foram desenvolvidos para tais aplicações e um conjunto de programas para resolver diversos tipos de problemas desta natureza caracterizam um sistema para resolução de problemas. Tais sistemas podem ser usados diretamente por um usuário final, ou seja, ele não precisa saber programar numa linguagem específica. Por outro lado, os programas que resolvem os problemas funcionam como uma "caixa preta" onde o usuário fornece os dados de entrada e recebe os resultados fornecidos pelo sistema. Normalmente pouco se sabe a respeito dos procedimentos efetuados para a resolução do problema propriamente dito. Isto trouxe a necessidade de se ampliar a capacidade de integração com o usuário, surgindo então novas ferramentas que permitissem a praticidade das rotinas prontas, assim como a possibilidade de se desenvolver novas rotinas.

2.4 Ferramentas de desenvolvimento de software científico

Seguindo uma linha de pensamento evolutivo, muitos dos sistemas para resolução de problemas foram ampliados, através da introdução de uma linguagem de programação apropriada. Desta forma, tais sistemas passaram a oferecer um ambiente adequado de programação para uma aplicação científica. Foi então que surgiram as ferramentas para desenvolvimento de software científico.

O objetivo destas é atender tanto a usuários finais como usuários que desejam desenvolver suas próprias soluções, porém utilizando os recursos avançados que o ambiente possa lhe oferecer.

Atualmente existem disponíveis uma infinidade destas ferramentas, sendo que uma característica comum entre elas é a preocupação em oferecer resultados confiáveis e flexibilidade de ambientes . Certamente que cada um delas apresenta suas vantagens e também suas limitações. Exemplos que podem ser citados são: ACRITH [IBM 86], ARITHMOS [SIE 86], ABACUS, MATLAB (Sistemas de computação numérica); MAPLE, MACSYMA, REDUCE, DERIVE (Sistemas de computação algébrica).

Dentro do contexto deste trabalho não cabe explorar detalhes de tais sistemas. Uma boa referência para revisões de software científico (numa abordagem bastante ampla) é a publicação Notices of American Mathematical Society que publica frequentemente artigos desta natureza.

2.5 Aritmética Computacional e Representação de dados

Nos primeiros computadores a forma de representar os números era através de numeração de ponto fixo. (Os números são armazenados com um número fixo de casas decimais ou binárias, tendo um número máximo fixo de dígitos antes do ponto decimal ou binário).

Neste sistema de numeração, a propriedade de fechamento não vale sobre várias operações aritméticas e as leis distributivas podem falhar de acordo com as regras de arredondamento e do tamanho dos registradores usados na unidade aritmética. Uma vantagem deste tipo de sistema é a simplicidade da análise de erro, pois a única medida apropriada de precisão é o erro absoluto e as regras de propagação de erro são simples.

As limitações deste sistema levaram ao desenvolvimento de ariméticas computacionais mais evoluídas: os sistemas de ponto flutuante. Um ponto bastante questionável foi a exatidão de tais computações. Ela pode não ser confiável. Os sistemas de ponto flutuante foram melhorados e o que se tem de melhor é a aritmética estabelecida pela IEEE de simples e dupla precisão.

O conjunto de números representáveis aumentou bastante com relação a um sistema de ponto fixo, entretanto dificuldades de "overflow" e "underflow" ainda permanecem.

Alguns outros incovenientes se mostram pela não validade da propriedade de fechamento para algumas operações, da propriedade associativa da adição e ainda relações de ordem básica.

Os resultados de uma computação composta de diversas operaçès básicas podem estar drasticamente errados. Como exemplo citado por [KUL 86], considere a determinação da soma abaixo:

$$10^{50} + 812 - 10^{50} + 10^{35} + 511 - 10^{35} = 1323$$

Se estes números forem somados da esquerda para a direita, muitos computadores digitais devem fornecer 0 como resposta. Isto ocorre pela eventual presença de um erro de "overflow" ao se tentar executar tal operação. Entretanto, os elementos desta soma podem ser rearranjados de forma que este erro não ocorra. Percebe-se assim que a propriedade associativa não é válida para os números de ponto flutuante (ou números de máquina).

Uma boa discussão sobre aritmética de ponto flutuante, procedimentos para controle de erro são encontrados em [KUL 86].

Outro problema que existe é com respeito a múltipla representação para um mesmo dado, conforme observa [TRE 91].

Exemplo 1:

$\frac{40}{5}$ e 40*0.2 podem resultar em valores diferentes: A primeira expressão resulta no valor 8, e a segunda resulta em 7.9999. Isto ocorre porque 0.2 não é exatamente representável num formato binário ou hexadecimal.

Exemplo 2:

$\frac{1}{3} + \frac{1}{3} + \frac{1}{3} = 1$

Ao se usar representação num sistema de ponto flutuante com 8 dígitos de precisão, tem-se:

0.3333333+ 0.3333333+0.3333333=0.9999999 (que é diferente de 1).

Da mesma forma:

$\frac{2}{3} - \frac{1}{3} = \frac{1}{3}$ que é diferente de

0.6666667 - 0.3333333 = 0.3333334

Ou seja, tem-se duas representações para um mesmo número.

Embora as operações elementares possam ter erro reduzido, quando se trabalha com espaços do tipo vetores de reais, matrizes, números complexos, intervalos, o problema se agrava. Uma operação bastante utilizada nestes espaços é o produto escalar. Exemplo: multiplicação de números complexos, multiplicação de matrizes, etc. O número de arredondamentos que são realizados nestas operações é bastante grande, podendo acarretar um erro final considerável.

Algoritmos eficientes para produto escalar são quase impossíveis de serem realizados num sistema padrão de ponto flutuante. Um "hardware" adequado para que tais operações possam ser executadas com máxima exatidão foi desenvolvido, e algoritmos foram incorporados num pacote de aritmética intervalar desenvolvido por Kulisch e sua equipe da Universidade de Karlsruhe [KUL 86].

Outra proposta alternativa de aritmética é dada por Peter Turner [TUR 91]: a aritmética **sli** (symmetric level-index). A idéia básica deste sistema é a seguinte: a representação de um número maior que a unidade pode ser obtido, tomando-se repetidamente logaritmos naturais até o resultado pertencer ao intervalo [0, 1). O número original é representado então pelo número de vezes que o logaritmo foi feito (nível) e o logaritmo final (índice). Algumas vantagens e limitações desta alternativa: a quebra da relação de ordem é menos severa: números positivos para os quais 1+x=x deve ser bem menor para o sistema **sli** do que no sistema de ponto flutuante convencional. Continuam as limitações de um conjunto finito de representações disponíveis e a não associatividade da adição. A grande vantagem do sistema **sli** é a propriedade de fechamento que vale para todas operações aritméticas (exceto divisão por zero), deixando de existir o problema de "overflow" e "under-

flow". Uma das maiores críticas a este sistema é o retardamento provocado nas computações.

Do ponto de vista matemático, a estrutura do espaço dos números de máquina com operação em ponto flutuante é muito pobre [CLA79]. Abordagens alternativas continuam sendo propostas para controlar o problema. Dentre estas merecem destaque verificação de inclusões e manipulação simbólica de expressões [RUM 87].

3 A ENUMERAÇÃO DE ZEROS POLINOMIAIS

Conforme já colocado anteriormente, as fases de enumeração e isolamento das raízes de um polinômio são problemas importantes no processo da aproximação destas raízes. Este capítulo tem como objetivo expor e discutir problemas que dizem respeito a enumeração de zeros polinomiais. Serão analisados algoritmos para diferentes regiões do plano (círculos , retângulos e semi-planos), identificando-se problemas e propondo soluções. Toda a fundamentação matemática necessária para o entendimento dos algoritmos será introduzida no decorrer do texto (As demonstrações não são feitas neste trabalho, pois não são prioritárias neste trabalho, fazendo-se uma referência para localização das mesmas). Com relação as regiões retangulares, apenas uma breve apresentação dos conceitos é apresentada. Uma atenção especial a estas regiões será dada nos capítulos seguintes, devido a sua utilização nos principais algoritmos de isolamento que serão analisados. (Veja seção 4.3 e seção 4.5).

3.1 Colocação do problema

Dado um polinômio de coeficientes complexos $p(z)$, e uma região R no plano complexo, deseja-se saber quantos zeros de $p(z) \in R$. As regiões alvo de busca podem ter diferentes geometrias: circulares, retangulares, poligonais, semi-planos, lineares, etc. Alguns resultados teóricos foram desenvolvidos há muito tempo como por exemplo o princípio do argumento [MAR 49], que diz respeito ao número de zeros que uma função tem no interior de uma curva simples e fechada. Também de grande importância é o teorema de Sturm [MAR 49], através do qual é possível determinar-se o número de zeros de um polinômio real num intervalo aberto. Tais resultados contribuem para a elaboração de algoritmos que resolvam o problema estabelecido.

Entretanto estes não se constroem pela mera utilização das teorias existentes. Além disto, outro problema que surge é a utilização de um algoritmo correto num ambiente computacional, utilizando números de máquina. Estes problemas ocorrem devido à não equivalência do universo matemático e o universo computacional (veja seção 2.5). A situação é mais crítica quando existem raízes muito próximas entre si ou muito próximas da fronteira da região que está sendo analisada.

Cabe salientar que a resposta da enumeração deve ser um número inteiro, o que caracteriza um processo necessariamente algébrico. Valores aproximados não fazem sentido algum. Isto implica que este processo deve ser feito de forma rigorosamente exata. Assim, esta fase justifica o uso de técnicas de computação simbólica.

Apesar do custo computacional ser relativamente alto neste tipo de tratamento, é a única alternativa para se obter resultados exatos, assegurando a confiabilidade do processo de enumeração. Entretanto, vale lembrar que esta escolha está diretamente relacionada com a necessidade de uma aritmética exata. Existem aplicações que não requerem tanto rigor para se obter resultados aceitáveis, como por exemplo, a aproximação de raízes polinomiais (supondo-se que estas já foram isoladas previamente).

Uma tentativa de compartilhar abordagem numérica e simbólica nesta fase de enumeração não obteve sucesso. A avaliação de funções em pontos estritamente numéricos foi realizada utilizando-se aritmética de ponto flutuante. Isto resultou para alguns casos, na contagem errada de zeros na região pesquisada.

A questão do alto custo computacional deve ser gerenciada no desenvolvimento dos algoritmos, conforme será visto adiante. Assim como na abordagem numérica, as computações realizadas em ponto flutuante requerem cuidados especiais para evitar "overflow", "underflow" e algoritmos instáveis, na abordagem simbólica, alguns cuidados devem ser tomados. Como exemplo pode-se citar a forma de representação dos dados: a possibilidade de se trabalhar com números racionais em sua representação de numerador e denominador requer sempre o trabalho de simplificar

frações. Desta forma, os dados manipulados por um algoritmo devem ser tratados adequadamente, minimizando as operações dispendiosas.

3.2 O algoritmo de Schur-Cohn para círculos

Uma forma de se determinar o número de zeros que um polinômio tem em um círculo arbitrário é através do uso do algoritmo de Schur-Cohn. Este algoritmo é usado para contar o número de zeros no interior de um círculo unitário. Seja $p(z)$ um polinômio de coeficientes complexos e $Z = (0,1)$ o círculo unitário (com centro na origem e raio 1).

O algoritmo de Schur-Cohn requer a formação de uma sequência de polinômios $f_i(z) = \sum_{k=0}^{n-i} a_k^{(i)} z, i = 1,2,...n$, definidos por $f_0(z) = p(z)$

$$f_{i+1}(z) = f_i(z) - m_{i+1} f_i^*(z), i = 0,1,...,n-1 \tag{3.1}$$

onde $m_{i+1} = \frac{a_{n-i}^{(i)}}{\overline{a}_0^{(i)}}$ e $f_i^*(z)$ é o polinômio recíproco de $f_i(z)$, isto é

$$f^*(z) = z^n \overline{p}(\frac{1}{\overline{z}}) = \sum_{i=0}^{n} \overline{a_i} z^{n-i}$$

Os coeficientes desta sequência podem ser calculados da seguinte forma:

$$a_i^{(k+1)} = (\overline{a}_0^k a_i^{(k)} - a_{n-k}^{(k)} \overline{a}_{n-k-i}^{(k)})$$

$i = 0,1...n-k-1$, $k = 0,1...$,

Considerando agora $\gamma_k := a_k(0) = a_0^{(k)}$, $k = 1...n$, γ_k será sempre um número real para todo k, pois

$$\gamma_k = |a_0^{(k-1)}| - |a_{n-k}^{(k-1)}|$$

é real.

Se todos γ_k satisfizerem $\gamma_k \neq 0$, para todo k, então, o número de zeros no interior do círculo unitário pode ser determinado da seguinte forma:

Toma-se todos $\gamma_k < 0, k = 1...n$ e denota-se os índices k com $\gamma_k < 0$ por k_j, $j = 1...m$, onde $k_1 < k_2 < ...k_m$. então o número $h(p)$ de zeros z_i satisfazendo $|z_i| < 1$ é dado por:

$$h(p) = \begin{cases} \sum_{j=1}^{m}(-1)^j k_j & se\ m\ par \\ \sum_{j=1}^{m}(-1)^j k_m + n + 1 & se\ m\ impar \end{cases} \tag{3.2}$$

contando suas multiplicidades.

Para demonstração, consulte [HEN 74].

Como exemplo, considere o seguinte polinômio:

$$p(z) = p^0(z) = 9 + 3z - 14z^2 - 8z^3$$

Então:

$$\overline{a}_0^1 = 9 \times 9 - (-8 \times -8) = 17$$

$$\overline{a}_1^1 = 9 \times 3 - (-8 \times -14) = -85$$

$$\overline{a}_2^1 = 9 \times -14 - (-8 \times 3) = -102$$

$$\overline{a}_3^1 = 9 \times -8 - (-8 \times 9) = 0$$

Assim, $p^1(z) = -102z^2 - 85z + 17$

$$\overline{a}_0^2 = 17 \times 17 - (-102 \times -102) = 933$$

$$\overline{a}_1^2 = 17 \times -85 - (-102 \times -85) = 10115$$

$$\overline{a}_2^2 = 17 \times -102 - (-102 \times 17) = 0$$

Assim, $p^2(z) = 10115z + 933$

$$\overline{a}_0^3 = 933 \times 933 - (10115 \times 10115) = -101442736$$

$$\overline{a}_1^3 = 933 \times 10115 - (10115 \times 933) = 0$$

Assim, $p^3(z) = -101442736$.

Tem-se então que $\gamma_1 = 17$, $\gamma_2 = 933$ e $\gamma_3 = -101442736$. Logo $k_1 = 3$, e como só existe um γ_k negativo, $m = 1$. Então $h(p)$ é dado por $(-1)^1 \times 3 + (3+1) = 1$, ou seja $p(z)$ possui um zero no círculo unitário.

Existem outras formas de expressar $h(p)$ (veja [HEN 74]). Se algum $\gamma_k = 0$ para algum k, este algoritmo não pode ser usado. Tal fato pode ocorrer quando p tem zeros na fronteira do círculo ou quando p=p*.

O número de raízes dentro de um círculo arbitrário aberto com centro a e raio r pode ser computado através da transformação polinomial:

$$q(z) = p(a + rz).$$

Qualquer zero z_i do polinômio complexo p que está no interior de um disco com raio r e centro a equivale a um zero de polinômio complexo $q(z)$ no interior do círculo unitário.

A questão da limitação do uso do algoritmo de Schur-Cohn para polinômios sem zeros na fronteira do círculo pesquisado pode ser contornada determinando-se o número de zeros que um polinômio tem no interior do círculo unitário, através do mapeamento do círculo unitário no semiplano esquerdo, através da seguinte transformação:

$$w = \frac{(z-1)}{(z+1)}, z = \frac{(1+w)}{(1-w)}$$

O número de zeros dentro do círculo unitário é o mesmo que o polinômio transformado:

$$F(w) = (1-w)^n f\left(\frac{1+w}{1-w}\right) \tag{3.3}$$

tem no semiplano esquerdo. A determinação do número de zeros de um polinômio num semiplano pode ser feita usando o teorema de Routh. Ver teorema (3.4.1).

Como exemplo, tome-se o seguinte polinômio:

$$p(z) = 300z^4 - 400z^3 + 187z^2 - 35z + 2 \tag{3.4}$$

Fazendo a transformação 3.3, e adequando os coeficientes para serem inteiros, tem-se:

$$p(w) = 27w^4 - 231w^3 + 719w^2 - 961w + 462$$

Ou seja é preciso determinar quantos zeros p(w) tem no semiplano esquerdo. (veja o mesmo exemplo na seção 3.4, que mostra a equivalência de resultados, usando o teorema de Routh).

Um dos problemas que se tem com o algoritmo de Schur-Cohn é o rápido crescimento dos coeficientes para se obter polinômios transformados.

Uma implementação numérica deste algoritmo está sujeita a erros de arredondamento inerentes do sistema de ponto flutuante. Além disto, os números resultantes podem ter magnitude muito grande, podendo levar a ocorrência de "overflow". Desta forma, os resultados podem não ter sucesso ou serem duvidosos, principalmente se existirem zeros muito próximos da fronteira.

Como uma alternativa foi desenvolvida e implementada uma versão algébrica do método de Schur-Cohn. Nesta primeira tentativa, observa-se que embora os resultados obtidos fossem exatos, as expressões intermediárias eram muito grandes, tornando o algoritmo ineficiente.

Para reduzir o tamanho dos coeficientes, a seguinte modificação foi realizada: Para cada polinômio gerado na sequência de Schur, toma-se a sua parte primitiva. A parte primitiva de um polinômio $p = \sum_{i=0}^{n}(a_i x^i)$ é dada por $\frac{p}{mdc(a_i)}$. Note-se que ao multiplicarmos (ou dividirmos) os polinômios $f_i(z)$, i=0, ... n por uma constante, o sinal de γ_k não se altera.

O pseudo-código abaixo mostra o algoritmo.

Entrada: $P(z)$, $Z(c,r)$, onde $P(z)$ é um polinômio de grau n e $Z(c,r)$ é o círculo com centro c e raio r.

Saída: N, que é o número de zeros de $P(z)$ no interior de $Z(c,r)$.

Procedimento:

$f_0 \leftarrow P(c + z.r)$

$f_0^* \leftarrow$ recíproco(f_0)

$L = ()$ {L é uma lista de trabalho}

Para $k \leftarrow 0$ até n faça

$f_{k+1} \leftarrow f_k - m_{k+1}.f_k^*$ conforme 3.1

$f_{k+1} \leftarrow$ pp(f_k), {pp = parte primitiva}

Se termo independente de $f_{k+1} < 0$

Então inclui k+1 em L

Obtém N, conforme 3.2, k_j é o j-ésimo elemento de L

O uso da parte primitiva assegura que os coeficientes gerados são sempre inteiros e menor que na forma original, facilitando o tratamento com estes coeficientes.

Um programa para implementar este algoritmo com estas modificações foi desenvolvido no sistema de Computação Algébrica Macsyma. A execução de diversos testes mostrou que há uma redução de tempo em relação à forma original [CAM 93].

Uma característica importante que deve ser ressaltada neste algoritmo adaptado é a exatidão garantida dos resultados fornecidos, mesmo para raízes próximas à fronteira do círculo, assim como raízes aglomeradas.

A seguir são apresentados alguns polinômios que foram testados usando a implementação proposta do algoritmo de Schur-Cohn.

- $p1 : z^3 - \frac{29889}{10000} * z^2 + \frac{29778111}{10000000} * z - \frac{988911099}{1000000000}$ e $Z = (0, 1)$
- $p2 : z^4 - \frac{985}{348} * z^3 + \frac{1}{348} * z^2 + \frac{985}{174} * z - \frac{697}{174}$ e $Z = (2, 3)$
- $p3 : (z - 3) * (z - \frac{11}{10}) * (z - \frac{9}{10}i) * (z - \frac{1}{10}i) * (z - \frac{97}{100}i)$ e $Z = (0, 1)$

Tabela 3.1: Tempos observados na execução do algoritmo de Schur-Cohn, em sua forma original e usando a parte primitiva

	Tempo de execução(seg)	
Polinômio	Sem usar pp	Usando pp
$p1$	0.482	0.480
$p2$	0.663	0.623
$p3$	3.046	1.733
$p4$	1.640	1.306
$p5$	1.245	1.073
$p6$	86.250	22.015
$p7$	56.47	24.08
$p8$	30.726	15.711

- $p4 : (z \pm \frac{3}{10}) * (z \pm \frac{1}{5}) * (z \pm \frac{1}{10}) * (z - \frac{2}{5})$ e $Z = (0, 1)$
- $p5 : (z - \frac{1}{3}) * (z - \frac{1}{6})^5 * (z - 2)$ e $Z = (0, 1)$
- $p6 : (z - 1)^5 * (z - 5/2)^5$ e $Z = (1, 11/2)$
- $p7 : (z - 2) * (z + 3)^{14}$ e $Z = (1, 5)$
- $p8 : (z - 1)^{20}$ e $Z = (0, 2)$

Uma característica deste algoritmo é que ele pode ser usado como uma forma de se fazer um teste rápido para descartar as regiões livres de zero no posterior processo de isolamento de raízes. Se houver algum $\gamma_k \leq 0$, então o círculo pode conter zeros. Caso contrário, o círculo está livre de zeros e pode ser desprezado.

3.3 O Princípio do Argumento e Sequências de Sturm para retângulos

Uma das formas clássicas de se contar zeros de uma função dentro de uma região fechada é através do princípio do Argumento.

Teorema 3.3.1 (Princípio do Argumento) *Seja R 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. Então o número N de zeros de $p(z)$ dentro de R, contando suas multiplicidades é dado por:*

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

Veja [MAR 49]

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 $\frac{Im\, p}{Re\, p}$ quando z percorre δR. Quando $p(z)$ cruza o eixo imaginário num sentido anti-horário, $\tan \arg p(z)$ 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 3.3.1 (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 onde $\frac{p}{q}$ salta de $+\infty$ to $-\infty$.*

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

Teorema 3.3.2 *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 [MAR 49]

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:

$$\Delta_{\delta R} \arg p(z) = -\pi \sum_{i=1}^{m} I_l^0 \frac{Im\, p_j}{Re\, p_j} \tag{3.5}$$

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

Usando os teoremas 3.3.1 e 3.3.2, pode-se concluir que o número de zeros polinomiais dentro de um retângulo pode então ser obtido por:

$$N = -\frac{1}{2} \sum_{i=1}^{4} I_0^l \frac{Im\, p_j}{Re\, p_j} \tag{3.6}$$

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

Definição 3.3.2 (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 3.3.3 *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 [HEN 74]

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

Teorema 3.3.4 (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:*

$$I_a^b(\frac{f_1(x)}{f_0(x)}) = V(a) - V(b) \tag{3.7}$$

Veja [HEN 74]

Combinando as expressões 3.6 e 3.7, tem-se:

$$N = \frac{1}{2} \sum_{1}^{4} (V(l_j) - V(0)) \tag{3.8}$$

Veja [WIL 78]

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 teorema 3.3.3, 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, [DAV 88], 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 $R_k = \sum(a_i x^i)$.

$$f_{k+1} = \begin{cases} R_k/pp(R_k) & se\ cont(R_k) < 0 \\ -R_k/pp(R_k) & caso\ contrário \end{cases}$$

onde pp(R_k) é a parte primitiva de R_k e cont(R_k)=mdc(a_i).

Esta sequência continua sendo uma sequência de Sturm. (Generalizando o teorema proposto por Heindel [HEI 71].

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

$$f_0 = 27z^4 - 719z^2 + 462 \text{ e}$$

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

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.

3.4 O Teorema de Routh para enumerar zeros num semiplano

A enumeração de zeros polinomiais nos semiplanos superior e inferior do plano complexo pode ser feita usando sequências de Sturm. O teorema de Routh estabelece o seguinte:

Teorema 3.4.1 (Teorema de Routh) *Dado $P(z) = P1(z) + iP2(z)$, onde $P1(z)$ e $P2(z)$ são polinômios reais, com $P2(z)$ diferente de 0, e que não tenha zeros reais, $n1$ zeros (contando muliplicidades) no semiplano superior do plano complexo e $n2$ zeros (contando multiplicidades) no semiplano inferior do plano complexo. Seja $V(x)$ variação de sinal obtida no ponto x para a sequência de Sturm iniciada com*

P1 e P2, avaliando-se x em $(-\infty, +\infty)$. *Então para n=grau(P),*

$$n1 = 1/2[n + V(+\infty) - V(-\infty)] \text{ e}$$

$$n2 = 1/2[n - V(+\infty) + V(-\infty)].$$

Para que este teorema possa ser usado para enumerar semiplanos arbitrários, faz-se necessário o uso de transformações adequadas: Para se contar zeros em semi-planos horizontais arbitrários, deve-se fazer uma transformação do tipo translação:(seja $z = a + bi$, $z^* = (a + d) + bi$). Para se contar zeros em semi-planos verticais arbitrários, deve-se fazer uma transformação do tipo rotação: (se $z = a+bi$, $z^* = i.(a + bi)) = ai - b$.

Seja o polinômio do exemplo 3.4 (veja seção 3.2).

$$p(w) = 27w^4 - 231w^3 + 719w^2 - 961w + 462$$

Para determinar-se o número de zeros de p(w) no semiplano esquerdo, equivale a determinar o número de zeros que p(iw) tem no semiplano inferior. Assim

$$\overline{p}(w) = p(iw) = 27w^4 + 231iw^3 - 719w^2 - 961iw + 462$$

Seja

$$pr = 27w^4 - 719w^2 + 462 \text{ e}$$

$$pi = 23w^3 - 961w$$

onde pr são os coeficientes reais de $p(iw)$ e pi são os coeficientes da parte imaginária de $p(iw)$.

A sequência de Sturm gerada, iniciando-se com $f_0 = pr$ e $f_1 = pi$ fica então da seguinte forma:

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

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

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

$$f_3 = z$$

$$f_4 = 1$$

Para obter $V(\infty)$, basta verificar a variação de sinal nos termos de mais alta potência na sequência de Sturm. Assim, $V(\infty)$=0. $V(-\infty)$ equivale ao número de variação de sinal entre os termos de mais alta potência na sequência de Sturm, trocando-se os sinais destes termos de grau ímpar [DAV 88]. Assim, $V(-\infty)$=4.

Aplicando o teorema de Routh, tem-se então que o número de zeros $n2$ no semiplano inferior é dado por

$$\frac{1}{2}(4 - 0 + 4) = 4.$$

A combinação de semiplanos horizontais e verticais pode fazer com que se consiga determinar o número de zeros dentro de um retângulo. Tal forma é utilizada no algoritmo de Pinkert para isolar raízes que será visto no capítulo seguinte (veja seção 4.2).

Uma outra abordagem para contar zeros no interior de um retângulo usando o princípio do argumento 3.3.1 foi introduzida por Collins e Krandick [COL 92]. Uma das formas de se contar as mudanças em arg $p(z)$, quando z percorre a curva definida pela fronteira de um retângulo (δR), é através da verificação

das trocas de quadrantes de $p(z)$ quando z percorre a fronteira δR. Isto pode ser reduzido ao problema do isolamento de zeros reais que o polinômio tem ao percorrer δR. Este algoritmo será descrito em detalhes no capítulo seguinte, pois apesar de ter um funcionamento relativamente simples, a apresentação do mesmo é extensa e requer a utilização de um subalgoritmo para isolar raízes reais.

3.5 Comparando a enumeração em círculos e retângulos

Tendo em vista analisar as diferenças que ocorrem no processo de enumerar zeros polinomiais em círculos e retângulos, foram testados dois programas desenvolvidos que aplicam os teoremas apresentados anteriormente para contar o número de zeros no interior destas regiões. A partir de polinômios com coeficientes gerados aleatoriamente, a tabela 3.2 expressa os tempos observados para enumeração de zeros em círculos e retângulos. Particularmente considerou-se um quadrado com semidiagonal igual ao raio do círculo. Os tempos estão expressos em segundos

Os tempos observados mostram que a enumeração em círculos é mais rápida do que a enumeração em retângulos, para polinômios de grau até 25.

Cabe lembrar novamente, que o processo de enumeração está intimamente relacionado com o problema de isolamento das raízes de um polinômio. Embora as regiões circulares mostrem-se aparentemente melhores para a contagem dos zeros, elas não se mostram adequadas para um processo de bissecção, normalmente requeridas no processo de isolamento. Não se pode esquecer entretanto que a determinação do número de zeros nessas regiões é um problema relevante e os resultados conseguidos poderão ser utilizados em trabalhos futuros.

Tabela 3.2: Tempos observados para enumerar zeros num círculo e num retângulo

grau	Schur-Cohn	Sturm
2	0,23	0,73
3	0,3	1,17
4	0,43	1,67
5	0,65	2,28
6	0.92	4.12
7	1,42	5,43
8	3	7,52
9	3,97	11,62
10	6,33	15,23
11	8,9	22,75
12	9,9	28,17
13	16.23	43,65
14	39.22	65,28
15	45,4	76,12
20	185,3	275,4
25	765,25	773.50
30	2027	1804.15

4 LOCALIZAÇÃO EXATA DE RAíZES POLINOMIAIS

O processo de isolamento de raízes polinomiais é de fundamental importância para que se possa posteriormente calcular todas as raízes de um polinômio. O resultado deste processo deve ser um conjunto de regiões disjuntas, cada uma contendo um único zero. Existem muitos métodos de determinação simultânea de zeros que requerem que estes estejam previamente isolados. Entretanto, este não é um problema que se resolve trivialmente, sob o ponto de vista computacional.

Algoritmos baseados em procedimentos de busca de regiões mostram-se adequados para isolar raízes devido à sua característica de convergência global. Tais algoritmos baseiam-se na execução de sucessivos testes para detectar a existência de zeros em regiões que posteriormente são subdivididas. Quando uma região contém zeros, é preciso contar quantos zeros existem nesta região, para decidir se nova subdivisão deve ser feita ou não. Isto implica na necessidade imprescindível de algoritmos exatos para contar (enumerar) as raízes de um polinômio numa região.

Dispondo de algoritmos para enumerar raízes, diferentes formas de isolá-las podem ser efetuadas. A maioria delas consiste em algum processo de particionamento (ou bissecção) de regiões que contém mais de um zero. Este particionamento realizado recursivamente pode levar ao isolamento de todos os zeros polinomiais, ou seja, obter uma sequência de regiões disjuntas, cada uma contendo exatamente um zero.

Entre outros aspectos, serão discutidas as diferentes geometrias (círculos e retângulos) no processo de isolamento. Também será realizada uma análise de algoritmos para isolar zeros polinomiais, identificando-se seus problemas e limitações, bem como a apresentação de novas idéias para solucionar alguns dos problemas.

Um aspecto importante que deve ser lembrado é a questão da determinação das multiplicidades das raízes. Os algoritmos simbólicos (ou algébricos) abordados neste capítulo utilizam a aplicação prévia de uma decomposição livre de quadrados, a qual fornece fatores, cujas raízes são necessariamente simples. Os algoritmos então trabalham com cada um dos fatores separadamente.

Decomposição Livre de quadrados

Seja p um polinômio mônico de coeficientes inteiros. Considere p como um produto de fatores lineares

$$p = \prod_{i=1}^{n} (x - a_i)^{n_i}.$$

onde n_i é a multiplicidade a cada fator.

A derivada p' de p, pode ser calculada algebricamente, a partir dos fatores de p :

$$p' = \sum_{i=1}^{n} \left(n_i (x - a_i)^{n_i - 1} \prod_{j=1, i \neq j}^{n} (x - a_j)^{n_j} \right).$$

Portanto,

$$mdc(p, p') = \prod_{i=0}^{n} (x - a_i)^{n_i - 1}.$$

Então

$$q = p/mdc(p, p') = \prod_{i=0}^{n} (x - a_i)$$

Assim,

$$mdc(q, mdc(p, p')) = \prod_{i=1, n_i>1}^{n} (x - a_i)^{n_i}.$$

Do qual conclui-se que

$$\frac{q}{mdc(q, mdc(p, p'))} = \prod_{i=1, n_i=1}^{n} (x - a_i)$$

Nesta decomposição de p, cada fator separadamente não tem fatores múltiplos e todos eles são relativamente primos. Observe que somente operações de derivação, divisão exata e máximo divisor comum (mdc) foram usadas. Estas operações ocorrem no domínio dos inteiros, o que implica num método eficiente. Diversos sistemas de computação algébrica possuem algoritmos eficientemente implementados para obter uma decomposição livre de quadrados, facilitando assim a determinação prévia das multiplicidades das raízes de um polinômio. Serão considerados a partir daqui, polinômios livre de quadrados.

Uma ênfase especial será dada ao desenvolvimento de um novo algoritmo baseado no método de Wilf [WIL 78] numa abordagem simbólica (seção 4.3) e ao algoritmo de Collins [COL 92] (seção 4.5).

Os procedimentos utilizados para isolar as raízes devem iniciar por uma região que contenha todas elas. Para isso, utiliza-se o conceito de quota superior do módulo de uma raiz sobre o qual se constrói uma região inicial de pesquisa.

Dado um polinômio $p(z) = \sum_{i=0}^{n} a_i z^i$, e α uma raiz de p, uma quota superior bastante usada, é dada por:

$$b = |\alpha| \leq 2 \max_{1 \leq i \leq n} |a_{n-i}/a_n|^{1/i} \tag{4.1}$$

[MAR 49]

Uma forma alternativa, proposta por Collins [COL 77] é calcular esta quota superior como uma potência de 2, evitando a presença de números irracionais.

Seja

$$l = \max_{1 \leq i \leq n \& a_{n-i} \neq 0} |(log_2 |a_{n-i}/a_n|)/i| \; então \; \overline{b} = 2^{l+1} \tag{4.2}$$

Outras quotas superiores que foram estabelecidas podem ser encontradas em [MAR 49]. Collins [COL 77] observa que os diversos métodos para se definir uma quota superior de raízes podem ser melhores ou não, dependendo do caso. Entretanto, as diferenças existentes entre os métodos conhecidos não são significativas, a ponto de incentivar pesquisa nesta área.

O algoritmo descrito a seguir utiliza regiões circulares para isolar os zeros de um polinômio.

4.1 Algoritmo de Henrici modificado

Stefan Geörg [GEO 90] propôs uma implementação do algoritmo de Henrici [HEN 69] baseado no método de Schur-Cohn para determinar número de zeros no círculo unitário. Lehmer [LEH 61] já havia desenvolvido um método para isolar raízes polinomiais usando os resultados de Schur-Cohn, porém a localização se dá para uma raiz de cada vez.

O número de raízes dentro de um círculo arbitrário aberto com centro a e raio r pode ser computado através da transformação polinomial $q(z) = p(a + rz)$.

Qualquer zero z_i do polinômio complexo p que está no interior de um disco com raio r e centro a equivale a um zero de polinômio complexo $q(z)$ no interior do círculo unitário.

Esta transformação pode ser efetivada em 2 passos:

1. Uma transformação de deslocamento: $P(z) = p(a + z)$; que pode ser realizada pelo esquema de Horner.

$$P_j^{(-1)} = p_j, j = 0..n;$$

$$P_n^{(k)} = p_n, k = 0..n;$$

$$p_j^{(k)} = \diamond(aP_{j+1}^{(k)} + P_j^{(k-1)}), k = 0..n;\ j = n-1, n-2..k$$

onde $\diamond$ denota arredondamento intervalar. [KUL 86]

Para evitar erros de arredondamento e cancelamento no esquema de Horner, usa-se aritmética intervalar complexa para incluir os coeficientes em intervalos complexos.

2. Uma transformação de escala $Q(z) = P(r * z)$ que pode ser feita determinando-se $Q_i = r^i P_i, i : 0..n$.

Se n é grande e $r > 1$ a chance de ocorrer "overflow" de expoente é grande.

Assim, torna-se melhor tomar:

$$Q_i = \begin{cases} (tr^i)P_i & se\ r < 1 \\ (tr^{(i-n)}P_i & se\ r > 1 \end{cases}$$

onde t denota a maior potência de 10 satisfazendo $(0 < t < maxpf)$, onde $maxpf$ denota maior número de ponto flutuante representável no computador.

Os coeficientes da sequência de Schur são calculados da seguinte forma:

$$a_i^{(k+1)} = \diamond(\overline{a}_0^k a_i^{(k)} - a_{n-k}^{(k)} \overline{a}_{n-k-i}^{(k)})$$

, $i = 0, 1...n-k-1$, $k = 0, 1...$, onde $\diamond$ denota arredondamento intervalar.[KUL 86]

O lado direito desta expressão é computado usando produto escalar exato que efetua um único arredondamento [KUL 86].

Combinando o algoritmo de Schur-Cohn com as transformações acima referidas, pode-se determinar número de zeros em qualquer região circular do plano complexo.

O procedimento se faz da seguinte forma:

Parte-se de um disco inicial com raio determinado pela conhecida quota superior $b = 2* > \max_{1 \leq i \leq n} \frac{|a_{n-i}|}{a_n}^{1/i}$ contendo todos zeros do polinômio complexo. A operação $* >$ denota produto com arredondamento para cima [KUL 86], que é usada para garantir que todos os zeros estejam dentro do círculo.

O disco é cercado por um quadrado. Este quadrado é então dividido em 4 subquadrados. Cada subquadrado é cercado por um disco. Aplica-se o teste de Schur-Cohn a cada um destes discos. Os discos que não possuem zeros são descartados. Os outros são denominados discos suspeitos. Para cada disco suspeito o procedimento é repetido. Isto acontece até que os zeros estejam separados ou a exatidão desejada tenha sido atingida ou ainda quando o raio dos círculos não pode ser melhorado. Esta última situação acontece quando se tem zeros muito próximos entre si ou próximos da fronteira (o que levaria a uma incerteza com respeito ao resultado da contagem dos zeros num determinado disco).

Os primeiros incovenientes de se usar regiões circulares num processo de particionamento de regiões foram resolvidos neste algoritmo. Isto porque o processo de isolamento não poderia ser realizado utilizando somente círculos, uma vez que círculos quando bisseccionados têm partes componentes que não são círculos. A alternativa adotada de se trabalhar com retângulos (quadrados neste caso) envolvendo os círculos resolve este problema, entretanto cria um outro, que é o aumento das regiões de intersecção que aparecem neste processo de bissecção.

Os problemas de incerteza quanto ao número de zeros numa região poderia ser resolvido através da utilização de aritmética exata. Entretanto uma versão algébrica deste algoritmo não se mostrou adequada, pelo rápido aumento das zonas de intersecção, tornando necessária a contagem de zeros num maior número de regiões.

A utilização de regiões retangulares por sua vez, apresenta facilidades no processo de bissecção, pois seus componentes conservam a geometria inicial.

4.2 Algoritmo de Pinkert

O algoritmo proposto por Pinkert [PIN 76] foi um dos pioneiros a adotar aritmética exata para o problema de isolar zeros de polinômios complexos. Para o caso de polinômios reais, [HEI 71] foi o precursor desta idéia.

Para isolar raízes o algoritmo proposto usa basicamente o método de determinação do número de zeros num semi-plano arbitrário pelo teorema de Routh, conforme descrito na seção 3.4.

Dado um polinômio P(z)=P1(z)+iP2(z), onde P1 é a parte real e P2 é a parte imaginária, e H = mdc(P1,P2). Se α é raiz de P, então $P1(\alpha)$ e $P2(\alpha)$ são reais, logo α é raiz de H.

Ao se dividir P por H, obtém-se um polinômio sem raízes reais, que será denotado por $\overline{P}$. A este polinômio, pode aplicar-se o **teorema de Routh**, que determina o número de zeros acima e abaixo do semi-plano complexo.

O método requer o conhecimento do número de raízes do polinômio nos bi-quadrantes I,III e II,IV. Estes valores podem ser obtidos usando polinômios cujas raízes são o quadrado das raízes de p(z), junto com os algoritmos anteriores para raízes de um polinômio acima e abaixo do eixo x.

Seja P^* um polinômio cujas raízes são α_j^2, então o número de raízes de P^* no semi-plano superior é igual ao número de raízes de P nos quadrantes I e II mais o número de raízes de P sobre o eixo imaginário positivo. Este último número pode ser calculado usando o teorema de Sturm (teorema 3.3.4).

Para se determinar P^*, o seguinte teorema pode ser utilizado:

Teorema 4.2.1 *Seja P um polinômio de uma variável sobre os números complexos, de grau* $n > 0$. *Seja* $p(x) = \sum_{j=0}^{n}(a_j x^j) = a_n \prod_{j=1}^{n}(x - \alpha_j)$ *com* $\alpha_j \in C$ *para* $1 \leq j \leq n$. *Seja* $P_1(x) = \sum_{j=0,j,par}^{n}(a_j x^j)$, $P_2 = \sum_{j=0,j,impar}^{n}(a_j x^j)$, $B = P_1^2 - P_2^2$ *e* $P^*(x^2) = B(x)$. *Então* $P^*(x) = (-1)^n a_n^2 \prod_{j=1}^{n}(x - \alpha_j^2)$.

(Veja [PIN 76]).

Seja q_i o número de raízes de P no i-ésimo quadrante e a_i o número de raízes sobre o i-ésimo semi-eixo. Pode-se então calcular (q1+q2) e (q1+q3), que equivalem aos quadrantes I e III. Através de uma rotação é possível obter-se $P(x) = P(ix)$, podendo-se assim calcular (q2+q3) e (q2+q4) que equivalem aos quadrantes II e IV. Assim, tendo-se os valores de (q1+q2), (q1+q3), (q2+q3) e (q2+q4) é possível obter cada valor de q_i.

Isto é, através do uso do teorema 4.2.1 e rotação, o número de raízes em cada quadrante pode ser calculado.

O número de raízes num retângulo pode então ser obtido, usando 4 cálculos do número de raízes num quadrante deslocado.

O número de raízes de um quadrante deslocado pode ser calculado através de uma transformação de deslocamento: P(x):P(x+a).

Sucessivas aplicações deste método para determinar o número de raízes num retângulo utilizadas no processo de bissecção para isolar as raízes envolve

cálculos redundantes, conforme observa [COL 77]. Na prática, este algoritmo mostrou-se ineficiente, fazendo necessário o surgimento de soluções melhores.

4.3 O desenvolvimento de um algoritmo exato para isolar zeros polinomiais

Baseado num algoritmo numérico proposto por Wilf, [WIL 78], desenvolveu-se um algoritmo simbólico para isolar zeros polinomiais. Esse algoritmo utiliza sequências de Sturm para determinar o número de zeros no interior de um retângulo, juntamente com um esquema desenhado por Wilf, que reutiliza sequências de Sturm. Um estudo detalhado esclarecendo os diversos passos do método básico é feito previamente e as modificações sugeridas são apresentadas posteriormente.

Seja $p(z)$ um polinômio complexo e R um retângulo no plano complexo, com vértices Q1, Q2, Q3 e Q4, conforme a figura 4.1 . O procedimento usado para enumerar os zeros de p(z) dentro de R é baseado na aplicação da fórmula 3.8. No decorrer do texto, quando houver referência a quadrado, está se referindo a um caso particular de retângulo.

Dispondo de um retângulo inicial, 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.

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

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

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

iii) Gera-se a Sequência de Sturm associada a lateral j, iniciando com $f_1 = \alpha_j$ e $f_2 = \beta_j$

A região inicial proposta pelo autor é de um quadrado com centro na origem com comprimento de lateral aleatório. Calcula-se o número de zeros neste quadrado pela fórmula 3.8 e se o valor obtido for menor que o grau do polinômio, dobra-se o comprimento da lateral e repete-se o processo até que se consiga um quadrado que contenha todas as raízes.

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 4.1 mostra o retângulo bisseccionado, com os 4 subretângulos (I, II, III e IV), e os 4 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 denotados 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).

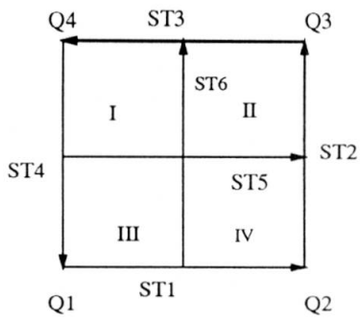

Figura 4.1: O retângulo particionado e as Sequências de Sturm

De acordo com a numeração de subretângulos da figura, pode-se verificar que para cada subretângulo, ao ser percorrido no sentido anti-horário, 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, ST4,

para II, usa-se ST5, ST2, ST3, ST6,

para III, usa-se ST1, ST6, ST5, ST4,

para IV, usa-se ST1, ST2, ST5, ST6.

A determinação do número de zeros para cada subretângulo I, II, III e IV é feita pela utilização da fórmula 3.8, 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.

A título de ilustração, será apresentado os intervalos considerados para o primeiro nível de subdivisão, conforme tabela 4.1. (Estes intervalos mudam conforme vai-se procedendo às sucessivas subdivisões).

Nesta tabela, L é o tamanho do lado do retângulo que está sendo pesquisado.

Tabela 4.1: Intervalos considerados para o primeiro nível de subdivisão, usando sequências de Sturm

lateral	*Subretangulos*			
	I	II	III	IV
1	[-L,0]	[0,L]	[0,L]	[L, 2L]
2	[0, L]	[L, 2L]	[-L,0]	[0, L]
3	[L, 2L]	[0,L]	[0,-L]	[L,0]
4	[0, L]	[L, 0]	[L, 2L]	[0, -L]

4.3.1 A presença de zeros na fronteira do retângulo

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 a aplicação da fórmula 3.8 .

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 3.8 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.

Uma outra estratégia é adotada no algoritmo desenvolvido neste trabalho para solucionar este tipo de problema, garantindo a exatidão dos resultados obtidos.

Seja $p(z)$ um polinômio de coeficientes complexos e livre de quadrados.

Através da determinação de uma quota superior para o módulo das raízes de um polinômio (por exemplo 4.1), 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.

O método usado para contar o número de zeros dentro de um retângulo foi aquele descrito na seção 3.3.

Com relação ao problema da presença ou não de zeros nas fronteiras dos retângulos pesquisados, o tratamento deve ser diferenciado para os diferentes casos. A fórmula 3.8 descrita 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 (3.3.4), 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 3.8, 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 3.8 fica então modificada para

$$N = \frac{1}{2}\left(\sum_{j=1}^{4}\left(V(l_j) - V(0)\right) - \sum_{j=1}^{4}(b_j)\right) \tag{4.3}$$

4.3.2 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.

4.3.3 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 subalgoritmos mais apropriados 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 quadrado I e 2 no quadrado II,) 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. Aqui, cabe lembrar que a referência a coeficientes

reais diz respeito a números sem a parte imaginária (na verdade os coeficientes são racionais).

- 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 [BAR 89], 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. Uma alternativa que pode ser explorada posteriormente é o algoritmo proposto por Ben-Or [BEN 90], que usa um tipo de sequência de Sturm normalizada, ou ainda o algoritmo proposto por Akritas [AKR 80], que usa uma modificação do teorema de Vincent.

- 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 esta condição é apresentado em [TRE 90]. 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 complexas, estas ocorrem em pares conjugados. Então a pesquisa só precisa ser realizada na parte superior do eixo x.

- Para verificar 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. [DAV 88].

  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, é possível identificar quantos zeros estão sobre o eixo x. Se os coeficientes são 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.

4.4 O algoritmo de Uspensky modificado

Uspensky [USP 48] propôs um método baseado na regra de troca de sinais dos coeficientes para isolar zeros reais.

Este método é usado como subalgoritmo do método baseado nas trocas de quadrantes (ver seção seguinte).

Dado um polinômio $p(x) = a_0 + a_1x + a_2x^2 + ... + a_nx^n$, com $a_i \in R$, denota-se $var(p)$ à variação dos sinais ao longo da sequência de $a_i's$ (i=0...n). Pela regra de Descartes, $var(p)$ excede o número de raízes reais positivas de um polinômio (contando suas multiplicidades) e é dado por um inteiro par não negativo.

O método de Uspensky baseia-se em dois casos particulares da Regra de Descartes. Se $var(p) = 0$ então p não tem zeros reais positivos e se var (p) = 1 então p tem exatamente um zero positivo. Quando $var(p) > 1$, executa-se um procedimento recursivo até que var(p) $\leq$ 1).

As raízes positivas de p podem estar no intervalo $[0, \infty)$. Este intervalo é considerado originariamente em 2 partes $(0, 1)$ e $(1, \infty)$.

Os zeros de $p(x)$ em (0,1) equivalem aos zeros positivos de

$$p^*(x) = (x+1)^n p(\frac{1}{(x+1)}) \tag{4.4}$$

Esta transformação mapeia o círculo unitário no semiplano direito $R(z) \geq 0$.

Os zeros de p(x) em (1,∞) equivalem aos zeros positivos de

$$p^{**}(x) = p(1+x) \tag{4.5}$$

A regra de Descartes é então aplicada a p^* e p^{**}. Para cada polinômio transformado (p^*, p^{**}), se $var(p^*) > 1$ ou $var(p^{**}) > 1$, as transformações (4.4 e 4.5) são aplicadas recursivamente, até que $var(p^*) \leq 1$ e $var(p^{**}) \leq 1$. O término do

algoritmo é garantido desde que o polinômio $p(x)$ tenha somente raízes simples, pelo teorema de Vincent (veja [USP 48]).

Collins [COL 82] propôs uma modificação para o referido algoritmo:

Dado um polinômio p, (real ou complexo), determina-se uma quota superior para o módulo das raízes de p conforme (4.2).

Como esta quota é uma potência de 2 ($\overline{b} = 2^k$), tem-se que o número de raízes de p entre $(0, \overline{b})$ é equivalente ao número de raízes de $\overline{p}$ entre (0,1), onde :

$$\overline{p}(x) = \begin{cases} p(2^k x) & se\ k \geq 0 \\ 2^{-kn} p(2^k x) & se\ k < 0 \end{cases} \tag{4.6}$$

Desta forma, é suficiente contar as raízes de $\overline{p}(x)$ no intervalo (0,1).

Se var($\overline{p}^*$) > 1, o intervalo (0, 1) é bisseccionado e aplica-se a regra de Descartes aos 2 polinômios transformados.

Quando o intervalo (0, 1) for dividido ao meio, tem-se os subintervalos (0, 1/2) e (1/2, 1). Quando se mapeia o subintervalo (0, 1/2) para (0, 1) tem-se que $\overline{b} = 2^{-1}$. Assim, a transformação usada é $p' : \overline{p}(x) = 2^n * p(\frac{x}{2})$. Para o intervalo (1/2, 1), a transformação deve ser $p'' : \overline{p}(x+1)$. Sucessivas bissecções dos subintervalos de (0, 1) são efetuadas, através das transformações p' e p''.

A cada bissecção, o subintervalo da esquerda deve ser mapeado para (0,1), pela aplicação da transformação (4.4) e o subintervalo da direita para $(1, \infty)$, pela aplicação de 4.5.

Depois do algoritmo ter isolado as raízes positivas, obtém-se $\overline{p}(-x)$, e o mesmo procedimento é aplicado para isolar as raízes negativas.

As seguintes funções descrevem de uma forma mais concisa o algoritmo:

Entrada: P, um polinômio primitivo livre de quadrados.

Saída: L, uma lista de intervalos isolados para os zeros reais positivos de P.

(1) Cálculo da quota superior (conforme 4.2)). Seja b esta quota calculada.

(2) Aplica transformação 4.6

(3) Isola($\overline{p}$,L') Isola é um subalgoritmo que isola todos os zeros reais de $\overline{p}$ no intervalo (0,1). O resultado retorna em L'

(4) Troca todo intervalo (a_i, b_i) de L' por $(2^k a_i, 2^k b_i)$ e coloque o resultado em L.

(5) Repita o processo para as raízes negativas, considerando p(-x), e a saída em L2.

(6) Retorne (concatena(L,L2)).

Subalgoritmo Isola

Entrada: P, um polinômio livre de quadrados integral.

Saída: L, uma lista de zeros de P no intervalo (0,1)

(1) Aplica T1 em P :$P^* = (x+1)^n * P(1/(x+1))$ Os zeros de P em (0, 1) são transformados nos zeros de p^* em $(0, \infty)$

(2) Se var(P^*) = 0 então retorne {L=()}

(3) Se var(P^*)=1 então retorne {L=((0,1))}

(4) Bisseciona o intervalo

(4.2) Calcula ponto médio. Se P(1/2)=0 então inclui(1/2,1/2) em L e reduz o grau de P, substituindo P por P/(2*x-1).

(4.3) Metade esquerda: $P' = 2^n P(x/2)$ Os zeros de P em (0,1/2) são transformados nos zeros de P' em (0,1).

Isola(P',L').

Troca todo intervalo (a'_i, b'_i) em L' por $(a'_i/2, b'_i/2)$ em L.

(4.4) Metade direita: $P'' = \overline{P}(x+1)$ os zeros de P em (1/2,1) são transformados nos zeros de P' em (0, 1)

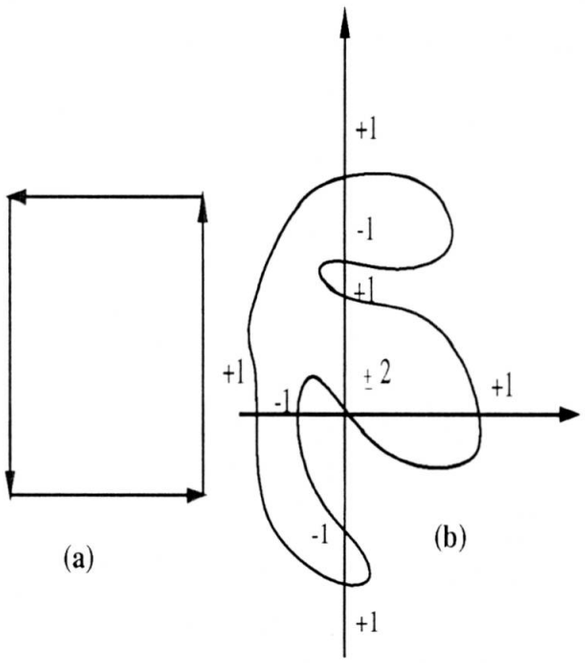

Figura 4.2: Algoritmo de Collins-Krandick

Isola(P",L"),

Troca todo intervalo (a_i'', b_i'') em L" por $((a_i'' + 1)/2, (b_i'' + 1/2)$ em L.

(4.5) Retorne L.

4.5 O Algoritmo de Collins-Krandick

Considerando o princípio do argumento (3.3.1), o número N de raízes dentro de R é igual ao número de vezes que a curva $p(\delta R)$ passa ao redor da origem. Particularmente aqui, R é um retângulo.

Uma outra forma de contar as mudanças em $argp(z)$ quando z percorre δR é pela determinação da sequência de quadrantes percorridos por $p(z)$ quando z percorre δR. Partindo-se de um ponto $z_0 \in \delta R$, verifica-se as trocas de quadrantes de $p(z)$ quando z percorre δR [COL 92].

A figura 4.2 ilustra como é realizado tal procedimento:

Em (b) tem-se a imagem de p para os pontos z $\in \delta R$.

As mudanças de quadrante ocorrem sempre em pontos do eixo real x e em pontos do eixo imaginário y. (Tais pontos são chamados pontos críticos).

Quando a imagem de uma lateral do retângulo fica totalmente sobre um eixo, tem-se uma linha axial.

A determinação do número de voltas que a imagem de $p(z)$ dá ao redor da origem pode ser feita por se acumular as mudanças de $\arg p(z)$ de $p(\delta R)$ em unidades de $\pi/4$ (um quadrante) e dividido por 2π (uma volta completa).

Considerando que z percorre δR num sentido antihorário, as mudanças de quadrante de $p(z)$ no mesmo sentido são computadas positivamente e as mudanças de quadrante no sentido horário são computadas negativamente.

Se a curva da imagem p(z) passar pela origem, significa que existe zero na fronteira de δR. Neste caso, tem-se uma ambiguidade quanto a mudança de argumento. É preciso então decidir o caminho que se irá percorrer. Se a curva da imagem tiver um trecho que passa do terceiro quadrante para o primeiro quadrante, existem 2 caminhos possíveis: no sentido horário (III, II, I), ou no sentido antihorário (III, IV e I). Isto pode ser traduzido da seguinte forma: o caminho no sentido antihorário é para o caso da raiz de uma fronteira que deve ser considerada na contagem do total de zeros dentro do retângulo, e o caminho no sentido horário é para o caso da raiz de uma fronteira não ser considerada.

4.5.1 A contagem das trocas de quadrantes

Para que se possa determinar as mudanças de quadrantes é preciso identificar como p se comporta ao redor dos pontos críticos. Por questões de simplicidade, considerou-se a seguinte equivalência: Uma volta completa (2π) é composta por 4 trocas de quadrantes. Assim, 1 troca de quadrante equivale a $\frac{\pi}{2}$.

Considerando inicialmente os casos sem ambiguidade, tem-se:

- Se um ponto crítico toca um dos eixos (não na origem) sem cruzá-lo, conta-se 0 na mudança de argumento. (Não houve troca de quadrantes).

- Se a troca de quadrantes ocorre entre 2 quadrantes adjacentes cruzando um dos eixos (não na origem), conta-se 1 (no sentido antihorário) e -1 (no sentido horário).

- Se ocorrer a troca de um quadrante para uma linha axial (não na origem), conta-se 1/2 (no sentido antihorário) e -1/2 (no sentido horário).

- Para um ponto crítico que é um canto do retângulo, cujas 2 linhas incidentes são linhas axiais que não passam pela origem, conta-se 0.

Trocas de quadrante que geram ambiguidades (passam pela origem).

Pelo princípio do Argumento, o número de mudanças de argumento obtida para o caso de se contar a raiz na fronteira é maior por 2π do que a troca de argumento obtida sem contar a raiz na fronteira. Nos possíveis casos citados abaixo, o primeiro valor diz respeito ao caso de se contar a raiz da fronteira e o segundo diz respeito ao caso de não se contar a respectiva raiz.

- Troca de um quadrante para outro adjacente, cruzando a origem, conta-se +1 ou -3.

- Troca de um quadrante para outro oposto diagonalmente, cruzando a origem, conta-se +2 ou -2.

- Troca de um semi-eixo negativo para o mesmo semi-eixo positivo, conta-se +2 ou -2.

- Troca de um semi-eixo para outro semi-eixo, (passando pela origem e virando a esquerda), conta-se -1 ou +3.

- Troca de um semi-eixo para outro semi-eixo, (passando pela origem e virando a direita), conta-se +1 ou -3.

- Troca de um quadrante para um semi-eixo adjacente (passando pela origem), num sentido antihorário, conta-se 4,5 ou 0,5. Se a troca for no sentido oposto, conta-se +3.5 ou -0.5.

- Troca de um quadrante para um semi-eixo adjacente (passando pela origem), num sentido horário, conta-se +3.5 ou -0.5. Se a troca for no sentido oposto, conta-se 4.5 ou 0.5.

- Troca de um quadrante para o primeiro semi-eixo não adjacente, indo-se num sentido antihorário, conta-se 1.5 ou -2.5. Se a troca for no sentido oposto, conta-se 2.5 ou -1.5.

- Troca de um quadrante para o segundo semi-eixo não adjacente, indo-se num sentido antihorário, conta-se 2.5 ou -1.5. Se a troca for no sentido oposto, conta-se 1.5 ou -2.5.

O número total de zeros é dado então por se acumular estes valores obtidos para cada ponto crítico, dividido por 4. A forma de se obter estes pontos críticos é através do método para isolar raízes reais (Uspensky modificado).

Tendo estabelecido o processo de enumeração em retângulos arbitrários, o processo de isolamento fica então da seguinte forma:

A partir da determinação de uma quota superior **b** (4.2) para o módulo das raízes, gera-se um quadrado inicial com centro na origem e lateral = **2b**, contendo todos os zeros. Usando o método para enumerar raízes descrito anteriormente, todas as raízes de um polinômio podem ser isoladas por bisseccionar-se as regiões alternadamente na vertical e na horizontal [COL 92]. Quadrados são particionados por um linha vertical. Retângulos são particionados por uma linha horizontal. Para cada região subdividida, conta-se o número de raízes para as 2 metades. Regiões que não contém zeros são descartadas. Retângulos com um zero e laterais menor que um mínimo estipulado são incluídas numa lista de regiões isoladas. Regiões com

mais de um zero devem ser bisseccionadas até que se obtenha retângulos isolados do tamanho desejado.

Dado um polinômio p(z), obtém-se a representação real do mesmo:

$$P(z) = P(x + iy) = P1(x, y) + iP2(x, y)$$

Considerando R um retângulo qualquer, estabeleceu-se o canto inferior esquerdo como ponto de partida. Na verdade poderia ser qualquer outro vértice do retângulo. Esta determinação foi feita por questões de melhor clareza no entendimento do algoritmo. Partindo deste ponto e percorrendo o retângulo no sentido antihorário, as respectivas laterais serão denominadas por L1, L2, L3 e L4.

Para cada lateral, executa-se os seguintes passos:

- **Especialização**

  Seja (a_0, a_1) os pontos que delimitam uma lateral de um retângulo. Se lateral = L1 ou L3, tem-se que a coordenada de y é fixa. Assim, $\overline{P_1}(x) = P_1(x, s)$, $\overline{P_2} = P_2(x, s)$, onde s é a coordenada imaginária da lateral.

  Se lateral = L2 ou L4, tem-se que a coordenada de x é fixa. Assim, $\overline{P_1}(y) = P_1(s, y)$, $\overline{P_2}(y) = P_2(s, y)$, onde s é a coordenada real de a_0 da respectiva lateral.

- **Localização dos pontos críticos**

  Os pontos críticos são determinados por se localizar as raízes de $\overline{P_1}$ e $\overline{P_2}$.

  Os polinômios $\overline{P_1}$ e $\overline{P_2}$ são fatorados de forma livre de quadrados. Assim, para cada fator, sabe-se que as raízes são simples e o método da variação dos sinais pode ser usado. (Uspensky modificado).

Aos intervalos resultantes da aplicação deste subalgoritmo devem ser associados os respectivos polinômios. Intervalos isolados que se referem a diferentes polinômios devem ser refinados até que se tornem disjuntos.

Os intervalos associados a $\overline{P_1}$ e $\overline{P_2}$ são então ordenados em ordem crescente para as laterais L1 e L2, e em ordem decrescente para as laterais L3 e L4. Isto é feito para manter o percurso no sentido antihorário.

- **Classificação dos pontos críticos**

No processo de determinação dos pontos críticos, obtém-se na verdade intervalos isolados, nos quais estão contidos os pontos críticos. Eventualmente estes intervalos podem ser um ponto, tendo-se neste caso efetivamente um ponto crítico.

A classificação de um ponto crítico consiste em determinar o comportamento da imagem de $p(z)$ nas vizinhanças deste ponto crítico.

Se nenhum dos polinômios é identicamente nulo, e se o ponto crítico não for um canto do retângulo, as mudanças de quadrantes podem ser obtidas pela simples verificação das multiplicidades obtidas no processo de fatoração livre de quadrados para cada polinômio especializado.

Quando não existe zero na fronteira, a contagem procede-se da seguinte forma:

Se multiplicidade par: conta-se 0. Se multiplicidade ímpar, conta-se 1 .

Quando existem zeros na fronteira, as ambiguidades são identificadas quando um intervalo isolado pertence a $\overline{P_1}$ e $\overline{P_2}$ simultaneamente. As diferentes contagens serão expressas em primeiro lugar para o caso de contar os zeros na fronteira e em segundo lugar para o caso de não contar os zeros na fronteira.

Se multiplicidade par :conta-se +1 ou -3.

Se multiplicidade ímpar : conta-se +2 ou -2.

Para os casos em que um dos polinômios é identicamente nulo, ou o ponto crítico é um ponto, verifica-se as trocas de quadrantes, de acordo com os casos identificados em 4.5.1.

Para identificar se a contagem é feita no sentido horário ou antihorário, define-se o quadrante inicial, através da avaliação de $\overline{P_1}$ e $\overline{P_2}$ em q1. A partir daí, armazena-se os quadrantes percorridos, por se verificar o quadrante atual e determinar o quadrante seguinte. Isto pode ser feito, considerando-se que, se o intervalo isolado pertence ao polinômio $\overline{P_1}$, então ocorrerá um cruzamento no eixo y, e se pertence ao $\overline{P_2}$, o cruzamento será no eixo x.

Tendo determinado o número de zeros de um retângulo, se este for menor ou igual a 1, encerra-se a busca neste retângulo. Caso contrário, o mesmo deve ser bisseccionado e os passos acima devem ser repetidos.

Os zeros que ficam nas fronteiras devem ser contados somente em um retângulo. Tal condição pode ser garantida por se contar os zeros somente nas linhas superior e direita do retângulo, e somente o canto incidente destas duas linhas de cada retângulo.

Para cada bissecção realizada, somente para esta nova linha divisória vai ser necessário a determinação dos pontos críticos.

Além disto, para cada lateral bisseccionada é preciso verificar se o ponto de intersecção resultante não está contido em algum intervalo isolado para algum dos polinômios associados àquela lateral.

Para que se percorra os quadrantes sempre no sentido antihorário deve-se cuidar para que a ordenação dos intervalos isolados seja feita, observando-se que para subretângulo do mesmo nível de pesquisa, uma lateral comum é percorrida em sentidos opostos, dependendo do subretângulo que se está pesquisando.

5 ANÁLISE COMPUTACIONAL DOS ALGORITMOS

Os algoritmos que são analisados neste capítulo são basicamente os seguintes: o algoritmo desenvolvido neste trabalho (Wilf modificado) e o algoritmo de Collins-Krandick (usando método da troca de quadrantes).

5.1 Complexidade Teórica dos Algoritmos

Nesta seção são introduzidas algumas noções de complexidade, bem como é estabelecida a forma pela qual foram determinadas as complexidades dos algoritmos analisados.

Definição 5.1.1 *Seja f, g funções de valor real definidas sobre um domínio comum D. Então pode-se dizer que:*

- *f é dominada por g e escreve-se f=O(g) no caso de existir um número real* c_1 *e* x_1 *em D, tal que* $|f(x)| \leq c_1|g(x)|$ *para todo* $x \in D$, $x > x_1$.
- *f domina g e escreve-se* $f = \Omega(g)$ *no caso de exitir um número real positivo* c_2 *e* $x_2 \in D$, *tal que* $|f(x)| \geq c_2|g(x)|$ *para todo* $x \in D$, $x > x_2$.
- *f e g são codominantes e escreve-se f g se acontece de f=O(g)=*$\Omega(g)$.

Os algoritmos aqui analisados são baseados em sucessivos particionamentos de um retângulo inicial, até que se obtenha retângulos isolados de um polinômio livre de quadrados.

Definição 5.1.2 *Seja k um número inteiro, e* $L_\beta(k)$, *o comprimento de k, pelo número de dígitos de k, em sua representação relativa à base* β. *Se* $\lfloor x \rfloor$ *é a função*

que retorna o menor inteiro maior ou igual a x e $\lceil x \rceil$ *é a função que retorna o maior inteiro menor ou igual a x então:*

$$L_\beta(k) = \begin{cases} 1 & se\ k = 0 \\ \lceil log_\beta(|k|+1) \rceil & caso\ contrário \end{cases} \quad isto\ é$$

$$L_\beta(k) = \begin{cases} 1 & se\ k = 0 \\ \lfloor log_\beta |k| \rfloor + 1 & caso\ contrário \end{cases}$$

Definição 5.1.3 *Seja* $p(z) = \sum_{i=0}^{n} a_i z^i$ *um polinômio de uma variável com coeficientes inteiros. (Se os coeficientes são racionais, primeiro transforma-se para inteiros). A quase-norma máxima de p(z) é dada por* $|p|_\infty = \max(|a_i|)$ *e a sub-norma é* $|p|_1 = \sum_{i=0}^{n} |a_i|$.

Assim $|p|_\infty \leq |p|_1 \leq (n+1)|p|_\infty$ onde n é o grau de P. Portanto $L(|p|_\infty) \sim L(|p|_1)$ são codominantes.

Limites de tempos computacionais para operações sobre polinômios são comumente obtidos como funções dos graus e normas dos polinômios.

Se R é um retângulo isolado para uma raiz α de um fator livre de quadrados S_i, e S é um retângulo isolado para uma raiz β de um fator livre de quadrados S_j, $i \neq j$, então o número máximo de vezes que R e S devem ser bisseccionados antes que eles se tornem disjuntos vai depender da distância entre as raízes α e β.

Definição 5.1.4 *Seja p(z) um polinômio com coeficientes inteiros. Supondo que o grau de* $p(z) = n \geq 1$ *e que ele tem* $k \leq n$ *raízes distintas* $\alpha_1, \alpha_2, ...\alpha_k$. *Se* $k \geq 2$, *define-se a separação mínima de raízes de P, por:*

$$sep(p) = \min_{1 \leq i < j \leq k} |\alpha_i - \alpha_j|$$

Teorema 5.1.1 *Se $P(z)$ é um polinômio com coeficientes inteiros e de grau $n \geq 2$, com $d = |p|_\infty$ então*

$$sep(p) > 1/2(e^{1/2}n^{3/2}.d)^{-n}$$

[COL 82]

Outras estimativas para sep(p) podem ser encontradas em [RUM 79].

A partir deste teorema, pode-se concluir que:

$$|log(sep(p)^{-1})| = O(nL(nd))$$

5.1.1 Complexidade do algoritmo de Wilf modificado

Para estabelecer a complexidade para o pior caso do algoritmo proposto, deve-se considerar as complexidades dos subalgoritmos usados. Considerando-se que o maior esforço dispendido está na fase de enumeração, obteve-se inicialmente a medida de complexidade para esta fase.

O tempo total para se enumerar zeros de um retângulo é composto das seguintes partes: tempo para gerar uma sequência de Sturm, e tempo para se fazer as avaliações dos termos da sequência nos respectivos intervalos.

O tempo máximo para se gerar uma sequência de Sturm é dado por

$$O(n^4 L(d)^2).$$

Veja [COL 82].

Tempo máximo para se avaliar um polinômio em um ponto racional é dado por

$$O(n^4 L(nd)^4).$$

Veja [COL 82]

Como na sequência de Sturm, os graus dos polinômios são diferentes, considerar-se -á d_0, a quase-norma máxima do polinômio de grau 1 , e d_n a quase-norma máxima de p(z).

Considerando-se que uma sequência de Sturm tem no máximo $n + 1$ termos, o custo de se avaliar uma sequência de Sturm é dado pode ser dado por

$$O(n + 1\ (n^4 L(nd_0)^4))$$

Assim, o tempo total para a enumeração é dado por:

$$O(n^5 L(nd_0)^4) + O(n^4 L(d_n)^2)$$

que pode ser majorado por

$$O(n^5 L(nd_0)^4).$$

pois $L(nd_0) > L(d_n)$

Considerando-se a quota superior definida em 4.1, $b = |\alpha| \leq 2 \max_{1 \leq i \leq n} |a_{n-i}/a_n|^{1/i}$, a distância inicial entre os extremos de um segmento que compõe o retângulo é dado por 2b. Considerando-se d como $|p|_\infty$, tem-se que:

$$b \leq 2d \text{ e}$$

$$2b \leq 4d.$$

Após k particionamentos, a distância entre os extremos é dada por $b.2^{-k}$.

O número máximo de particionamentos pode então ser estabelecido pela seguinte relação:

$$\frac{4d}{sep(p)} = \frac{2b}{b.2^{-k}} =$$

$$k = log_2(2d/sep(p))$$

Como $log(sep(p)^{-1})$ é da ordem O(nL(nd)), tem-se que o número k de particionamentos é limitado por O(nL(nd)).

Assim, o tempo total para o isolamento é dado por:

$$O(nL(nd_n)).(O(n^5L(nd_0)^4) \text{ ou seja}$$

$$O(n^6L(nd_0)^5)$$

5.1.2 Complexidade do algoritmo de Collins-Krandick

O algoritmo de Collins baseado na troca de quadrantes tem como subalgoritmo básico o método de Uspensky modificado, cujo tempo máximo é dado por:

$$O(n^6L(d)^2)$$

(Veja [COL 82]).

Depois das raízes reais isoladas, há a necessidade de refinar os intervalos não disjuntos que pertencem à polinômios diferentes. O pior caso pode acontecer em duas situações: primeiro, quando se tem 2 intervalos iguais tão grandes quanto o intervalo inicial,que vão sendo refinados (através da bissecção dos intervalos) sucessivamente e continuam iguais indefinidamente. Segundo: Quando todos os intervalos são iguais e precisam ser refinados. Para que uma raiz seja considerada ambígua (raiz comum dos dois polinômios reais) é preciso que se tenha na verdade um intervalo pontual. Para se estabelecer um número finito de refinamentos, deve-se fazer tantas bissecções quantas forem necessárias até que a distância entre os extremos do intervalo seja suficientemente pequena ($< \epsilon$), que se possa assumir que o intervalo é um ponto.

Usando a quota superior de módulo de raízes $\overline{b}$ proposto por Collins (4.2), tem-se que qualquer intervalo resultante vai ter a distância entre os extremos sendo um número que é uma potência de 2.

A distância entre os extremos após k bissecções é dada por $\overline{b}.2^{-k}$.

Inicialmente têm-se que a distância inicial entre os extremos de um segmento que compõe o retângulo é dada por $2\overline{b}$.

$$2\overline{b} \leq 8d'$$

onde d' é a norma máxima dos polinômios especializados.

O número máximo de bissecções pode então ser estabelecido da seguinte relação:

$$\frac{8d'}{\epsilon} = \frac{2\overline{b}}{\overline{b}.2^{-k}}$$

da qual obtém-se o valor de k, dado por:

$$k = log_2(8d'/\epsilon)$$

Como no processo de bissecção sempre se despreza uma metade, tem-se então que o número de bissecções é limitado por

$$k = \lceil log_2(8d'/\frac{1}{2}\epsilon)\rceil$$

Este valor também é válido para a segunda situação (todos os intervalos necessitam ser refinados), pois neste caso, tem-se

$$n.\frac{8d'}{n\epsilon}$$

que recai no mesmo caso anterior.

O tamanho de ϵ pode grosseiramente ser estimado como sep(p). Considerando ainda d' a quase norma máxima dos polinômios especializados e $d' \approx d$, pode-se dizer que o número de bissecções é então limitado por $O(n(L(nd))$.

Cada refinamento requer uma bissecção e consequente avaliação dos polinômios no ponto médio do intervalo bisseccionado. Assim, como cada avaliação custa no máximo $O(n^4L(nd)^2)$, e tem-se no máximo k bissecções, tem-se um custo total de refinamento de $O(nL(nd).n^4L(nd)^2)$.

Assim, o custo total máximo para enumerar raízes é dado por:

$$O(n^6L(d)^2) + O(n^5L(nd)^3) \text{ ou seja}$$

$$0(n^6 L(nd)^3)$$

Considerando que o número de particionamentos necessários para se obter as raízes isoladas é dado por $O(nL(nd))$, tem-se que o custo total máximo para isolar as raízes é dado por:

$$O(nL(nd).O(n^6 L(nd)^3) =$$

$$O(n^7 L(nd)^4)$$

5.2 Avaliação experimental

5.2.1 Sobre a implementação dos algoritmos

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 desempenhos dos algoritmos desenvolvidos.

5.2.2 Seleção dos polinômios para teste

Os polinômios utilizados para testar a funcionalidade dos algoritmos foram obtidos da seguinte forma:

- um conjunto de polinômios com coeficientes gerados aleatoriamente.
- Uma coletânea de polinômios usados em trabalhos realizados nesta área de pesquisa. (Consulte [PET 91a], [PET 91b], [NOD 91], [SAK 91], [HOE 92], [GEO 90], [GHA 90], [RUM 83] 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 com 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 (números complexos com componentes cujos componentes são inteiros), 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 (showtime), 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 a qual já existem algoritmos eficientes para tal fim.

5.2.3 Desempenho dos algoritmos testados

Os algoritmos testados experimentalmente foram separados em duas categorias: uma para isolar raízes reais e outra para isolar raízes complexas (coeficientes reais e complexos). Apesar de não ser objetivo direto deste trabalho a localização de raízes reais, uma breve análise de desempenho de 2 algoritmos é realizada. Isto justifica-se pelo fato de que para certas classes de polinômios (só com raízes reais, por exemplo), um algoritmo específico para este caso pode ser mais eficiente do que um algoritmo mais genérico. Foram testados o algoritmo de Uspensky modificado [COL 82] descrito na seção 4.4 e o algoritmo usando teorema de Sturm com sequência primitiva [HEI 71].

Os polinômios utilizados para testar estes algoritmos foram gerados a partir de raízes aleatórias (todas reais).

Os tempos observados para os dois algoritmos podem ser vistos na tabela 5.1

A figura 5.1 mostra a variação do tempo de execução dos dois algoritmos (Sturm e Uspensky modificado) para isolar raízes reais, de acordo com a variação do grau do polinômio.

Tabela 5.1: Tempos observados para isolar todas raízes reais para Sturm e Uspensky modificado

grau	Sturm	Uspensky
2	0,57	0,53
3	0,72	1,04
4	1,32	2,08
5	3,4	6,5
6	3,1	5,6
7	8,88	15,07
9	15,24	24,68
10	24,48	43
11	33,6	43
12	42,48	75,28
13	60	70
14	79,38	97,55
15	125,28	128,28
16	143,23	144.18
17	177,6	170,43
18	269,88	198,71
19	275,55	219,88
20	319,23	286.77
21	507,88	391,88

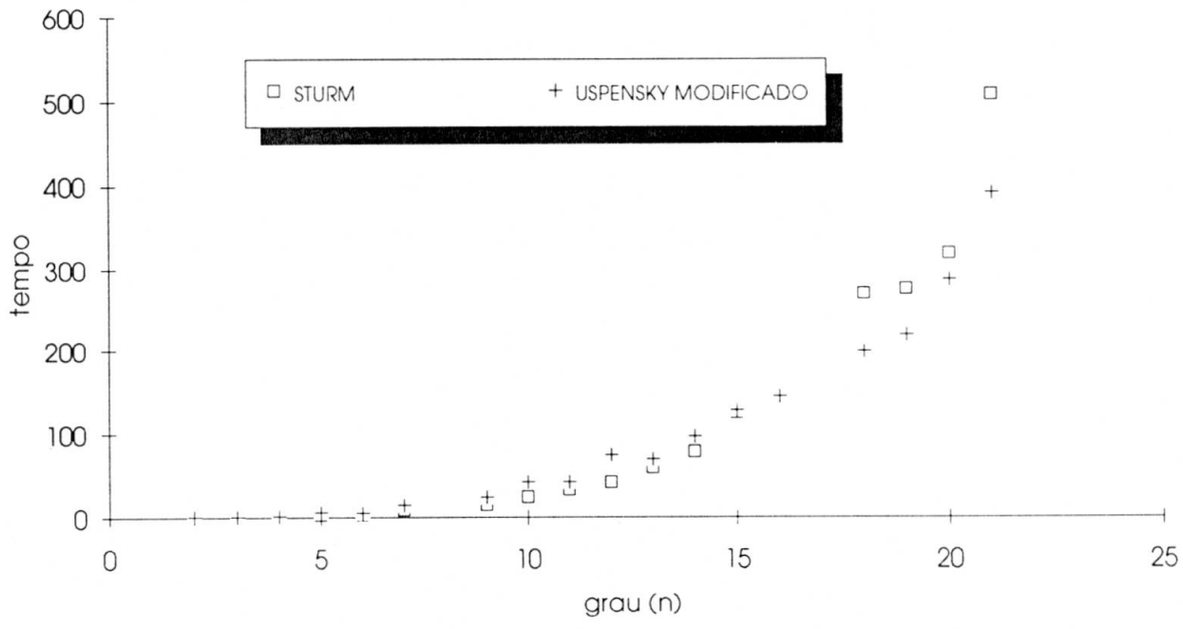

Figura 5.1: Variação dos tempos de execução para isolar todos os zeros reais (Teorema de Sturm) e Uspensky modificado

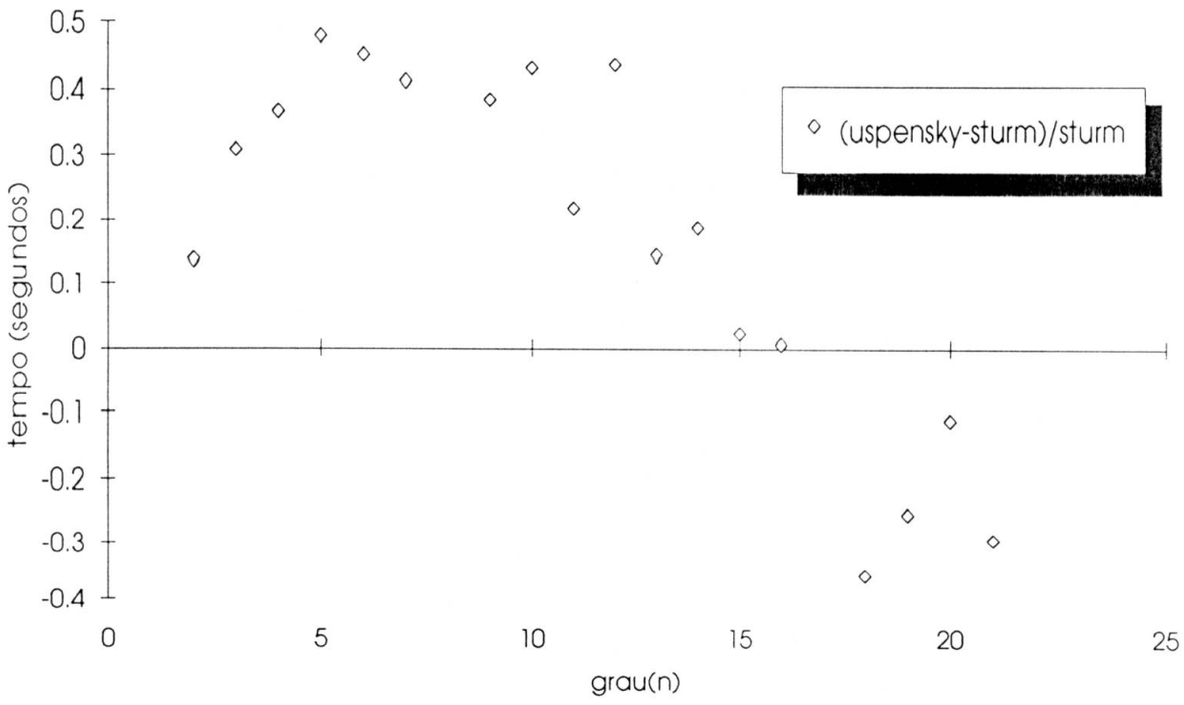

Figura 5.2: Diferença relativa entre os tempos para isolar zeros reais (Uspensky modificado e Sturm

Observando a tabela 5.1, verifica-se que a partir de grau 17 o algoritmo de Uspensky modificado mostra-se mais rápido do que o de Sturm. A figura 5.2 mostra como varia a diferença relativa entre os tempos observados para Uspensky modificado e Sturm.

Para isolar raízes complexas foram testados o algoritmo de Collins-Krandick [COL 92] descrito na seção 4.5 e o algoritmo desenvolvido neste trabalho (Wilf modificado) descrito na seção 4.3. A escolha foi feita com o objetivo de comparar os resultados do algoritmo proposto com os de um trabalho recente na literatura.

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. Isto também favorece uma comparação com as medidas teóricas de complexidade estabelecidas para os algoritmos testados.

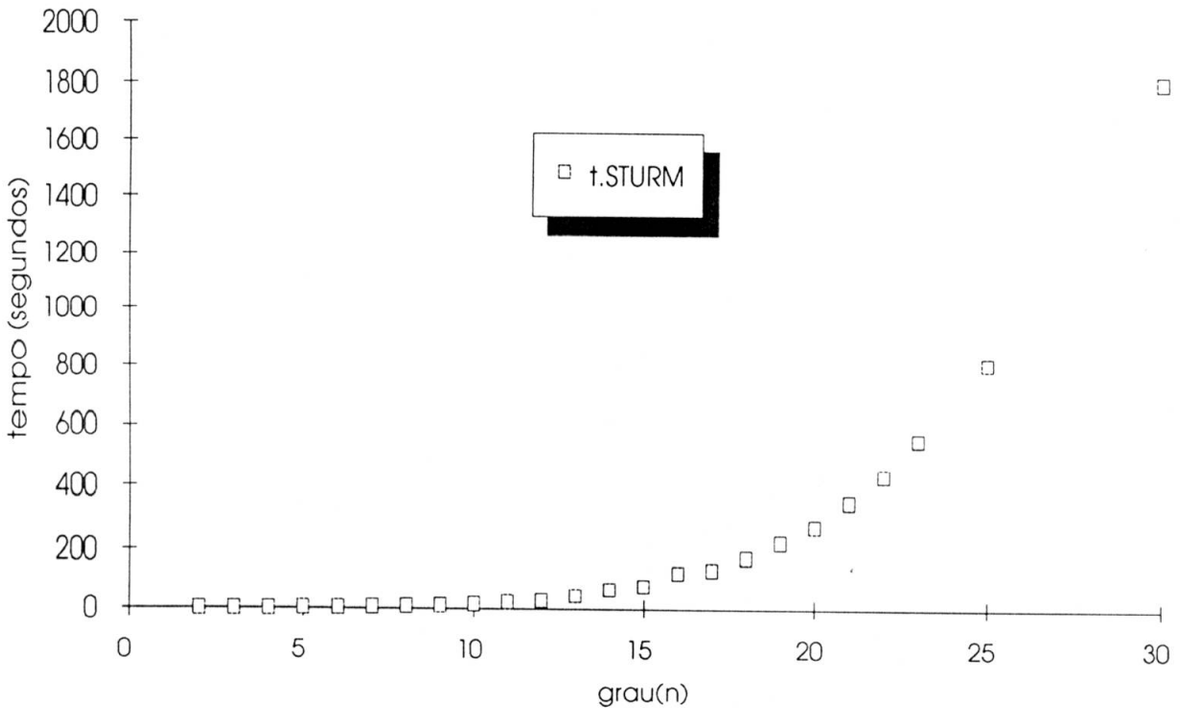

Figura 5.3: Tempos de execução para enumerar zeros de polinômios reais num retângulo usando o algoritmo proposto

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 baseado na troca de quadrantes terá seus tempos observados referenciados por t.Collins. Os dados utilizados para a construção dos gráficos seguintes constam na tabela 5.2.

A figura 5.3 mostra a variação do tempo de execução do algoritmo proposto na fase de enumeração, de acordo com a variação do grau do polinômio, para o caso de coeficientes reais.

A figura 5.4 mostra a variação do tempo de execução do algoritmo de Collins na fase de enumeração de acordo com a variação do grau do polinômio de coeficientes reais.

A dependência do tempo em relação ao grau do polinômio mostra-se similar para os dois algoritmos no caso de coeficientes reais (até grau 25 aproxima-

Tabela 5.2: Tempos observados para enumerar zeros em retângulos para o algoritmo proposto e o algoritmo de Collins, e a diferença relativa entre eles

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	11,02	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	-	-	-

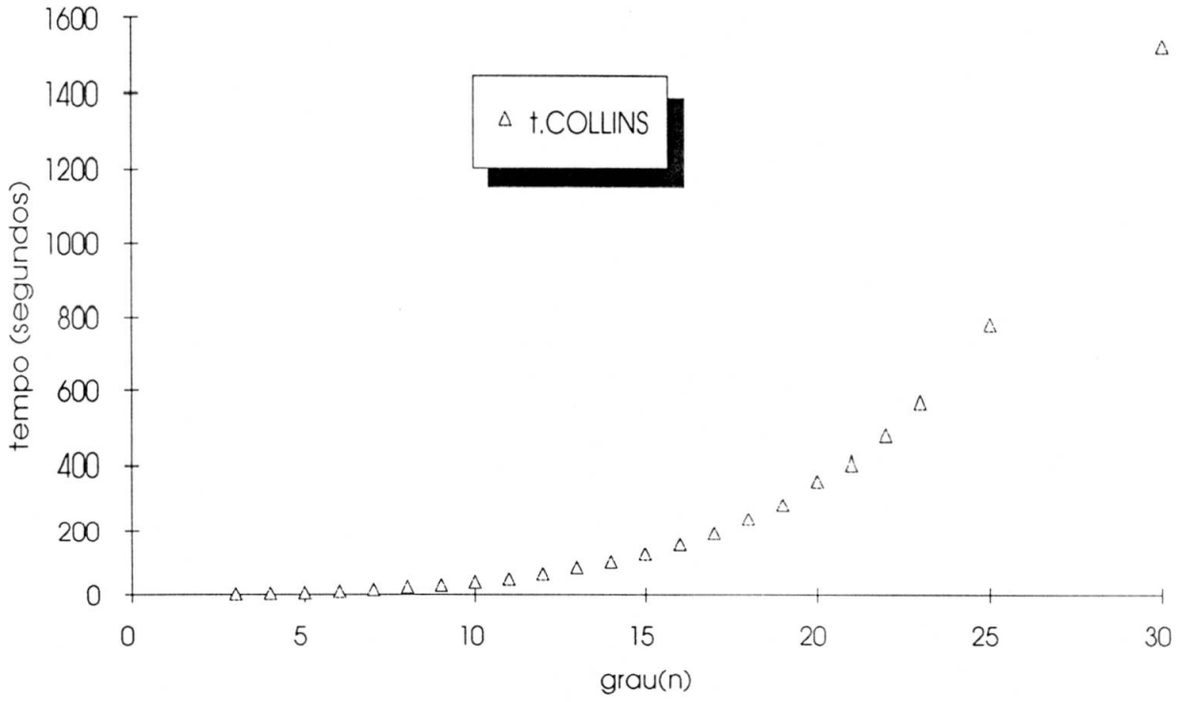

Figura 5.4: Tempos de execução para enumerar zeros num retângulo usando algoritmo (Collins-Krandick)

damente). Ela pode ser representada pela função $n^4.log(4n)/1000$. O gráfico da figura 5.5 mostra este comportamento.

Uma comparação entre os dois algoritmos, apresentando a diferença relativa de tempos observados para os mesmos é representada pela figura 5.6.

Pelo gráfico é possível observar-se que a diferença relativa dos tempos de execução do método de Collins-Krandick em relação ao método proposto (usando Sequências de Sturm) vai diminuindo de acordo com o aumento do grau do polinômio. Para graus mais elevados (25 em diante), o algoritmo de Collins-Krandick mostrou-se mais eficiente. Um comportamento similar é observado para polinômios de coeficientes complexos, porém para polinômios de grau 20 em diante. (veja fig. 5.7). Uma justificativa para tal fato ainda não foi encontrada e pode ser alvo de estudos futuros.

A variação de tempo em função do grau do algoritmo proposto para polinômios com coeficientes complexos pode ser visto na figura 5.8.

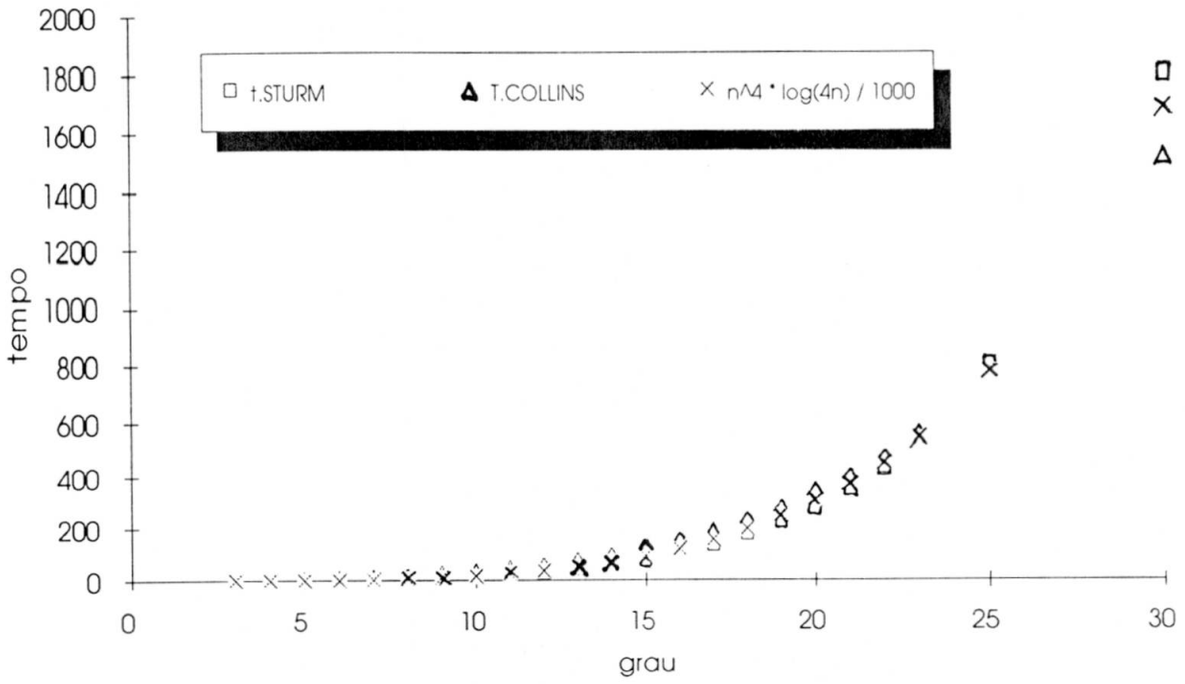

Figura 5.5: Função que expressa a dependência do tempo em relação ao grau do polinômio real para o algoritmo de Collins e o algoritmo proposto

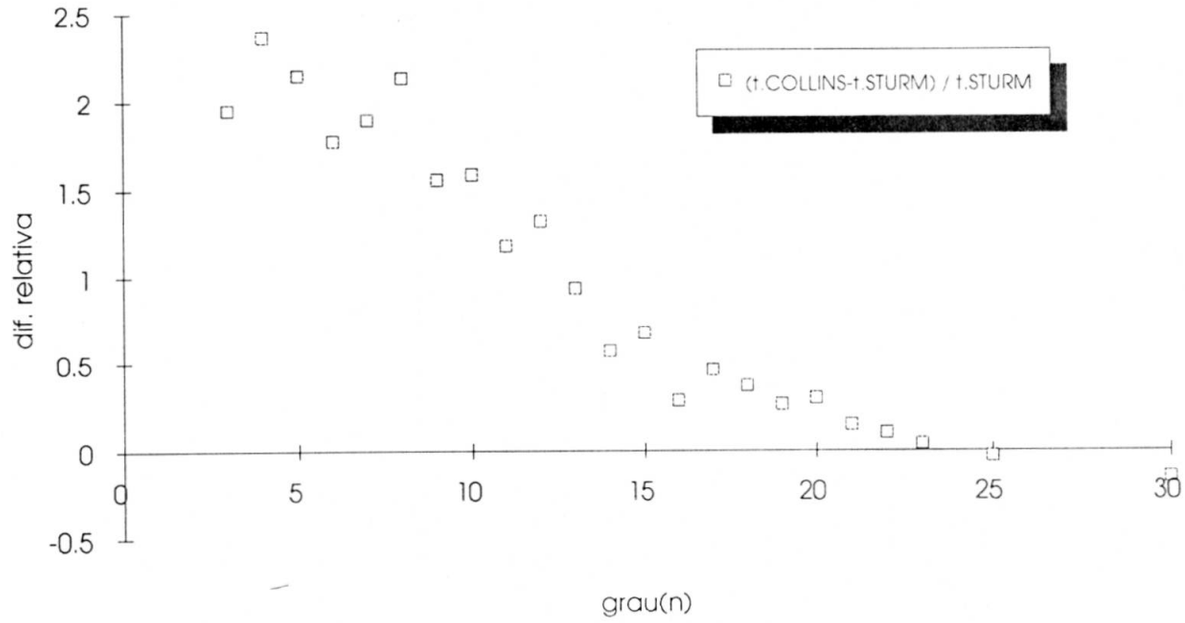

Figura 5.6: Diferença relativa de tempo do algoritmo de Collins em relação ao algoritmo proposto para polinômios reais

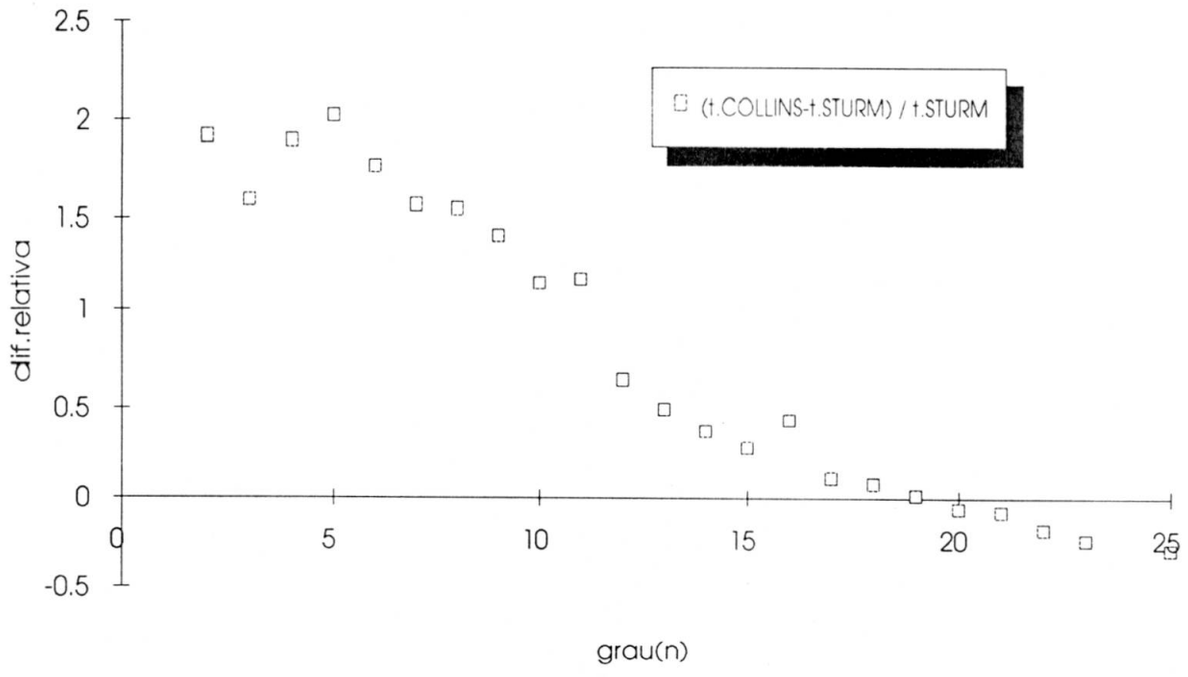

Figura 5.7: Diferença relativa de tempo do algoritmo de Collins em relação ao algoritmo proposto para polinômios complexos

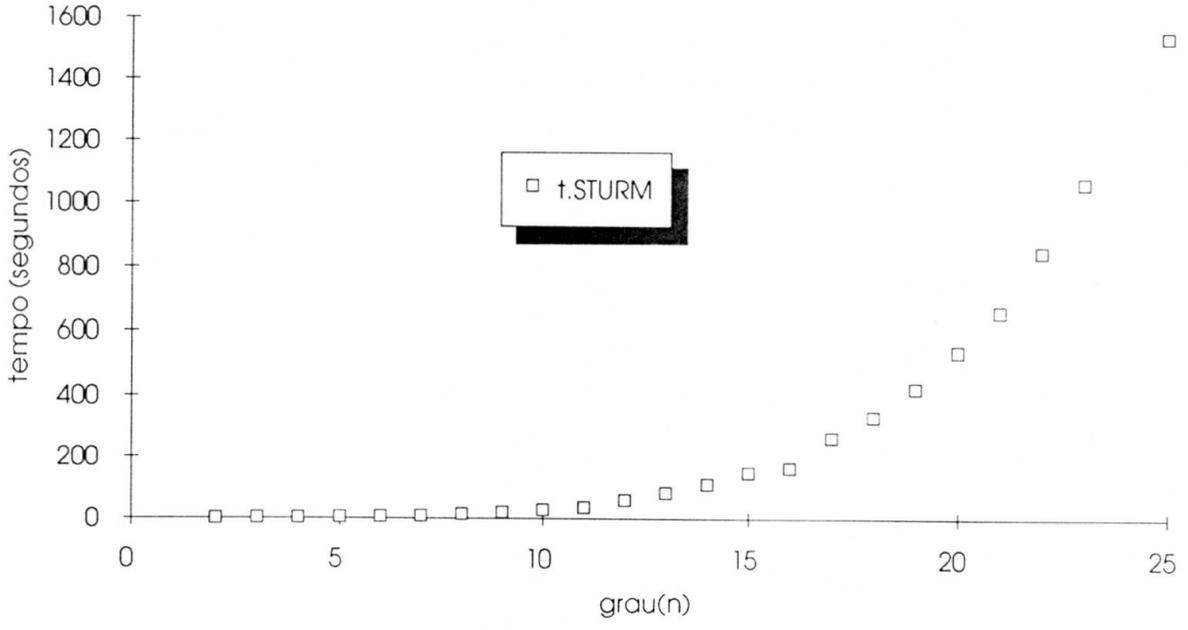

Figura 5.8: Tempos de execução para enumerar zeros num retângulo do algoritmo proposto para polinômios complexos

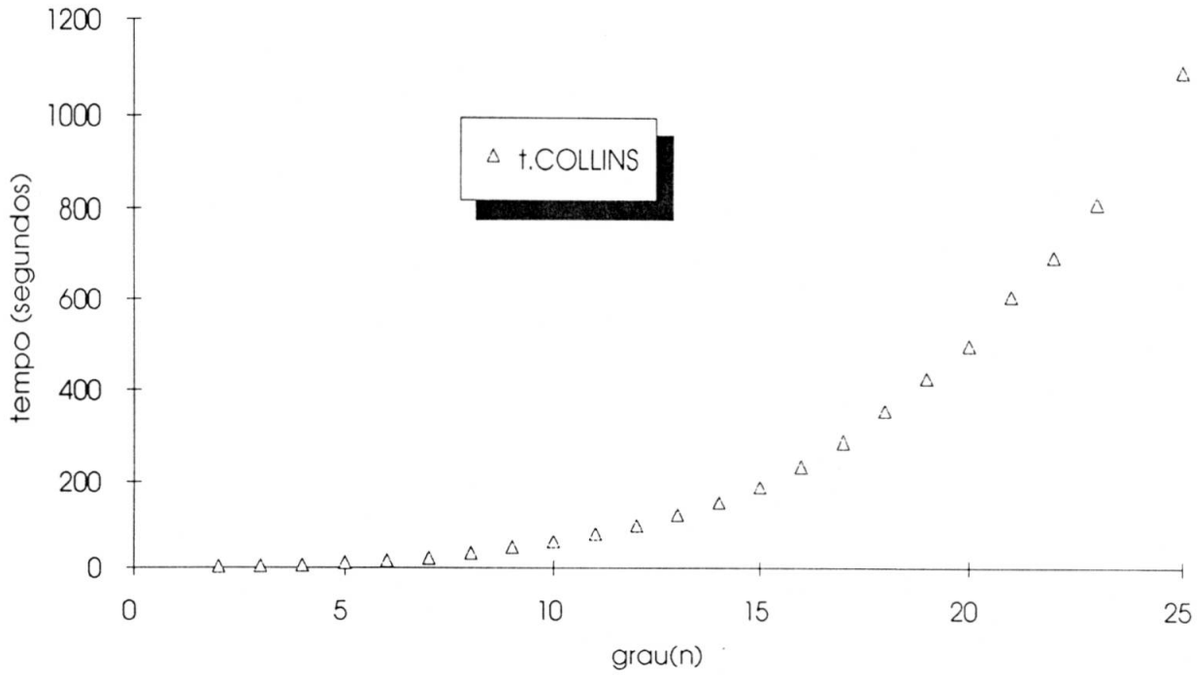

Figura 5.9: Tempos de execução para enumerar zeros num retângulo do algoritmo de Collins-Krandick para polinômios complexos

A variação de tempo em função do grau do algoritmo de Collins-Krandick para polinômios com coeficientes complexos pode ser visto na figura 5.9.

A função que expressa a dependência do tempo em relação grau do polinômio difere um pouco para os dois algoritmos, quando se trata de polinômios com coeficientes complexos. A figura 5.10 mostra a função que expressa esta dependência para o caso do algoritmo proposto, e a figura 5.11 mostra a função que expressa a dependência para o caso do algoritmo de Collins-Krandick.

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 (bissecionar) os intervalos isolados de polinômios distintos que não são disjuntos. Tendo em vista que não se tem informação sobre 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 necessário de acordo com o grau destes.

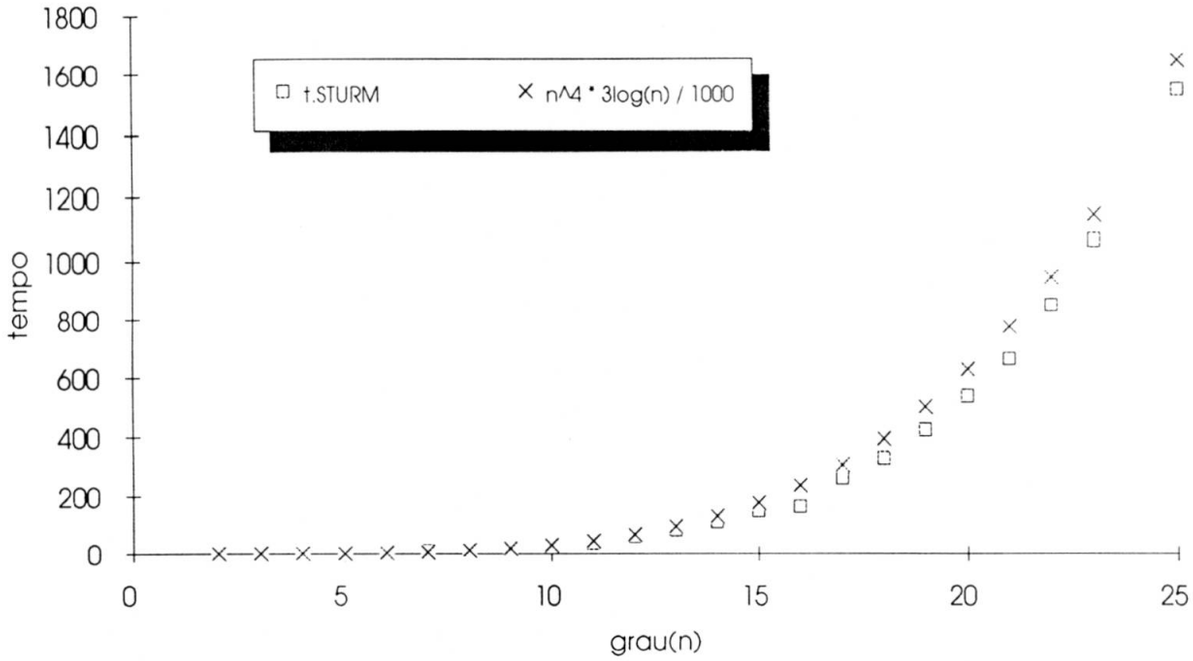

Figura 5.10: Função que expressa a dependência do tempo em relação ao grau do polinômio para o algoritmo proposto (caso complexo)

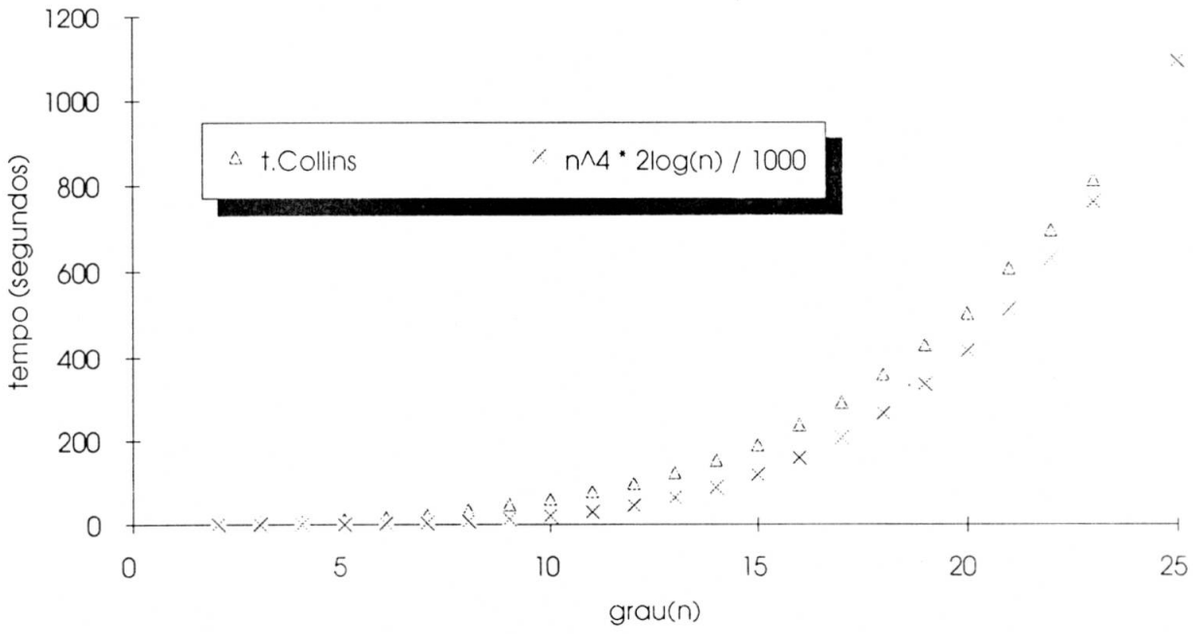

Figura 5.11: Função que expressa a dependência do tempo em relação ao grau do polinômio para o algoritmo de Collins-Krandick (caso complexo)

Tabela 5.3: Número de bissecções necessárias no algoritmo de Collins-Krandick

grau	caso real	caso complexo
3	4	8
4	16	16
5	20	24
6	22	32
7	32	32
8	32	30
9	36	36
10	40	42
11	36	56
12	42	56
13	64	54
14	54	64
15	68	66
16	68	70
17	70	74
18	84	76
[illegible]	[illegible]	[illegible]
20	80	90
21	72	94
22	80	102
23	100	96
25	100	116
30	118	-

A tabela 5.3 mostra o número de bissecções necessárias após isolar as raízes reais dos polinômios especializados no algoritmo de Collins-Krandick.

A figura 5.12 mostra que a variação do número de bissecções em relação ao grau de um polinômio real tende a ser linear.

O mesmo comportamento é observado para o caso de polinômios com coeficientes complexos, conforme mostra a figura 5.13.

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:

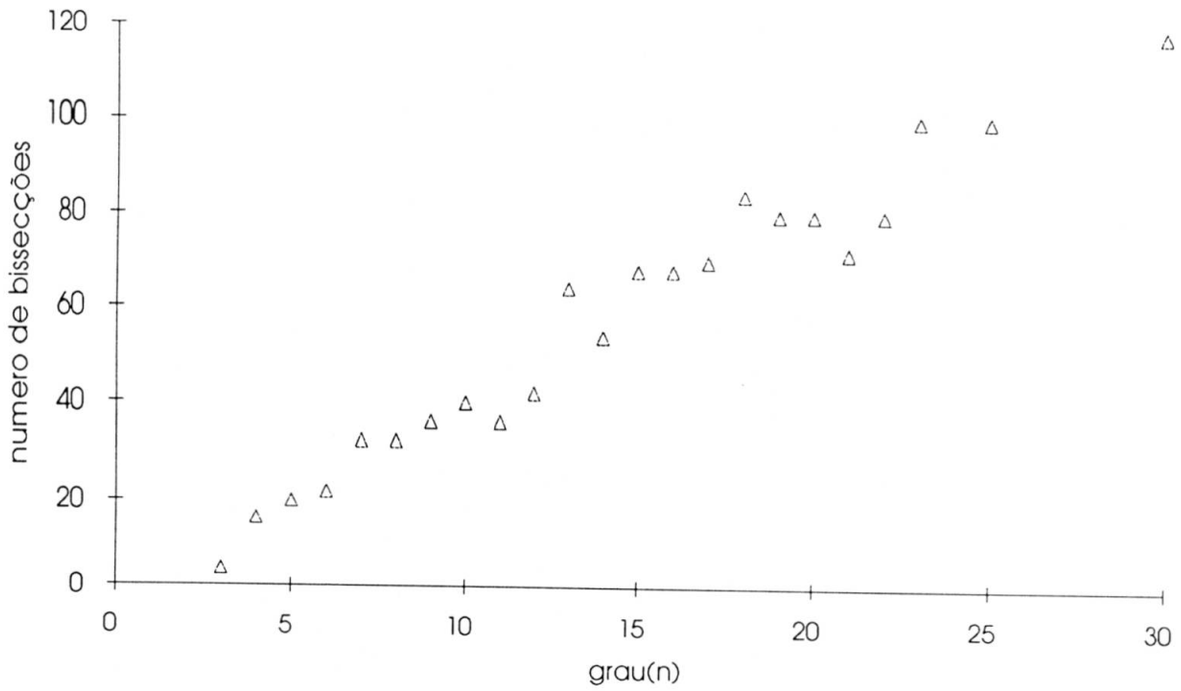

Figura 5.12: Variação do número de bissecções conforme grau do polinómio real

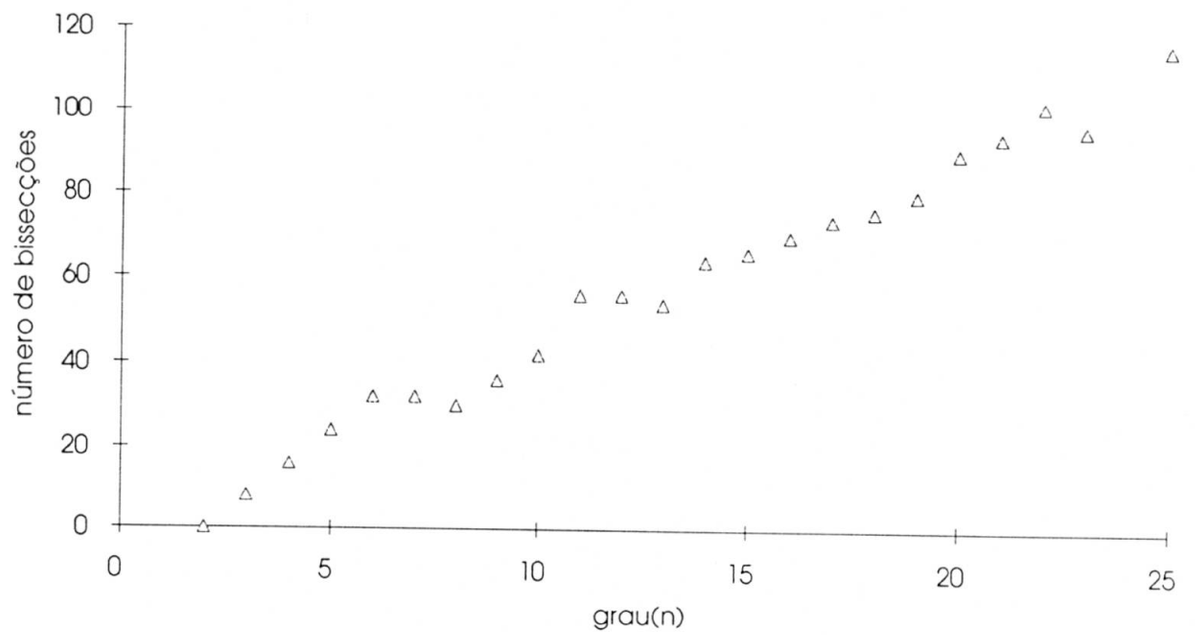

Figura 5.13: Variação do número de bissecções conforme grau do polinómio (caso complexo

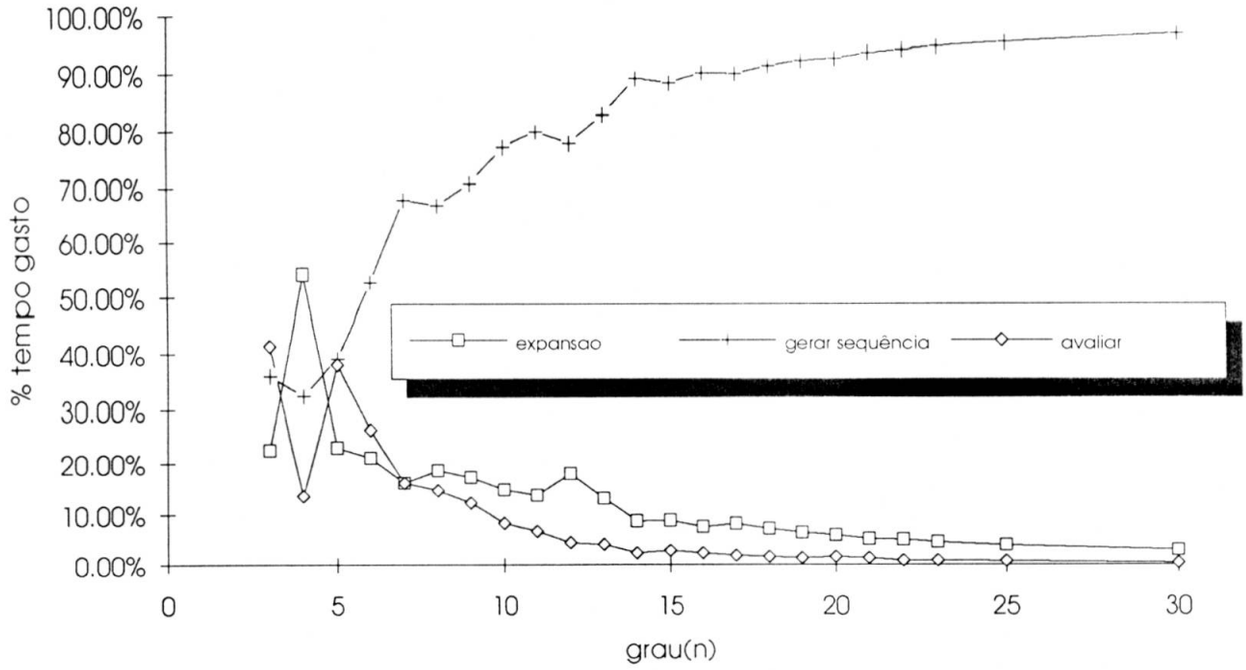

Figura 5.14: Tempo percentual gasto para cada uma das tarefas necessárias para um nível de busca no algoritmo proposto

. Para 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 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.

Os tempos necessários para cada uma destas operações foram observados para um nível de busca. A figura 5.14 mostra os percentuais de tempo gastos com cada uma das operações necessárias para se pesquisar os 4 subretângulos resultantes do particionamento de um retângulo no algoritmo proposto para polinômios reais.

A figura 5.15 mostra os percentuais de tempo gastos com cada uma das operações necessárias para se pesquisar os 2 subretângulos resultantes do particio-

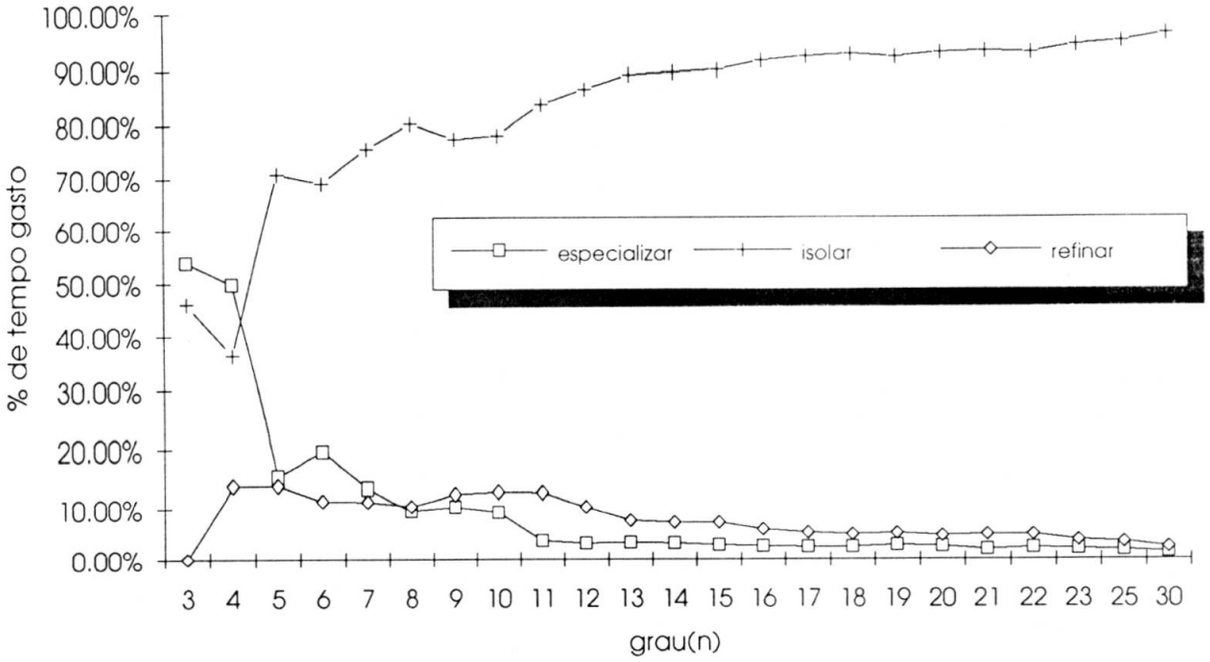

Figura 5.15: Tempo percentual gasto para cada uma das tarefas necessárias para um nível de busca no algoritmo de Collins-Krandick

namento de um retângulo usando o algoritmo de Collins-Krandick para polinômios reais

A figura 5.16 mostra a variação do tempo total necessário para se realizar o particionamento de um retângulo para os dois algoritmos.

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

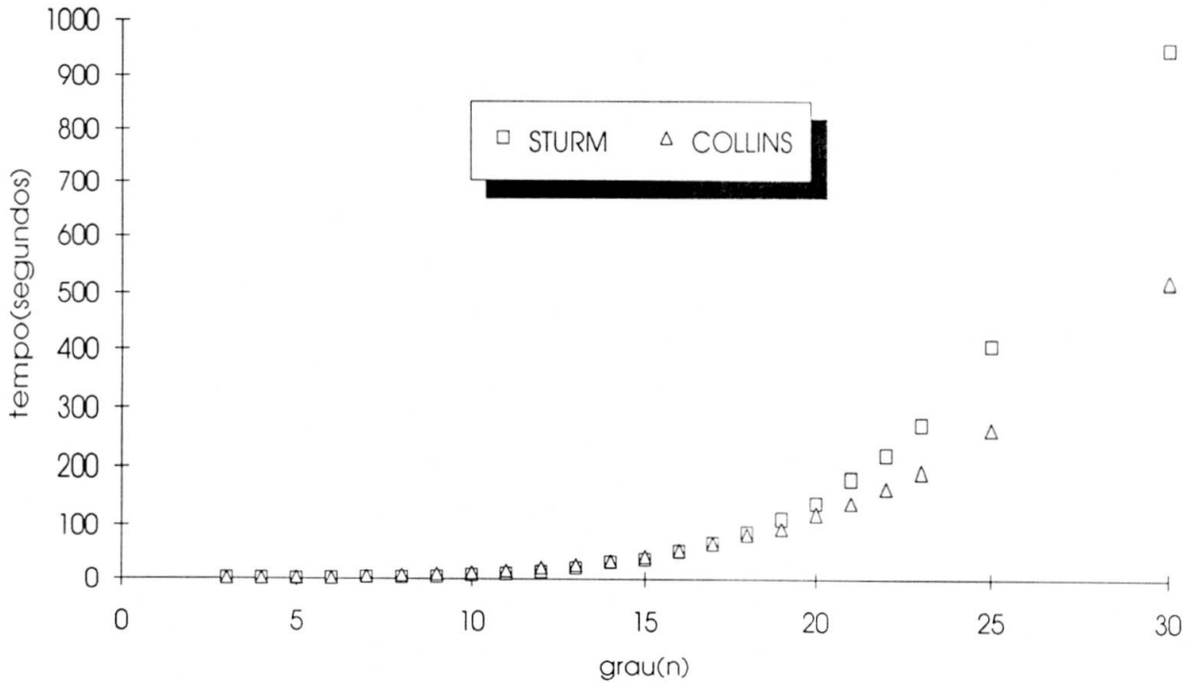

Figura 5.16: Variação do tempo para se fazer a pesquisa em um nível de busca para o algoritmo proposto e o algoritmo de Collins-Krandick

testes com polinômios de grau fixo (10), variando-se o tamanho dos coeficientes gerados aleatoriamente (3, 6, 9, 12 e 15 dígitos decimais). A figura 5.17 mostra os tempos observados para os 2 algoritmos.

A variação do tempo para isolar zeros de polinômios com coeficientes reais usando o algoritmo proposto pode ser vista na figura 5.18, juntamente com a função que expressa tal comportamento.

Outros polinômios foram testados, conforme descrito anteriormente. Os polinômios abaixo têm seus resultados na tabela 5.4.

$p1 : 1000z^{10} - 2500z^9 - 460800z^8 - 9133400z^7 - 50761800z^6 - 88653100z^5 - 53510400z^4 - 37313000z^3 - 197170000z^2 - 364800000z - 198000000;$

$p2 : (z - (1 + \frac{9}{10}i)) * (z - (1 + \frac{99}{100}i)) * (z - (1 + \frac{999}{1000}i)) * (z - (1 + \frac{9999}{10000}i))$

$p3 : z^4 - (5 + 3i)z^3 + (6 + 7i)z^2 - (54 - 22i)z + (120 + 90i)$

$p4 : 470832z^3 - 665857(i + 1)z^2 - 1883328iz - 2663128(1 - i)$

$p5 : z^6 - 1,2z^5 + 0,04z^4 - 0,132z^3 - 0,7955z^2 - 1,13576z + 0,89496$

$p6 : z^9 - 1,1z^8 + 1,12z^7 - 1,232z^6 + 2,084z^5 - 2,2924z^4 + 1,075648z^3 - 1,1832128z^2 +$

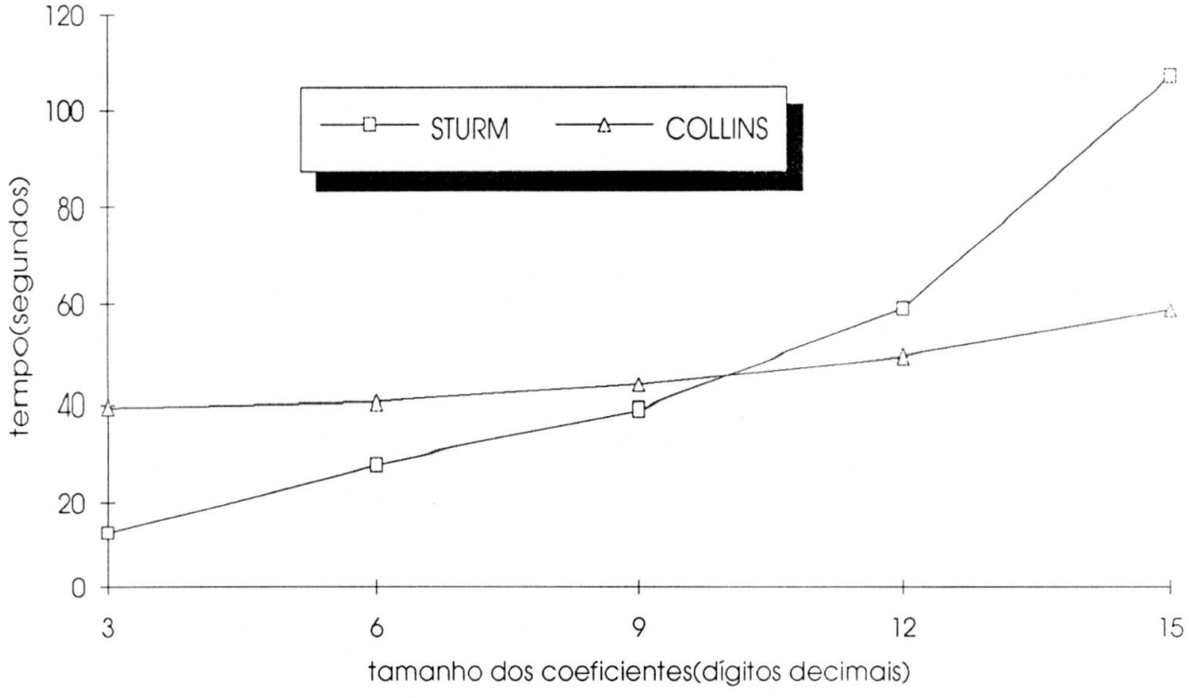

Figura 5.17: Variação do tempo de execução na enumeração de polinômios reais de acordo com o tamanho dos coeficientes

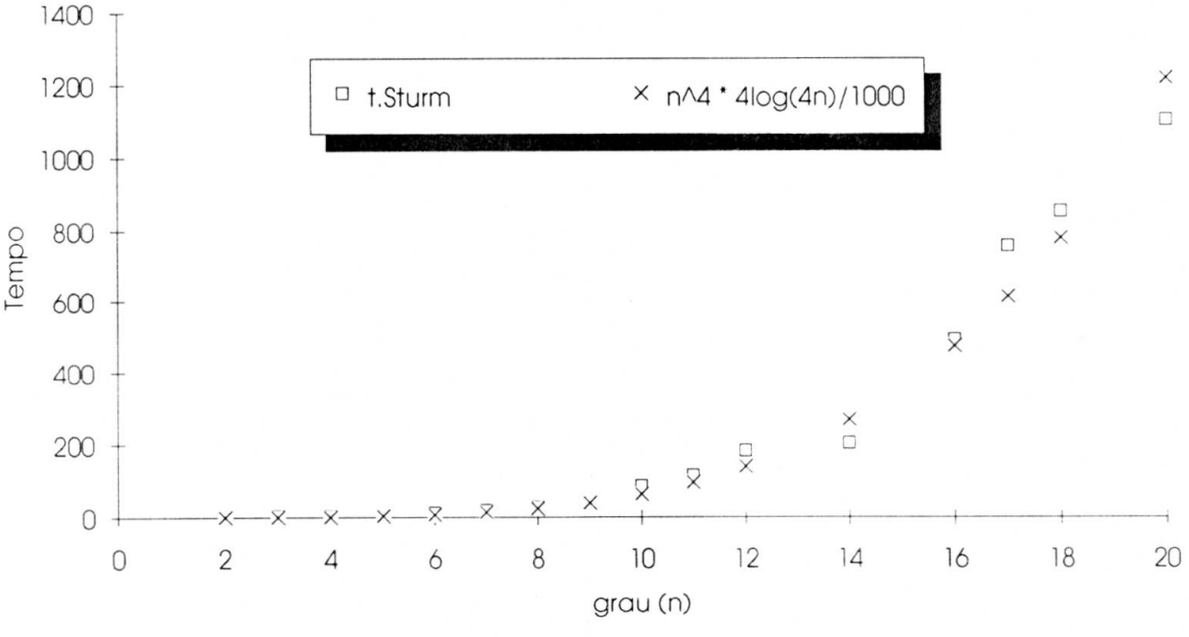

Figura 5.18: Variação do tempo de execução para isolar zeros do algorito proposto de acordo com o grau do polinômio real

Tabela 5.4: 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

$1.07910544z - 1.187015984$

$p7 : 13652098z^4 - (38613965 + 38613965i)z^3 + (2i)z^2 + (-154455860 + 154455860i)z + 218433576$

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

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

cujas raízes são: $\alpha_k = e^{\frac{2\pi k_1}{n}}$.

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.

6 CONCLUSÕES E SUGESTÕES DE TRABALHO FUTURO

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 desenvolvimento e implementação de algoritmos exatos para enumerar zeros de polinômios complexos em diferentes regiões do plano (círculos e retângulos) proporcionou alternativas seguras para se identificar a presença ou não de raízes em qualquer parte do plano complexo. Além disto, numa situação onde se necessita apenas saber quantos zeros existem numa determinada região, os testes realizados sugerem que para polinômios de grau abaixo de 25, os círculos necessitam menos tempo de processamento do que os retângulos. Entretanto, verificou-se que as regiões circulares não são as mais adequadas para a tarefa de isolar as raízes de um polinômio.

O algoritmo para isolar raízes desenvolvido neste trabalho permitiu resolver 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 subalgoritmos mais adequados 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.

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 operações necessárias, observou-se que até grau 20, o aumento de tempo em relação ao grau é similar para ambos os algoritmos. A partir daí, o aumento de tempo ocorre bem mais rápido para o algoritmo proposto, daí o seu incoveniente 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 aumento do tamanho dos coeficientes cresce bem mais rápido).

Problemas pendentes

O algoritmo desenvolvido neste trabalho , apesar de produzir resultados exatos, ainda não tem um desempenho satisfatório para graus mais elevados (acima de 20). Um dos aspectos observados é que muitas vezes a quota superior para o módulo das raízes é muito grande. A utilização de um método para produzir quotas superiores menores, (como por exemplo [ZEH 91] não resolve a situação. Embora as quotas superiores possam ser menores, ocorre com frequência números racionais onde numerador e denominador têm tamanho de 4 a 5 dígitos decimais.

Isto já é suficiente para degradar o tempo necessário para se enumerar os zeros dentro de um retângulo cujos vértices são racionais de alta magnitude. Ao se tomar o menor número inteiro maior que a quota superior calculada, observa-se que a diferença de tempo é desprezível, se comparada com a utilização da quota superior 4.1. Algumas hipóteses foram levantadas a respeito de minimizar a região inicial de pesquisa, porém ainda precisam ser comprovadas se realmente reduzem o trabalho computacional ou não. A primeira delas diz respeito a utilização de testes para eliminar regiões livres de zeros, sem a aplicação de sequências de Sturm. Como as regiões livres de zero ocorrem com bastante frequência, parece razoável que se utilize algum procedimento para detectá-las, reduzindo assim a área que deve ser pesquisada. Por exemplo, se o polinômio tem coeficientes complexos, não tem raízes nos eixos, pode-se então aplicar um teste de exclusão para cada um dos subretângulos componentes do retângulo inicial.

Outro aspecto que pode ser verificado é a sequência de subretângulos que é pesquisada. Na forma como está, o algoritmo segue sempre a mesma ordem dos subretângulos pesquisados (I, II, III, IV), conforme figura 4.1. Como não se conhece a priori a distribuição das raízes dentro do retângulo inicial, poder-se-ia alternar esta ordem, estabelecendo uma sequência alternada (ora percorrer no sentido horário, ora no sentido antihorário), ou através de uma determinação aleatória do subretângulo que vai ser pesquisado.

6.1 Oportunidades de Paralelismo

Tanto o algoritmo de Collins, como o algoritmo aqui proposto para isolar raízes polinomiais, favorecem, pelas suas características, a utilização de paralelismo.'

Neste trabalho, apenas foi possível identificar as fases que se mostram adequadas para paralelização. Certamente, estudos futuros serão necessários para comprovar efetivamente se haverá benefícios ou não na utilização deste recurso.

Tendo em vista que o polinômio pode ter raízes múltiplas, neste caso o resultado da decomposição livre de quadrados fornece fatores que podem ser tratados de forma independente (em paralelo) para determinação de suas raízes.

No processo de isolamento, a cada particionamento, gera-se subregiões que podem ser pesquisadas simultaneamente na fase de enumeração .

No algoritmo de Collins, sempre se precisa isolar raízes reais de 2 polinômios, que são independentes, podendo esta tarefa ser realizada em paralelo. No algoritmo proposto, sempre se trabalha com 2 sequências de Sturm, que também são independentes podendo ser tratadas simultaneamente.

Tendo-se as raízes isoladas, o cálculo posterior destas pode também ser realizado em paralelo.

6.2 Sugestões de trabalhos futuros

O trabalho aqui desenvolvido na verdade é um subconjunto de um projeto maior que deve proporcionar a utilização de diversos módulos que tratam de problemas polinomiais, tanto de forma individual como global, integrando-se os diversos módulos. O tratamento global deve resolver o problema desde a etapa inicial: Inicialmente, identifica-se a existência de raízes múltiplas através da decomposição "square-free". Para cada fator então, estabelece-se a região inicial de pesquisa. Na sequência, identifica-se as características das raízes e do polinômio para adequação de um algoritmo para isolar as raízes. Tendo as regiões isoladas, aplicar um algoritmo numérico para aproximar as raízes com o menor erro possível.

Tendo resolvido as etapas iniciais de enumeração e isolamento (que são as mais problemáticas), pretende-se realizar o processo de aproximação das raízes usando um método numérico iterativo para determinação simultânea de raízes, cujos valores iniciais seriam o centro das regiões isoladas. Não se pode negar que os cálculos

realizados em ponto flutuante, apesar de estarem sujeitos a erros de arredondamento, são bem mais rápidos do que os cálculos realizados de forma algébrica (aritmética exata). Além disto, na fase de aproximação não se necessita do rigor da exatidão que havia nas fases de enumeração e isolamento. Para garantir se os resultados estão suficientemente próximos dos zeros procurados, uma verificação dos resultados pode ser feita através da utilização de uma última iteração intervalar, possibilitando obter limites de erro para as aproximações calculadas.

Desta forma, o resultado seria um tratamento do problema de zeros polinomiais com abordagem híbrida, usando computação simbólica nas fases iniciais (que necessariamente precisam de valores exatos), e uma abordagem numérica na etapa de cálculo (aproximação dos zeros). A introdução de paralelismo nas diversas fases favoráveis para tal deve então ser estudada detalhadamente para verificar sua viabilidade e os prováveis benefícios que esta abordagem pode trazer.

ANEXO A-1 EXEMPLOS DE POLINÔMIOS DE COEFICIENTES COMPLEXOS PARA TESTES

1. $p(Z) = (-757\,i - 559)\,Z^2 + (-688\,i - 695)\,Z - 724\,i - 580$

2. $p(Z) = (804 - 567\,i)\,Z^3 + (954\,i - 699)\,Z^2 + (-718\,i - 545)\,Z + 926\,i - 758$

3. $p(Z) = (838\,i + 835)\,Z^4 + (-669\,i - 725)\,Z^3 + (-647\,i - 586)\,Z^2 + (851 - 633\,i)\,Z - 650\,i + 921$

4. $p(Z) = (-632\,i - 591)\,Z^5 + (-597\,i - 584)\,Z^4 + (-761\,i - 671)\,Z^3 + (934\,i - 527)\,Z^2 + (-756\,i - 745)\,Z + 840\,i + 814$

5. $p(Z) = (836\,i + 780)\,Z^6 + (996\,i - 745)\,Z^5 + (902\,i - 617)\,Z^4 + (846\,i - 652)\,Z^3 + (921 - 703\,i)\,Z^2 + (818\,i + 935)\,Z + 1014\,i + 852$

6. $p(Z) = (938\,i - 610)\,Z^7 + (780\,i + 942)\,Z^6 + (962 - 591\,i)\,Z^5 + (824\,i - 537)\,Z^4 + (922 - 700\,i)\,Z^3 + (1001 - 545\,i)\,Z^2 + (879\,i - 523)\,Z - 738\,i - 526$

7. $p(Z) = (798\,i - 538)\,Z^8 + (-715\,i - 656)\,Z^7 + (824 - 590\,i)\,Z^6 + (937 - 547\,i)\,Z^5 + (824 - 664\,i)\,Z^4 + (1008\,i - 575)\,Z^3 + (937 - 558\,i)\,Z^2 + (1008\,i + 939)\,Z + 908\,i + 776$

8. $p(Z) = (832\,i + 887)\,Z^9 + (1023\,i + 887)\,Z^8 + (810\,i - 646)\,Z^7 + (775\,i - 704)\,Z^6 + (-767\,i - 587)\,Z^5 + (-549\,i - 535)\,Z^4 + (905 - 555\,i)\,Z^3 + (-544\,i -$

$591)\,Z^2+(930-747\,i)\,Z-746\,i-679$

9. $p(Z)=(776-608\,i)\,Z^{10}+(963\,i-725)\,Z^9+(824-598\,i)\,Z^8+(797\,i-639)\,Z^7+(997\,i-555)\,Z^6+(800-738\,i)\,Z^5+(-732\,i-648)\,Z^4+(-665\,i-735)\,Z^3+(847\,i+820)\,Z^2+(1018\,i-522)\,Z-728\,i+925$

10. $p(Z)=(988\,i+860)\,Z^{11}+(-532\,i-680)\,Z^{10}+(796-517\,i)\,Z^9+(-593\,i-689)\,Z^8+(897\,i-729)\,Z^7+(876\,i+796)\,Z^6+(-764\,i-590)\,Z^5+(1009-720\,i)\,Z^4+(-547\,i-588)\,Z^3+(1006-635\,i)\,Z^2+(950-546\,i)\,Z+815\,i-609$

11. $p(Z)=(952\,i-603)\,Z^{12}+(-571\,i-598)\,Z^{11}+(874\,i-514)\,Z10+(800\,i-539)\,Z^9+(852\,i-658)\,Z^8+(802\,i+991)\,Z^7+(-732\,i-525)\,Z^6+(979\,i+800)\,Z^5+(853\,i-640)\,Z^4+(982\,i-712)\,Z^3+(829\,i-530)\,Z^2+(951\,i-648)\,Z+896\,i-716$

12. $p(Z)=(851-531\,i)\,Z^{13}+(906\,i-525)\,Z^{12}+(998\,i+908)\,Z^{11}+(-641\,i-701)\,Z^{10}+(-720\,i-577)\,Z^9+(820-728\,i)\,Z^8+(-708\,i-738)\,Z^7+(-605\,i-654)\,Z^6+(830\,i+776)\,Z^5+(1006\,i+780)\,Z^4+(948-569\,i)\,Z^3+(916\,i-526)\,Z^2+(922-512\,i)\,Z-553\,i-593$

13. $p(Z)=(840-610\,i)\,Z^{14}+(798\,i-664)\,Z^{13}+(967-745\,i)\,Z^{12}+(923\,i+817)\,Z^{11}+(970-681\,i)\,Z^{10}+(831\,i+1015)\,Z^9+(772\,i-674)\,Z^8+(878\,i-521)\,Z^7+(1022-598\,i)\,Z^6+(-544\,i-703)\,Z5+(1017\,i-643)\,Z^4+(1021\,i+809)\,Z^3+(809-569\,i)\,Z^2+(-751\,i-568)\,Z+976\,i+966$

14. $p(Z)=(810-594\,i)\,Z^{15}+(-586\,i-565)\,Z^{14}+(797\,i-512)\,Z^{13}+(775\,i+862)\,Z^{12}+(844\,i+820)\,Z^{11}+(931-568\,i)\,Z^{10}+(-667\,i-626)\,Z^9+(-715\,i-547)\,Z^8+(791-600\,i)\,Z^7+(866\,i-519)\,Z^6+(-705\,i-691)\,Z^5+(844\,i+$

$941)\,Z^4+(864\,i-676)\,Z^3+(-739\,i-685)\,Z^2+(922-531\,i)\,Z-634\,i-667$

15. $p(Z)=(996\,i+1003)\,Z^{16}+(-683\,i-764)\,Z^{15}+(967-548\,i)\,Z^{14}+(-692\,i-525)\,Z^{13}+(-755\,i-554)\,Z^{12}+(788-566\,i)\,Z^{11}+(857-515\,i)\,Z^{10}+(895\,i-552)\,Z^9+(829\,i+1002)\,Z^8+(-562\,i-676)\,Z^7+(-616\,i-733)\,Z^6+(-760\,i-513)\,Z^5+(972\,i+1022)\,Z^4+(1006-601\,i)\,Z^3+(770\,i-748)\,Z^2+(913\,i-601)\,Z+962\,i+793$

16. $p(Z)=(979\,i-627)\,Z^{17}+(918-764\,i)\,Z^{16}+(-642\,i-670)\,Z^{15}+(885-570\,i)\,Z^{14}+(-684\,i-664)\,Z^{13}+(852\,i-722)\,Z^{12}+(815\,i+952)\,Z^{11}+(-740\,i-693)\,Z^{10}+(-550\,i-759)\,Z^9+(-704\,i-721)\,Z^8+(845\,i+1000)\,Z^7+(-721\,i-739)\,Z^6+(-571\,i-512)\,Z^5+(-648\,i-706)\,Z^4+(772\,i-549)\,Z^3+(807-639\,i)\,Z^2+(-668\,i-758)\,Z+776\,i-585$

17. $p(Z)=(-628\,i-647)\,Z^{18}+(850\,i-618)\,Z^{17}+(-687\,i-514)\,Z^{16}+(975-647\,i)\,Z^{15}+(794\,i+895)\,Z^{14}+(786\,i+959)\,Z^{13}+(995\,i-613)\,Z^{12}+(977\,i-611)\,Z^{11}+(-682\,i-740)\,Z^{10}+(896-532\,i)\,Z^9+(791\,i+977)\,Z^8+(818-710\,i)\,Z^7+(831\,i+914)\,Z^6+(891\,i+873)\,Z^5+(-745\,i-555)\,Z^4+(814\,i-527)\,Z^3+(1007-585\,i)\,Z^2+(1023-618\,i)\,Z+768\,i-619$

18. $p(Z)=(770\,i+768)\,Z^{19}+(833\,i+958)\,Z^{18}+(-762\,i-719)\,Z^{17}+(847\,i-719)\,Z^{16}+(-581\,i-613)\,Z^{15}+(-559\,i-686)\,Z^{14}+(916\,i+830)\,Z^{13}+(823\,i-643)\,Z^{12}+(1021\,i-546)\,Z^{11}+(947\,i-655)\,Z^{10}+(906\,i-633)\,Z^9+(975\,i+877)\,Z^8+(-743\,i-530)\,Z^7+(-674\,i-662)\,Z^6+(1015-764\,i)\,Z^5+(797\,i-521)\,Z^4+(-603\,i-589)\,Z^3+(989\,i+913)\,Z^2+(923\,i+824)\,Z+1021\,i+936$

19. $p(Z)=(901\,i+850)\,Z^{20}+(964\,i-757)\,Z^{19}+(-661\,i-725)\,Z^{18}+(-732\,i-698)\,Z^{17}+(996\,i-625)\,Z^{16}+(928\,i-558)\,Z^{15}+(887\,i-665)\,Z^{14}+(-681\,i-$

$532)\,Z^{13}+(893-634\,i)\,Z^{12}+(-639\,i-721)\,Z^{11}+(930\,i-516)\,Z^{10}+(-690\,i-639)\,Z^9+(977\,i+881)\,Z^8+(945-572\,i)\,Z^7+(879-653\,i)\,Z^6+(883-512\,i)\,Z^5+(974\,i+956)\,Z^4+(870\,i-581)\,Z^3+(807\,i+933)\,Z^2+(1003\,i+957)\,Z-740\,i+771$

20. $p(Z)=(928-580\,i)\,Z^{21}+(866\,i+861)\,Z^{20}+(917-537\,i)\,Z^{19}+(-765\,i-587)\,Z^{18}+(808\,i+931)\,Z^{17}+(781\,i-513)\,Z^{16}+(-695\,i-682)\,Z^{15}+(990\,i+983)\,Z^{14}+(795-626\,i)\,Z^{13}+(-641\,i-530)\,Z^{12}+(1006\,i-636)\,Z^{11}+(787\,i+1010)\,Z^{10}+(918\,i-757)\,Z^9+(874\,i-647)\,Z^8+(-551\,i-686)\,Z^7+(907\,i+909)\,Z^6+(1020\,i+855)\,Z^5+(-684\,i-542)\,Z^4+(797\,i+943)\,Z^3+(-630\,i-759)\,Z^2+(987\,i+856)\,Z-712\,i-578$

21. $p(Z)=(791\,i+944)\,Z^{22}+(853\,i-734)\,Z^{21}+(830\,i+874)\,Z^{20}+(910\,i-553)\,Z^{19}+(-580\,i-699)\,Z^{18}+(793\,i+796)\,Z^{17}+(901\,i+779)\,Z^{16}+(-521\,i-546)\,Z^{15}+(811\,i-621)\,Z^{14}+(1006-522\,i)\,Z^{13}+(889\,i+848)\,Z^{12}+(939\,i-668)\,Z^{11}+(961\,i+931)\,Z^{10}+(-579\,i-608)\,Z^9+(1013\,i+913)\,Z^8+(966\,i-552)\,Z^7+(936-571\,i)\,Z^6+(782\,i-521)\,Z^5+(876\,i-703)\,Z^4+(778\,i-570)\,Z^3+(827-692\,i)\,Z^2+(992\,i-762)\,Z+936\,i+969$

22. $p(Z)=(898\,i+944)\,Z^{23}+(-728\,i-609)\,Z^{22}+(991-756\,i)\,Z^{21}+(939\,i+909)\,Z^{20}+(974-626\,i)\,Z^{19}+(827-733\,i)\,Z^{18}+(-588\,i-606)\,Z^{17}+(1019\,i+833)\,Z^{16}+(867\,i-741)\,Z^{15}+(1005\,i-644)\,Z^{14}+(946\,i+876)\,Z^{13}+(948-646\,i)\,Z^{12}+(1008\,i+857)\,Z^{11}+(834\,i-754)\,Z^{10}+(-544\,i-643)\,Z^9+(788\,i-576)\,Z^8+(-608\,i-752)\,Z^7+(768\,i+1022)\,Z^6+(1020\,i-613)\,Z5+(969-584\,i)\,Z^4+(1019\,i-737)\,Z^3+(-663\,i-522)\,Z^2+(-633\,i-706)\,Z+1004\,i-578$

23. $p(Z)=(950\,i+849)\,Z^{25}+(834\,i-627)\,Z^{24}+(781\,i+907)\,Z^{23}+(885\,i-757)\,Z^{22}+(976-531\,i)\,Z^{21}+(817\,i+853)\,Z^{20}+(800\,i-557)\,Z^{19}+(907-$

$573\,i)\,Z^{18}+(-712\,i-560)\,Z^{17}+(892\,i+769)\,Z^{16}+(-536\,i-735)\,Z^{15}+(1016\,i+934)\,Z^{14}+(-537\,i-591)\,Z^{13}+(957\,i-564)\,Z^{12}+(912-597\,i)\,Z^{11}+(832\,i+826)\,Z^{10}+(-766\,i-731)\,Z^{9}+(871\,i+913)\,Z^{8}+(-699\,i-626)\,Z^{7}+(-684\,i-547)\,Z^{6}+(851\,i-614)\,Z^{5}+(-551\,i-736)\,Z^{4}+(879-559\,i)\,Z^{3}+(878\,i+911)\,Z^{2}+(1020\,i+819)\,Z-722\,i+819$

24. $p(Z)=(-634\,i-705)\,Z^{30}+(812\,i+1016)\,Z^{29}+(947-635\,i)\,Z^{28}+(-577\,i-643)\,Z^{27}+(930\,i-564)\,Z^{26}+(913\,i+834)\,Z^{25}+(-685\,i-765)\,Z^{24}+(-647\,i-760)\,Z^{23}+(785\,i+797)\,Z^{22}+(902\,i-611)\,Z^{21}+(-691\,i-712)\,Z^{20}+(936\,i+837)\,Z^{19}+(794\,i+847)\,Z^{18}+(-512\,i-744)\,Z^{17}+(-523\,i-547)\,Z^{16}+(-626\,i-607)\,Z^{15}+(851\,i+968)\,Z^{14}+(-537\,i-712)\,Z^{13}+(879\,i-528)\,Z^{12}+(-655\,i-694)\,Z^{11}+(791-659\,i)\,Z^{10}+(783\,i-634)\,Z^{9}+(950\,i-565)\,Z^{8}+(884\,i-658)\,Z^{7}+(883\,i+941)\,Z^{6}+(955\,i-565)\,Z^{5}+(998\,i+976)\,Z^{4}+(830-543\,i)\,Z^{3}+(891\,i-612)\,Z^{2}+(852-588\,i)\,Z+902\,i+900$

ANEXO A-2 EXEMPLOS DE POLINÔMIOS GERADOS ALEATORIAMENTE PARA TESTE

1. $p(Z) = -659\,Z^2 - 703\,Z + 914$

2. $p(Z) = -610\,Z^3 - 532\,z^2 + 879\,Z - 666$

3. $p(Z) = 780\,Z^4 + 1017\,Z^3 + 774\,Z^2 + 782\,Z + 989$

4. $p(Z) = 876\,Z^5 - 537\,Z^4 + 777\,Z^3 - 584\,Z^2 + 980\,Z - 614$

5. $p(Z) = 1019\,Z^6 - 663\,Z^5 + 1020\,Z^4 - 681\,Z^3 + 827\,Z^2 + 811\,Z + 913$

6. $p(Z) = 833\,Z^7 - 533\,Z^6 - 734\,Z^5 + 845\,Z^4 - 554\,Z^3 - 556\,Z^2 - 610\,Z + 927$

7. $p(Z) = -729\,Z^8 + 841\,Z^7 - 758\,Z^6 - 679\,Z^5 + 814\,Z^4 - 544\,Z^3 - 609\,Z^2 + 803\,Z + 829$

8. $p(Z) = -620\,Z^9 + 925\,Z^8 + 918\,Z^7 - 535\,Z^6 + 884\,Z^5 + 796\,Z^4 - 726\,Z^3 - 733\,Z^2 + 919\,Z - 584$

9. $p(Z) = -589\,Z^{10} - 567\,Z^9 + 932\,Z^8 - 732\,z^7 + 788\,Z^6 + 984\,Z^5 - 666\,Z^4 - 708\,Z^3 - 725\,Z^2 - 691\,Z - 753$

10. $p(Z) = 840\,Z^{11} - 758\,Z^{10} + 929\,Z^9 - 652\,Z^8 - 759\,Z^7 - 762\,Z^6 - 564\,Z^5 - 590\,Z^4 - 586\,Z^3 - 632\,Z^2 - 543\,Z + 912$

11. $p(Z) = 987\,Z^{12} - 725\,Z^{11} + 774\,Z^{10} - 572\,Z^9 + 962\,Z^8 + 923\,Z^7 - 694\,Z^6 + 773\,Z^5 + 770\,Z^4 + 950\,Z^3 + 800\,Z^2 + 916\,Z - 713$

12. $p(Z) = -556\,Z^{13} - 607\,Z^{12} + 899\,Z^{11} + 775\,Z^{10} - 643\,Z^9 - 738\,Z^8 - 520\,Z^7 - 764\,Z^6 + 831\,Z^5 + 898\,Z^4 - 722\,Z^3 + 985\,Z^2 - 516\,Z + 945$

13. $p(Z) = -526\,Z^{14} + 868\,Z^{13} - 745\,Z^{12} - 666\,Z^{11} + 810\,Z^{10} - 748\,Z^9 - 717\,Z^8 + 1016\,Z^7 + 1022\,Z^6 - 515\,Z^5 + 791\,Z^4 - 547\,Z^3 + 982\,Z^2 + 985\,Z - 692$

14. $p(Z) = -579\,Z^{15} - 540\,Z^{14} + 774\,Z^{13} + 779\,Z^{12} + 930\,Z^{11} + 934\,Z^{10} - 764\,Z^9 - 554\,Z^8 + 887\,Z^7 + 834\,Z^6 + 853\,Z^5 + 1004\,Z^4 + 940\,Z^3 + 827\,Z^2 - 559\,Z - 656$

15. $p(Z) = 802\,Z^{16} + 968\,Z^{15} - 710\,Z^{14} + 813\,Z^{13} - 720\,Z^{12} - 665\,Z^{11} - 523\,Z^{10} - 657\,Z^9 - 580\,Z^8 - 750\,Z^7 + 796\,Z^6 + 946\,Z^5 + 863\,Z^4 + 904\,Z^3 + 892\,Z^2 + 804\,Z + 818$

16. $p(Z) = -547\,Z^{17} + 804\,Z^{16} - 535\,Z^{15} - 629\,Z^{14} - 689\,Z^{13} - 552\,Z^{12} - 530\,Z^{11} - 688\,Z^{10} + 955\,Z^9 - 538\,Z^8 - 763\,Z^7 - 555\,Z^6 + 781\,Z^5 - 536\,Z^4 + 838\,Z^3 - 651\,Z^2 + 895\,Z - 572$

17. $p(Z) = -725\,Z^{18} - 564\,Z^{17} - 547\,Z^{16} - 539\,Z^{15} - 570\,Z^{14} + 813\,Z^{13} - 727\,Z^{12} + 899\,Z^{11} - 735\,Z^{10} + 784\,Z^9 + 590\,Z^8 - 575\,Z^7 + 792\,Z^6 + 908\,Z^5 - 577\,Z^4 + 996\,Z^3 + 833\,Z^2 - 714\,Z - 723$

18. $p(Z) = -525\,Z^{19} - 746\,Z^{18} - 765\,Z^{17} - 531\,Z^{16} + 835\,Z^{15} + 1009\,Z^{14} + 841\,Z^{13} - 530\,Z^{12} + 853\,Z^{11} + 864\,Z^{10} + 967\,Z^9 - 638\,Z^8 - 585\,Z^7 + 998\,Z^6 + 1007\,Z^5 - 731\,Z^4 - 539\,Z^3 - 595\,Z^2 - 708\,Z + 955$

19. $p(Z) = -678\,Z^{20} + 791\,Z^{19} - 699\,Z^{18} + 808\,Z^{17} - 755\,Z^{16} - 724\,Z^{15} + 966\,Z^{14} + 919\,Z^{13} - 759\,Z^{12} + 840\,Z^{11} - 641\,Z^{10} + 937\,Z^9 + 978\,Z^8 + 927\,Z^7 + 946\,Z^6 - 623\,Z^5 + 945\,Z^4 - 518\,Z^3 - 559\,Z^2 + 776\,Z + 927$

20. $p(Z) = -690 * Z^{21} + 929 * Z^{20} - 524 * Z^{19} + 873 * Z^{18} - 525 * Z^{17} + 807 * Z^{16} + 916 * Z^{15} + 962 * Z^{14} - 635 * Z^{13} + 854 * Z^{12} + 980 * Z^{11} - 562 * Z^{10} - 595 * Z^9 + 871 * Z^8 - 675 * Z^7 - 545 * Z^6 + 928 * Z^5 - 766 * Z^4 + 923 * Z^3 - 758 * Z^2 + 794 * Z - 522$

21. $p(Z) = -645 * Z^{22} + 790 * Z^{21} - 552 * Z^{20} + 875 * Z^{19} + 942 * Z^{18} - 573 * Z^{17} + 794 * Z^{16} + 928 * Z^{15} + 871 * Z^{14} - 639 * Z^{13} + 876 * Z^{12} + 857 * Z^{11} - 613 * Z^{10} - 729 * Z9 + 881 * Z^8 + 937 * Z^7 - 708 * Z^6 - 745 * Z^5 + 866 * Z^4 + 837 * Z^3 - 735 * Z^2 - 694 * Z - 674$

22. $p(Z) = -705\,Z^{23} - 685\,Z^{22} + 951\,Z^{21} + 838\,Z^{20} + 883\,Z^{19} + 884\,Z^{18} + 853\,Z^{17} - 684\,Z^{16} + 920\,Z^{15} + 955\,Z^{14} - 611\,Z^{13} + 918\,Z^{12} + 936\,Z^{11} - 583\,Z^{10} - 523\,Z^9 - 602\,Z^8 - 576\,Z^7 - 678\,Z^6 - 739\,Z^5 - 760\,Z^4 + 977\,Z^3 - 743\,Z^2 + 876\,Z + 1015$

23. $p(Z) = 892\,Z^{25} - 713\,Z^{24} - 538\,Z^{23} + 993\,Z^{22} + 914\,Z^{21} - 556\,Z^{20} - 626\,Z^{19} - 526\,Z^{18} + 776\,Z^{17} - 572\,Z^{16} - 621\,Z^{15} + 843\,Z^{14} - 688\,Z^{13} - 552\,Z^{12} + 1023\,Z^{11} + 938\,Z^{10} + 892\,Z^9 - 684\,Z^8 - 669\,Z^7 + 829\,Z^6 + 950\,Z^5 - 575\,Z^4 - 667\,Z^3 - 611\,Z^2 - 729\,Z - 595$

24. $p(Z) = -760\,Z^{30} - 550\,Z^{29} + 1002\,Z^{28} - 713\,Z^{27} + 859\,Z^{26} - 683\,Z^{25} + 776\,Z^{24} - 692\,Z^{23} - 604\,Z^{22} + 946\,Z^{21} - 619\,Z^{20} + 820\,Z^{19} + 915\,Z^{18} - 564\,Z^{17} - 757\,Z^{16} + 893\,Z^{15} - 631\,Z^{14} + 790\,Z^{13} - 536\,Z^{12} - 551\,Z^{11} + Z^{10} - 515\,Z^{9} - 624\,Z^{8} + 966\,Z^{7} - 525\,Z^{6} + 1022\,Z^{5} + 1019\,Z^{4} - 593\,Z^{3} + 967\,Z^{2} + 942\,Z + 968$

BIBLIOGRAFIA

[AKR 80] AKRITAS, A. G. The fastest exact algorithms for the isolation of the real roots of a polynomial equation. Computing, New York, v.24, p. 299-313, 1980.

[BAR 89] BARBEAU, E. J. Polynomials. New York:Springer-Verlag, 1989. 441p. (Problem Books in Mathematics).

[BEN 90] BEN-OR, M. Simple algorithms for approximating all roots of a polynomial with real roots. Journal of Complexity, v.6, p.417-442, 1990.

[BOH 87] BOHLENDER, G.; RALL, L. B.; ULLRICH, C.; WOLFF, G. PASCAL-SC: a computer language for scientific computation. Orlando: Academic Press, 1987. v 17, 295p.

[CAM 90] CAMARGO, M. A. O. Análise de algoritmos para determinação de raízes complexas de equações não lineares. Porto Alegre: CPGCC da UFRGS, 1990. 76 p. (Trabalho Individual, 204).

[CAM 93] CAMARGO, M. A. O., TREVISAN, V.; CLAUDIO, D. Counting polynomial zeros in a disk using Symbolic Computation. In: INTERNATIONAL CONFERENCE ON MATHEMATICAL MODELLING AND SCIENTIFIC COMPUTATION, 1993, Bulgária. Proceedings ... Sofia:DATECS, 1993. 191p. p.3-6.

[CLA79] CLAUDIO, D. M. Beiträge zur Struktur der Rechnerarithmetik. Karlsruhe: Universität Karlsruhe, 1979, 85p. (Dissertation).

[CLA 92] CLAUDIO, D. M. ; RUMP, S. M. Inclusion methods for real and complex functions in one variable. Hamburg: TUHH, 1992. 16p. (Berichte des Forschungsschwerpunktes Informations- und Kommunikationstechnik 92.5).

[COL 77] COLLINS, G. E. Infallible Calculation of Polynomial Zeros to Specified Precision. In: RICE, J. R. Mathematical Software III. New York:Academic Press, 1977, 388p., p.35-68.

[COL 82] COLLINS, G. E. ; LOOS, G. K. Real Zeros of Polynomial. Computing, Berlin, Suppl. 4, p.83-94, 1982.

[COL 92] COLLINS, G. E. ; KRANDICK, W. An efficient algorithm for infallible polynomial complex root isolation. In: INTERNATIONAL SYMPOSIUM ON SYMBOLIC AND ALGEBRAIC COMPUTATION ISSAC, 1992, USA. Proceedings ... [S.l.:s.n.],1992. p.189-194.

[DAV 87] DAVENPORT, H. H. Survey of Symbolic Applications for Numerical Computation. 1987. (Doc. 03/2-7/1/b01.f - DIAMOND).

[DAV 88] DAVENPORT, J. H.; SIRET, Y. ; Tournier, E. Computer Algebra: Systems and algorithms for algebraic computation. London: Academic Press 1988. 138p.

[DER 91] DEREN, W. ; FENGGUANG, Z. On the determination of the safe initial approximation for the Durand-Kerner algorithm. Journal of Computational and Applied Mathematics, Antwerp, Bélgica, v.38, p.447-456, 1991.

[DOL 90] DOLEH, Y. ; WANG, P. "SUI: A system independent user interface for an integrated scientific computing environment. In: INTERNATIONAL SYMPOSIUM ON SYMBOLIC AND ALGEBRAIC COMPUTATION,ISSAC, 1990, Toquio, Japão. Proceedings ... New York: ACM Press, 1991. 307p., p88-95.

[FRA 89] FRANGEN, W. Verified inclusion of all zeros of a complex polynomial by means of circular arithmetic. In: ULLRICH,C.;WOLFF,V.;GUDENBERG,J. (Eds) Accurate Numerical Algorithms, a collection of DIAMOND Research Papers. Berlin: Springer-Verlag, p.116-136. 1989. (Project 1072 - DIAMOND)

[FRE 91] FREEMAN, T. L. ; BANE, M. K. Asynchronous polynomial zero-finding algorithms. Parallel Computing, Netherlandes, v.17,p.673-681, 1991.

[GAT 84] GATES, G. ; WANG, P. A LISP-based Ratfor Code Generator. In: MACSYMA USERS CONFERENCE, 1984. Proceedings ...[S.l.:s.n], 1984, p.319-329.

[GEO 90] GEORG, S. Two Methods for the verified inclusion of zeros of complex polynomials. Contributions to computer arithmetic and Self-Validating numerical methods. In:IMACS WORLD CONGRESS ON SCIENTIFIC COMPUTATION, 12, 1988, França. Proceedings... Basel, J.C. Baltzer AG, 1990, 526p., p.229-244.

[GHA 90] GHADERPANAH, S. ; KLASA, S. Polynomial scaling. SIAM J. NUMERICAL ANALYSIS, v.27, n.1, p.117-135, 1990.

[HEA 90] HEARN, A. C. Future directions for research in symbolic computation. In: WORKSHOP ON SYMBOLIC AND ALGEBRAIC COMPUTATION, 1990, Washington D.C.. Proceedings ...California: SIAM, 1990.

[HEI 71] HEINDEL, L. E. Integer arithmetic algorithms for polynomial real zero determination. Journal of the ACM, New York, v.18, n.4, p.533-548, 1971.

[HEN 90] HENG, A. K. Some remarks on numerical iterations in a symbolic manipulations system. J. Symbolic Computation, v.10, p.209-221, 1990.

[HEN 74] HENRICI, Peter. Applied and computational complex analysis. New Jersey: John Willey & Sons, 1974, v.1, 682p.

[HEN 69] HENRICI, P. ; GARGANTINI, I. Uniformly convergente algorithms for the simultaneous approximation of all zeros of a polynomial.

In: DEJON, P. and HENRICI, P. Constructive aspects of the fundamental theorem of algebra, New York: Willey, 1969.

[HOE 92] HOENDERS, B. J., GRONINGEN, C. H. ; SLUMP, E. On the determination of the number and multiplicity of zeros of a function. Computing, New York, v.47, n.3-4, p.323-336, 1992.

[HUS 91] HUSUNG, D.TPX Precompiler für Turbo Pascal. Hamburg: TUHH, 1991. 46p. (Bericht des Forschungsschwerpunktes Informations und Kommunikationstechnik , 91.1).

[IBM 86] IBM. ACRITH, High-Accuracy arithmetic subroutine library: Program description and user's guide. [S.l]:IBM, 1986. (Document Number SC33-6164-3).

[KOR 90] KORN, C. E. ; GUTZWILLER, S. E. ; KÖNING, S. Modula-SC, a Precompiler to Modula-2. In: IMACS WORLD CONGRESS ON SCIENTIFIC COMPUTATION,12 1988, Paris, França. Proceedings ... Basel, Suiça:J. C. Baltzer AG, 1990, 526 p., p.371-383.

[KUL 86] KULISCH, U. W. ; MIRANKER, W. L. The arithmetic of the digital computer: a new approach. SIAM Review, v.28, n.1, p.1-40, 1986.

[LEH 61] LEHMER, D. H. A machine method for solving polynomial equations. Journal of ACM, New York, v.8, p.151-162, 1961.

[MAR 49] MARDEN, M. The Geometry of the zeros of a polynomial in a complex variable. New York:AMS, 1949 183p. (American Mathematical Society Mathematical Surveys III)

[MET 90] METZGER, M. ; WALTER, W. FORTRAN-SC - A programming language for engineering/scientific computation. In:IMACS WORLD CONGRESS ON SCIENTIFIC COMPUTATION, 12, Pa-

ris, França. Proceedings ... Basel, Suiça: J. C. Baltzer AG, 1990, 526p., p427-441.

[NIC 91] NICOLAYEV, I. A., KRUKIER, L. A., Zharinov, S. A. Complex application of graphical, symbolic and numerical methods in packages for solving mathematical modeling problems. In: INTERNATIONAL FEDERATION FOR INFORMATION PROCESSING:IFIP WORKING CONFERENCE ON PROGRAMMING ENVIRONMENTS FOR HIGH-LEVEL SCIENTIFIC PROBLEM SOLVING, Karlsruhe. Proceedings ... Karlsruhe:IBM, 1991, 284p., p.280-284, 1991.

[NOD 91] 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.

[OGA 82] OGATA, K. Engenharia de Controle Moderno. São Paulo: Prentice/Hall do Brasil, 1982. 929p.

[OLI 92] OLIVEIRA, J. B. S. Uma ferramenta para determinação de zeros polinomiais. Porto Alegre: CPGCC da UFRGS, 1992. 104p. (Dissertação de Mestrado)

[PAL 91] PALDI, E. The intersection of regions and root clustering. Linear Algebra and its applications. v.149, p.151-164, 1991.

[PET 91a] 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.

[PET 91b] 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.

[PIN 76] 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.

[PRE 90] PRESS, W. H., FLANNERY, B. P., TEUKOLSKY, S. A. and VETTERLING, W. T. Numerical Recipies in C: The art of Scientific Computation. New York: Cambridge University Press, 1988. 735p.

[RUM 79] RUMP, S. M. Polynomial minimum root separation. Mathematics of Computation, v.33, n.145, p.327-336, 1979.

[RUM 83] RUMP, S. M. Solving algebraic problems with high accuracy. In: Kulisch, U.; Miranker, W. L. (eds). A new approach to scientific Computation . New York: Academic Press, 1983. p.51-120. 384p.

[RUM 87] RUMP, S. M. Algebraic computation, numerical computation and verified inclusions. 1987, p.177-197. (Lectures Notes in Computer Science - Trends in Computer Algebra).

[SAK 91] SAKURAI, T. TORII, T. ; SUGIURA, H. A high-order iterative formula for simultaneous determination of zeros of a polynomial. Journal of Computation and Applied Mathematics, Antwerp, Bélgica, v.38, 1991.

[SCH 90] SCHRODER, U. Verification of polynomial roots by closed coupling of computer algebra and self validating numerical methods. IMACS WORLD CONGRESS ON SCIENTIFIC COMPUTATION, 12, 1988, Paris, França. Proceedings ... Basel, Suiça: J. Baltzer AG, 1990. 526p., p.259-269.

[SIE 86] Siemens Ag, ARITHMOS - Benutzerhandbuch. Bestellnummer U 2900-J-Z87-1 , 1986. (Relatório Técnico)

[SUZ 87] SUZUKI, M. ; SASAKI, T. A Hybrid algebraic-numeric system ANS and its preliminary implementation. [S.l.:s.n.], 1987. p.163-171. (Lectures notes in Computer Science, Symbolic and Algebraic Computation)

[TRE 90] TREVISAN, V. Recognition of Hurwitz polynomials. SIGSAM Bulletin, New York, v.24, n.4, p.26-32, Oct. 1990.

[TRE 91] TREVISAN, V. Computação Algébrica e Simbólica. Boletim do SBMAC, São José dos Campos, S.P., v.2, n.2, p.1-18, 1991.

[TRE 92] TREVISAN, V. Computação Algébrica e Simbólica. In: MINICURSO - XV CNMAC, 1992, São Carlos, SP. Anais ... São Carlos,SP, 1982. 96p.

[TUR 91] TURNER, P. R. Will the Real Arithmetic Please Stand Up? Computers and Mathematics. Notices of the American Mathematical Society. v.38,n.4, p.298-304, 1991.

[UHL 92] UHLIG, F. Are the coefficients of a polynomial well-conditioned functions of its roots? Numerische Mathematik, Secaucus, New Jersey, v.61, 1992.

[USP 48] USPENSKY, J. V. Theory of Equations, New Jersey: McGraw-Hill, 1948. 353p.

[VAS 87] VASILIEV, N. N. Creation of efficient symbolic-numeric interface. [S.l.:s.n], 1987. p.118-119 (Lectures Notes In Computer Science, Symbolic and Algebraic Computation)

[WAN 89] WANG, P. ; WEERAWARANA, S. GENCRAY: A portable code generator for Cray Fortran. In: INTERNATIONAL SYMPOSIUM ON SYMBOLIC AND ALGEBRAIC MANIPULATION, 1989, Portland Oregon. Proceedings... Portland Oregon: [s.n.], 1989, p.186-191.

[WIL 78] 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, July 1978.

[WOL 90] WOLF,J. G. A Symbolic generic expression concept. In: IMACS WORLD CONGRESS ON SCIENTIFIC COMPUTATION, 12, 1988, Paris, França. Proceedings ... Basel, Suiça: J. C. Baltzer AG, 1990. 526p, p459-464.

[ZEH 91] ZEHEB, E. On the Largest Modulus of Polynomial Zeros. IEEE Transactions on Circuits and Systems, New York, v.38, n.3, p.333-337, Mar. 1991.

Informática
UFRGS

Algoritmos Algébricos para Enumerar e Isolar Zeros Polinomiais Complexos.

Tese apresentada aos Senhores:

Prof. Dr. Etzel Ritter von Stockert (UFSC)

Profa. Dra. Laira Vieira Toscani

Prof. Dr. Waldir Leite Roque (Instituto de Matemática/UFRGS)

Vista e permitida a impressão.
Porto Alegre, 11 / 10 / 1994

Prof. Dr. Dalcidio Moraes Claudio,
Orientador.

Prof. Dr. Vilmar Trevisan (Instituto de Matemática-UFRGS),
co-orientador.

Prof. Dr. José Palazzo Moreira de Oliveira,
Coordenador do Curso de Pós-Graduação
em Ciência da Computação.