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

Uma ferramenta para determinação de zeros polinomiais

por

João Batista S. Oliveira

Dissertação submetida como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dalcidio M. Claudio
Orientador

Porto Alegre, março de 1992.

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

"Uma ferramenta para determinação de zeros polinomiais".

Dissertação apresentada aos Srs.

Prof. Dr. Alwin Elbern (DENUC/UFRGS)

Profa. Carla Maria Dal Sasso Freitas

Prof. Dr. Dalcidio Moraes Claudio

Prof. Dr. Julio Cézar Ruiz Clayssen (DMPA/UFRGS)

Visto e permitida a impressão

Porto Alegre, 29 / 04 / 92

Prof. Dr. Dalcidio Moraes Claudio
Orientador

P/Prof. Dr. Ricardo Augusto da L. Reis
Coordenador do Curso de Pós-Graduação
em Ciência da Computação

ESR/CCB

\DIVCPGCC\ASSINA.BAN

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Oliveira, João Batista S.

Uma ferramenta para determinação de zeros polinomiais / João Batista S. Oliveira.—Porto Alegre: CPGCC da UFRGS, 1992.

104 p.: il.

Dissertação (mestrado)—Universidade Federal do Rio Grande do Sul, Curso de Pós-Graduação em Ciência da Computação, Porto Alegre, 1992. Orientador: Claudio, Dalcidio M.

Dissertação: Matemática Computacional
Determinação de zeros

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

Este trabalho define um ambiente que auxilia o usuário a determinar as raízes complexas de um polinômio, dados apenas seus coeficientes complexos. Para tanto é definida uma interface que habilita a comunicação e a apresentação de resultados de forma mais expressiva do que as habitualmente usadas, permitindo que o usuário perceba mais rapidamente as informações que a ferramenta lhe apresenta. Para capacitar o usuário a determinar as soluções o sistema faz uso de uma abordagem baseada em imagens, obtidas a partir da interpretação do polinômio sendo estudado como se fosse uma superfície originada por uma função de duas variáveis. Através do uso de imagens o usuário pode orientar o sistema para determinar as soluções de seu interesse particular, e pode adicionalmente perceber de forma muito clara detalhes do polinômio que não são apresentados por outras ferramentas.

São definidas operações que facilitam a manipulação das imagens obtidas, podendo-se efetuar uma série de computações que eventualmente dispensam os métodos de determinação numérica de raízes, obtendo-se de uma forma alternativa possíveis aproximações para as soluções. Esta abordagem apoiada em imagens é muito útil na determinação de raízes em regiões onde a avaliação do polinômio se mostra instável ou sujeita a erros grosseiros, como por exemplo raízes múltiplas. Quando são geradas imagens pode ser percebida claramente a presença de erros de avaliação, permitindo que o usuário tome a atitude que julgar adequada. Outros sistemas não permitem que tais instabilidades sejam notadas, fornecendo respostas que podem não ter qualquer nexo com a realidade.

Para uso na ferramenta que implementa a solução proposta, existe uma série de comandos definidos que capacitam o usuário a fazer quaisquer operações de seu interesse. Estes comandos foram já testados no período da implementação do sistema e permitem que sejam feitas operações sobre o polinômio, as imagens geradas e as raízes determinadas. Em todo o projeto do sistema existe a preocupação de tornar claro o que está acontecendo a cada momento com o polinômio ou com os dados dele obtidos, para que o usuário não

se canse em demasia, seja interpretando dados do sistema ou procurando decidir o que fazer em seguida. Assim, o uso da ferramenta é bastante natural, e tenta atingir a intuição daqueles que a irão usar.

Exemplos são apresentados, bem como descrições dos comandos utilizados na interface. De especial interesse são os algoritmos usados na construção das imagens, pois estes permitem uma grande economia nas avaliações do polinômio quando da geração das cenas, reduzindo sobremaneira o tempo de espera do usuário. Quanto às imagens, são fornecidos três tipos de representações: imagens coloridas em duas dimensões, curvas de nível e curvas de sinal. O uso de três dimensões nas imagens foi descartado, e as razões para tanto estão descritas no texto. Todos os tipos de imagens permitem que se determine e identifique com facilidade as soluções desejadas, e são construídas a partir de uma transformação sobre o polinômio original. Esta transformação é explicada em detalhe no transcorrer do texto.

Palavras-chave: matemática computacional, determinação de zeros

ABSTRACT

This work describes an environment that helps the user to find the complex solutions of a polynomial, given its coefficients. To do so, a quite expressive user interface is defined providing easy communication between man and machine and also allowing users to perceive much more rapidly the informations given by the tool. The system uses an image-based interface, obtained from a two-variable surface defined by the polynomial. By using these pictures, it is possible to direct the tool, finding any solutions of special meaning to the user. In addition, some details and features of the polynomial can be very easily seen.

To manipulate images a set of operations was defined, and this set has so many features that the use of some operations can even avoid the effort of numerically computing the zeros of the polynomial, also reducing the numerical error embedded in these computations. This image-based approach is very useful when finding domains where the evaluation of the polynomial is unstable or prone to errors, e. g., near multiple roots. In pictures, any evaluation problems are easily detected, and the user can decide how to take the answers given by the system. Other tools usually do not provide any means to detect such behavioral oscillations, and can give nonsense answers.

There is a set of commands embedded on the tool that implements the proposed solution, and they allow the user to perform any operations of his (her) interest. Such commands were tested at implementation time, and all were useful at some moment. They perform operations on the polynomial, on the pictures and on the detected zeros. All over the design phase there was the care of making clear what is happening to the polynomial or any data, so that the user can always work without being stressed by trying to find out what is happening. So, we try to build a natural approach to this interfacing problem, trying to reach user's intuition.

Examples and descriptions of the commands are given, and also there is a description of a quite important algorithm, the one that builds the pictures themselves.

This algorithm needs quite few evaluations of the function to generate pictures, thus reducing the waiting times. For the images, there are three presentation types: color images in two dimensions, height curves or signal curves. Three-dimensional pictures were discarded, and the reasons to such decision are described on the text. All kinds of images provide easy and simple identification of solutions. The images are produced from a simple transformation of the starting polynomial. This transformation is also explained in detail.

Keywords: computational mathematics, zero finding

1 INTRODUÇÃO

À medida que o uso de dispositivos automáticos de computação foi tornando-se popular com o emprego em larga escala de máquinas digitais, cresceu cada vez mais o número de problemas que poderiam ser investigados e resolvidos pela comunidade científica. Questões antes quase insolúveis, fosse pelo tempo de computação, fosse pela complexidade das tarefas necessárias a seu bom termo ou por causa das massas de dados a serem processadas, são hoje lugar comum em ambientes de pesquisa. O meio científico é atualmente capaz de solucionar problemas com ordens de grandeza nunca antes imaginadas, em tempo irrisório se comparado com tecnologias de outras épocas.

Se uma nova e revolucionária tecnologia foi introduzida, também sua terminologia e os conceitos que a fundamentam foram popularizados, perdendo muito de seu mistério inicial e deixando de ser domínio de uns poucos especialistas para tornar-se familiares ao público. Tal popularização permitiu que usuários com necessidades diversas fizessem exigências que direcionam ainda hoje o desenvolvimento de software específico para a solução de seus problemas. Aproveitando a disseminação desta tecnologia entre outros ramos de pesquisa, este trabalho objetiva explorar a utilização de computadores fazendo uso de conceitos e vantagens por eles introduzidos para resolver (ou no mínimo simplificar) a resolução de uma classe de problemas que, embora definida de uma maneira bastante simples, continua intratável sob uma abordagem analítica.

Desta forma pretende-se fazer uso das vantagens oferecidas para criar uma alternativa de solução para um problema ainda à espera de uma forma realmente simples de ser resolvido, ou seja, ainda aguardando o surgimento de uma solução de fácil utilização por parte dos usuários não experientes ou leigos no assunto. Adicionalmente, a comunidade que possui intimidade com o problema teria uma forma opcional de solução, podendo utilizá-la eventualmente.

A solução oferecida deve ser de uso simples e precisa fornecer ao usuário uma interface tão natural que ele não se sinta inibido ao usá-la, nem seja forçado a aprender

uma série de novos conceitos que lhe são estranhos para aproveitar completamente a ferramenta que implementa a solução.

Em termos simples, a proposta é usar computadores para obter aproximações (as melhores possíveis) para as raízes complexas de um polinômio qualquer $p(z)$, fornecido na forma abaixo:

$$p(z) = \alpha_n z^n + \alpha_{n-1} z^{n-1} + \cdots + \alpha_1 z + \alpha_0 \qquad \text{com } z, \alpha_i \in \mathbb{C}, i = 0(1)n.$$

Com o propósito de resolver o problema exposto acima, serão definidas quais as características consideradas essenciais para o bom funcionamento do sistema, seja no aspecto computacional (numérico), seja no aspecto comunicacional (interface). Estes são, respectivamente, os propósitos dos capítulos 2 e 3. Em seguida, o capítulo 4 faz uma descrição das operações implementadas sobre a plataforma definida (interface e métodos numéricos). Exemplos de utilização e considerações sobre desempenho serão expostos nos capítulos 5 e 6 respectivamente, seguidos nos capítulos 7 e 8 por dois tópicos especialmente abertos à imaginação, quais sejam, o tratamento de equações não polinomiais e as possíveis expansões futuras segundo a mesma abordagem usada para polinômios.

São apresentadas logo depois (capítulos 9 e 10) as considerações gerais decorrentes do desenvolvimento deste trabalho[1] e as conclusões finais. O anexo A-1 descreve uma ferramenta auxiliar no desenvolvimento e teste do sistema final.

[1] incluindo-se aqui a descrição de particularidades ou características que de alguma forma possam ser consideradas interessantes ou inesperadas na fase de projeto inicial.

2 CARACTERÍSTICAS NUMÉRICAS

Este capítulo tem por objetivo definir, analisar e detalhar com maior profundidade o problema a resolver, indicando possíveis alternativas de solução. Especialmente, os detalhes de natureza computacional serão analisados, pois que a parte analítica do problema está firmemente estabelecida, não apresentando quaisquer avanços de nosso particular interesse desde Abel, que mostrou a impossibilidade de determinar de forma analítica as raízes de um polinômio de grau maior do que quatro (veja [USP48]). Sendo assim, precisa-se tentar contornar a intratabilidade algébrica através de uma forma robusta de algoritmo que permita segurança quanto à exatidão dos resultados. Em síntese, deve-se definir de forma clara o problema, a fim de se buscar soluções que possam ser implementadas com a tecnologia que se nos oferece. Conhecendo previamente as limitações intrínsecas ao problema a resolver, será possível compreender e quantificar a qualidade de uma possível solução.

2.1 Dados iniciais e um algoritmo potencial

O objetivo final é obter aproximações[1] para as raízes de um polinômio, de uma forma ainda a ser determinada. Para obter-se um melhor julgamento de quais as possíveis maneiras de obter tais aproximações ou quais as melhores formas de abordagem para um algoritmo de solução, pode-se sumarizar os conhecimentos antes de se tomar uma decisão, e em seguida avaliar as alternativas coerentes para a realização da tarefa. As diretrizes gerais do problemas estão descritas a seguir:

[1] Durante o decorrer do texto, os termos *aproximações*, *raízes*, *soluções*, *respostas* ou ainda *resultados* serão usados de maneira equivalente: todos cinco indicam os valores numéricos obtidos usando-se um computador para solucionar o problema. Assim, devido à existência de uma representação finita para os números reais, surgirão erros que fazem com que os valores obtidos apenas **aproximem** os verdadeiros resultados. Ao planejar-se o processo de obtenção destes resultados, deve ser sempre lembrada sua imprecisão latente.

1. O polinômio possui coeficientes complexos e variável também complexa. Uma das conseqüências desta premissa é que as raízes complexas não necessariamente se apresentam aos pares (raiz e seu conjugado), e por isso não pode ser assumido que para cada raiz complexa seu conjugado seja também uma raiz do polinômio. As raízes devem ser portanto procuradas isoladamente, ou de forma independente umas das outras.

2. Somente existe o conhecimento dos coeficientes de $p(z)$, sem qualquer outro tipo de informação. Não há pois outros dados sobre regiões que possam conter as raízes ou possíveis valores para estas raízes.

3. Sabe-se de um resultado estabelecido (para detalhes, veja [USP48, BAR89]) que afirma não existir maneira de obter tais raízes por um número finito de operações algébricas, se o polinômio possuir grau maior do que quatro. Como decorrência desta afirmação, é necessário assumir que inexiste uma forma analítica para a resolução de polinômios quaisquer.

4. As operações serão feitas sobre *aproximações* dos números reais, ou seja, gerando resultados a partir de uma representação errônea, que não permite garantir totalmente a acuidade das respostas finais.

Deve ser notado como especialmente relevante o terceiro item, que impossibilita o uso de uma abordagem analítica para o caso genérico de um polinômio com grau qualquer. De forma definitiva, este fato conduz ao uso de algum tipo de método numérico para a determinação das soluções. Uma consulta aos algoritmos mais utilizados aponta para o método de Newton como uma forma razoavelmente segura de encontrar possíveis soluções. Posteriormente, pode ser usado o método de Laguerre para refinar as respostas encontradas previamente. Para uma descrição mais detalhada destes dois métodos de solução de equações polinomiais, veja [USP48, BAR89, PRE88]. Para uma descrição bastante interessante destes e de outros métodos numéricos para a determinação de raízes, consulte [HEI84].

É bastante importante o fato de não se possuir nenhuma outra informação adicional além dos coeficientes do polinômio e portanto seria desejável uma maneira de determinar-se pelo menos uma região inicial onde as raízes estariam contidas. Isto feito, torna-se necessário pesquisar somente o interior desta região, ou tendo em vista que optou-se por usar o método de Newton, usar uma estimativa inicial dentro desta "zona de probabilidade". De novo, faz-se uso de um método já estabelecido para achar a região inicial, pois este problema constitui-se em tópico importante de muitos livros que tratam de equações polinomiais, onde usualmente trata-se de encontrar cotas para as raízes das equações em estudo. Para um tratamento deste assunto de forma bastante rica em exemplos de cotas, consulte [HEI84]. Dentre as cotas apresentadas nesta obra, foi selecionada a cota de Cauchy, por sua relativa facilidade de avaliação e sua satisfatoriedade quanto ao objetivo de se obter cotas reduzidas, que diminuem a zona a ser pesquisada.

Partindo destes três métodos, pode ser elaborado o seguinte esquema para determinar o conjunto de soluções de um polinômio $p(z)$ de grau n:

1. Determinação de uma cota inicial, através da cota de Cauchy.
2. Criação de um polinômio inicial $p_1(z) \equiv p(z)$
3. Repetição dos seguintes passos k vezes ($1 \leq k \leq n$):
   - Dentro da região inicial dada pela cota, escolhe-se um ponto qualquer para ser valor inicial da iteração de Newton. É efetuada a iteração sobre o polinômio $p_k(z)$. Como este polinômio pode eventualmente conter erros devido a deflações sucessivas (ver adiante), será necessário um passo posterior de refino.
   - Terminada a iteração de Newton, o resultado parcial obtido será usado para a iteração de Laguerre, com a finalidade de refinar o resultado prévio. Esta iteração deve ser feita sobre $p(z)$, evitando assim quaisquer erros contidos em $p_k(z)$.
   - Terminada a iteração de Laguerre, o resultado final $\overline{x}$ é armazenado e deflaciona-se $p_k(z)$, dividindo-o por $(x - \overline{x})$ e obtendo $p_{k+1}(z)$.

Sobre o esboço de algoritmo acima, pode ser notado que são feitas deflações sucessivas sobre o polinômio inicial $p_1(z)$. Estas deflações fazem com que $p_k(z)$ afaste-se aos poucos do original $p(z)$, e portanto as estimativas obtidas pela iteração de Newton serão cada vez mais inexatas, embora normalmente possam ser consideradas razoáveis. Como correção, pode-se incluir um passo de ajuste composto por iterações de Laguerre, partindo deste valor inicial incerto fornecido pelo método de Newton e corrigindo-o com algumas iterações sobre o polinômio original $p(z)$.

A etapa de deflação, apesar de induzir um certo erro em $p_k(z)$, tem grande utilidade dentro do algoritmo. Em primeiro lugar, ela diminui o grau de $p_k(z)$, melhorando o desempenho das iterações de Newton. Em segundo lugar, a deflação evita que uma raiz seja encontrada duas vezes, garantindo que não haverá perda de tempo encontrando-se a mesma raiz várias vezes seguidas.

2.2 Cauchy, Newton, Laguerre

Embora não seja o mérito deste trabalho entrar em considerações sobre a construção dos métodos numéricos utilizados, é válido fazer uma descrição breve dos seus princípios de funcionamento, para que o leitor que não os conhece possa tornar-se familiarizado. Durante a descrição dos métodos, os termos α_k serão referentes aos coeficientes do polinômio $p(z)$, definido por

$$p(z) = \alpha_n z^n + \alpha_{n-1} z^{n-1} + \cdots + \alpha_1 z + \alpha_0 \qquad \text{com } z, \alpha_i \in \mathbb{C}, i = 0(1)n.$$

2.2.1 Cotas de Cauchy

Existem na verdade duas cotas que são chamadas cotas de Cauchy. Uma destas cotas se baseia em um método iterativo, e a outra, bem mais simples, faz uma avaliação sobre os coeficientes de $p(z)$. Ambas as cotas são encontradas em [HEI84, MAR49, MAR66]. Como a cota calculada iterativamente fornece resultados bem melhores do que a cota não-iterativa, optamos por fazer as iterações. Portanto, o método é o seguinte: dado $p(z)$, este é transformado em um polinômio auxiliar $q(z)$. Segundo Cauchy, o módulo da maior raiz real de $q(z)$ será uma cota para todas as raízes de $p(z)$. A seguir, está descrito o processo de obtenção de $q(z)$ e da fórmula de iteração, partindo-se dos coeficientes de $p(z)$:

$$q(z) = z^n + |\beta_{n-1}|z^{n-1} + |\beta_{n-2}|z^{n-2} + \cdots + |\beta_1|z + |\beta_0| \qquad \text{com } \beta_i = \frac{\alpha_i}{\alpha n}, i = 0(1)n-1.$$

Supõe-se que z é uma raiz de $q(z)$, e portanto $q(z) = 0$. Sendo assim, pode-se separar a potência z^n do resto de $q(z)$, obtendo-se

$$z^n = -(|\beta_{n-1}|z^{n-1} + |\beta_{n-2}|z^{n-2} + \cdots + |\beta_1|z + |\beta_0|)$$

Sendo uma fórmula iterativa o resultado será considerado como sendo o valor de z_{i+1}, obtendo-se portanto a expressão da iteração em função da estimativa atual z_i e obtendo-se a próxima estimativa z_{i+1}:

$$z_{i+1} = (-(|\beta_{n-1}|{z_i}^{n-1} + |\beta_{n-2}|{z_i}^{n-2} + \cdots + |\beta_1|z_i + |\beta_0|))^{1/n}$$

Em princípio, qualquer valor de z_0 pode ser usado, convergindo-se para o valor da cota de $p(z)$. Geralmente efetua-se uma série de iterações, e depois o resultado final

é aumentado ligeiramente. Para uso no algoritmo, deverão ser feitas 200 iterações e em seguida o valor resultante é aumentado em 5%. O número de iterações deve ser suficiente para assegurar a convergência, e o ligeiro aumento posterior tem por função cobrir uma eventual falha na convergência, bem como assegurar que todas as raízes estão contidas na tela, e nenhuma está localizada sobre as bordas.

2.2.2 Método de Newton

O método de Newton é um dos métodos mais difundidos de determinação de raízes não só para polinômios, e descrições detalhadas podem ser encontradas em [HEI84, BAR89, USP48, PRE88, ACT70]. O método baseia-se em uma fórmula de iteração originária da expansão de $p(z)$ em uma série de Taylor da seguinte forma:

$$p(w) = p(z) + p'(z)(w - z) + \cdots$$

Desprezando-se os termos posteriores à derivada primeira, e fazendo-se a suposição de que w é uma raiz de $p(z)$, chega-se à seguinte forma simplificada:

$$0 = p(z) + p'(z)(w - z)$$

Após algumas manipulações algébricas simples, obtém-se:

$$w = z - \frac{p(z)}{p'(z)}$$

Neste ponto, pode-se supor que z é um valor inicial, uma aproximação para uma raiz. Em seguida, w passa a ser considerada a próxima estimativa, e assim sucessivamente. Portanto na sua forma final a iteração de Newton é:

$$z_{i+1} = z_i - \frac{p(z_i)}{p'(z_i)}$$

Nesta iteração, ao contrário da iteração dada por Cauchy, o valor inicial z_0 é bastante importante quando se deseja *uma determinada raiz*. Ou seja, se o sistema fosse à procura de uma raiz em especial, o valor de z_0 deveria ser escolhido cuidadosamente para que a convergência para a raiz desejada fosse assegurada. No caso do algoritmo que está sendo desenvolvido esta preocupação não existe, pois não importa a ordem em que as raízes são encontradas já que deve-se determinar a todas de qualquer forma. O critério de parada para o método de Newton será detalhado na seção 2.3.

2.2.3 Método de Laguerre

O método de Laguerre também é um método iterativo, que deve obrigatoriamente ser usado com polinômios, ao contrário do método de Newton que pode ser usado com quaisquer equações. Também iterativo, ele parte de uma estimativa inicial (no algoritmo proposto, fornecida previamente pelo método de Newton) e refina-a. Neste ponto faremos um desenvolvimento bastante simplificado do método, pois este é bastante mais complexo do que o desenvolvimento aqui apresentado. Para operar, o algoritmo baseia-se na representação de $p(z)$ como

$$P_n(z) = (z - z_1)(z - z_2)\ldots(z - z_n)$$

onde os valores z_k são as raízes de $p(z)$. Extraindo-se os logaritmos dos valores absolutos, teremos

$$\ln|P_n(z)| = \ln|z - z_1| + \ln|z - z_2| + \ldots + \ln|z - z_n|$$

Avaliando-se a derivada da expressão acima, teremos:

$$\frac{d\ln|P_n(z)|}{dz} = \frac{1}{|z-z_1|} + \frac{1}{|z-z_2|} + \ldots + \frac{1}{|z-z_n|} \equiv G$$

De forma similar, a derivada segunda seria:

$$\frac{d^2\ln|P_n(z)|}{dz^2} = \frac{1}{(z-z_1)^2} + \frac{1}{(z-z_2)^2} + \ldots + \frac{1}{(z-z_n)^2} \equiv H$$

Definidos G e H, supomos que existe uma estimativa inicial z, que serve para a raiz z_1. A diferença entre z e z_1 é chamada a. Supomos adicionalmente que a distância de z para *todas* as outras raízes é b:

$$a = z - z_1$$
$$b = z - z_i, i = 2, 3, \ldots, n$$

Depois desta suposição, as fórmulas para G e H passam a ser:

$$\frac{1}{a} + \frac{n-1}{b} = G$$
$$\frac{1}{a^2} + \frac{n-1}{b^2} = H$$

Isolando-se a, chega-se na expressão

$$a = \frac{n}{G \pm \sqrt{(n-1)(nH - G^2)}}$$

O sinal deve ser escolhido de forma que seja obtido o maior denominador. Eventualmente o valor da raiz quadrada pode ser complexo, e conseqüentemente o valor

de a será complexo. Uma vez que a está calculado, $z - a$ é a próxima estimativa e o processo continua. A parada é feita quando a for considerado suficientemente pequeno. Em nosso algoritmo, este processo é repetido para todas as raízes, usando estimativas iniciais fornecidas pelo método de Newton.

2.3 Critérios de parada

Dentro do algoritmo proposto estão dois métodos que exigem algum controle sobre o número de iterações a ser feito, a saber, os métodos de Newton e de Laguerre. Ambos podem ser terminados por meio de um controle sobre a quantidade de iterações já feitas ou pelo acompanhamento da exatidão atingida a cada iteração. Infelizmente a escolha de somente uma condição de parada não é completamente segura, e usualmente é implementada uma forma qualquer de combinação entre os dois sistemas. Seguindo esta prática, opta-se por terminar as iterações de ambos os métodos ao ser atingido um limite de iterações ou até ser satisfeito um critério de exatidão.

Para obter maior flexibilidade, o usuário deve poder determinar o número máximo de iterações a ser feito para cada método. Evidentemente, poderá também alterar o critério de exatidão a ser respeitado. No entanto, este critério deve necessariamente ser plenamente compreendido antes que se façam tentativas de alterá-lo.

O critério de exatidão usado é o seguinte: o algoritmo (Newton ou Laguerre) recebe uma tolerância ϵ. Ora, ambos os algoritmos baseiam-se em iterações que corrigem o valor z_i de uma iteração através da adição de um certo δ_i calculado pelo método, obtendo-se z_{i+1}.

O critério de parada da execução do método em questão é satisfeito sempre que $|\delta_i| \leq \epsilon$. Assim definido, o controle de exatidão funciona como uma forma de interromper a execução do método em uso quando as correções de cada iteração não são significativas

o suficiente. A escolha de ϵ fica a critério do usuário, que poderá alterá-lo como achar mais vantajoso. Valores típicos são da ordem de 10^{-9}.

O critério adotado possui a vantagem de assegurar a parada dos algoritmos, pois à medida que o método usado convergir, o valor de δ tenderá a zero. Em testes, foram usados valores como 10^{-70} sem problemas. É claro que estes valores exageradamente pequenos consomem mais tempo para terminar as iterações, pois o sistema deve dispender mais tempo até satisfazer a condição.

Deve ser notado que a cota de Cauchy é calculada somente uma vez durante toda a execução do algoritmo, embora o polinômio que está sendo usado seja alterado a cada raiz encontrada. Admite-se que existam pequenas flutuações nos valores das raízes a cada deflação feita, mas existe a suposição de que o erro induzido pelas deflações é pequeno o bastante para que as cotas iniciais ainda sejam uma boa aproximação para as cotas dos polinômios resultantes das deflações. A validade desta suposição baseia-se no fato de que se C é cota de $p(z)$, então C é também cota de $r(z) \equiv \frac{p(z)}{q(z)}$, pois $r(z) = 0 \Leftrightarrow p(z) = 0$.

De qualquer forma, o método de Newton não está limitado à região das cotas, podendo sair delas durante o processo iterativo. As cotas são portanto usadas somente como uma aproximação inicial de onde podem ser retiradas estimativas iniciais para o método de Newton. Ao final desta discussão existe um esboço de algoritmo que vai ser refinado posteriormente, depois da análise das características comunicacionais desejadas no software. Uma versão final do algoritmo deverá levar em conta as facilidades de comunicação entre a aplicação e o usuário. A seguir serão discutidas e detalhadas as necessidades do sistema em matéria de comunicação com o usuário e apresentação do problema de forma realmente efetiva, especialmente ao usuário leigo.

3 INTERFACE PROPOSTA

Neste capítulo descreve-se a interface a ser usada, desde suas concepções básicas até considerações a respeito de seu desempenho. Em primeiro lugar, uma digressão sobre o ambiente objetiva investigar quais as características mais desejáveis em um computador, dentro do escopo das finalidades e requisitos. Em seguida, são feitas considerações sobre estas características e uma análise das formas de utilização destas para construir um ambiente de trabalho que se apresente de forma amigável ao usuário. Um passo importante é a transformação do problema, adaptando-o para ser melhor representado no equipamento disponível e transmitindo uma maior quantidade de informação ao usuário.

3.1 O ambiente disponível

A existência das mais variadas plataformas de software e hardware torna a escolha de um determinado computador tarefa bastante complexa, dada a disparidade de possibilidades existentes. Uma mesma máquina pode, com diferentes sistemas operacionais, oferecer um grande número de capacidades ou facilidades que tornam a escolha de um determinado ambiente um compromisso quase inalterável *a posteriori*. Portanto, a escolha de uma máquina deve ser feita de maneira criteriosa, levando em conta a possibilidade de eventuais expansões futuras ou novos requisitos não previstos no projeto original.

É objetivo deste trabalho definir e implementar uma ferramenta que proporcione ao usuário facilidades para operar sobre polinômios, e para tanto deve ser escolhida uma plataforma capaz de fornecer as operações primitivas suficientes e adequadas à realização desta tarefa. Dentro das facilidades que se oferecem para a implantação desta ferramenta e de acordo com os objetivos e necessidades deste trabalho, existem hoje em dia nesta Universidade dois tipos principais de plataformas: microcomputadores pessoais do

tipo IBM-PC em várias configurações ou ainda estações de trabalho, predominantemente de marca SUN. Tais ambientes são muito diferentes e praticamente excludentes, ou seja, a portabilidade de software entre estas máquinas se restringe a uns poucos casos, onde não é usada qualquer capacidade mais sofisticada tal como gráficos, som, cor, etc. Deve-se considerar portanto o software como não-portável em geral, embora existam exceções. Nas duas possíveis plataformas, pacotes completos e documentados de software possibilitam a implementação de capacidades específicas, e depois da análise dos requisitos necessários ao software proposto será necessário optar por uma delas. A seguir, faz-se uma série de conjecturas a respeito destes requisitos.

3.2 Interface: saída

Essencialmente, deve ser feito um esforço para tornar o diálogo homem-máquina o mais natural possível para o homem, mesmo exigindo um trabalho maior por parte da máquina. Assim, as facilidades projetadas devem prover ou capacitar a fácil compreensão das ações que a aplicação está apta a executar. Em outras palavras, o sistema deve dar ao usuário a capacidade de apreender o mais rápido e intuitivamente possível o que está sendo feito a cada instante, reduzindo o desgaste intelectual do usuário e aumentando sua produtividade.

Sabe-se que o uso de imagens e de formas visuais de apresentação de dados pode ser muito mais eficiente para a comunicação do que outras formas de transmissão tais como a palavra escrita, por exemplo. A compreensão de imagens é geralmente muito mais rápida, apelando para a intuição do usuário, não exigindo tanto de sua concentração e atenção. Isso quer dizer que o esforço de compreender uma imagem é muitas vezes menor do que o esforço de tentar reproduzi-la mentalmente através de outras formas descricionais.

Em síntese, seria muito útil encontrar uma forma de prover a *visualização* do que está sendo feito ou trabalhado, quando então o usuário seria capaz de perceber

imediatamente quaisquer novas condições de operação do sistema. Esta proposta afasta o projeto de forma definitiva de outras implementações que, desde os anos 50, permeiam a comunidade científica. Tais implementações são ainda hoje chamadas de **caixas pretas**, dificilmente podendo haver nome mais expressivo e adequado. Completamente fechadas ao usuário, elas recebem uma descrição do problema e depois de certo tempo devolvem respostas sem que o usuário tenha tido oportunidade de tomar qualquer atitude ativa no processo de determinação de soluções. Ainda hoje largamente utilizadas, estas caixas pretas não correspondem às expectativas quanto a comunicabilidade e, portanto, qualquer forma de implementação que aproxime o sistema deste tipo de comportamento deve ser evitada[1].

Assim sendo, busca-se uma abordagem diferente das que se apresentam usualmente, que são sistemas baseados em caixas pretas ou que apresentam uma interface inexpressiva no tocante à comunicação efetiva (formas textuais exaustivas de diálogo). Uma análise das necessidades que poderiam facilitar o processo de diálogo acima proposto indica entre outros itens:

- Capacidade gráfica, para prover formas de visualização de resultados. Evidentemente, as formas de visualização são totalmente dependentes do tipo de problema a ser resolvido, variando de acordo com as diferentes necessidades e dependem evidentemente da máquina escolhida para suportar a implementação, que deve fornecer as capacidades gráficas suficientes para sua realização. A representação gráfica do problema será desenvolvida na seção 3.3.

- Evitar o uso, comum em ambientes de pesquisa, da apresentação de resultados por meio de extensas tabelas de valores, contendo dezenas ou mesmo centenas de números que exigem do usuário uma grande capacidade de abstração e

[1] A favor das caixas pretas no entanto, deve-se dizer que são bastante portáveis, já que não fazem qualquer tipo de saída ou diálogo com o usuário, comunicando-se somente por meio de áreas de memória. Ademais, têm sido usadas e testadas durante tanto tempo que estão extensivamente validadas e documentadas, existindo várias bibliotecas de rotinas deste tipo atualmente.

concentração para compreender totalmente as informações fornecidas por estes dados.

- Fornecer meios para que um usuário possa expressar o que deseja de forma clara e simples, a ser compreendida pela máquina. Portanto, fornecer capacidades de expressão que sejam intuitivas ao usuário e que possam ser usadas sem maior esforço.

A lista acima pode ser sumarizada em um princípio básico: procurar atingir a intuição do público usuário, fornecendo-lhe as mesmas informações sob uma forma de apresentação que seja facilmente assimilada, dispensando esforço e concentração demasiadas e desnecessárias. Além de tais quesitos, podem ser incluídos outros que tornarão a operação do sistema bem mais proveitosa:

- Grande capacidade de operação sobre dados numéricos, ou seja, alta velocidade quando operando sobre valores em ponto flutuante. Essencial para a determinação de raízes, esta capacidade torna as avaliações do polinômio muito mais rápidas, poupando tempo ao usuário. Eventualmente ainda, pode-se vir a precisar de grande capacidade de cálculo em alguma outra operação do sistema.

- Entrada/saída por meio de arquivos de dados, pois o usuário tende a ficar desestimulado pela digitação continuada de informações repetidas a cada sessão de trabalho.

Para a entrada de dados durante a execução do programa deve-se preferencialmente optar pelo uso do teclado, evitando ou restringindo o uso do mouse pois este dispositivo não possui a multiplicidade de significados que poderiam ser atribuídos às teclas ou a comandos digitados. Adicionalmente, em caso de uma futura versão que deva ser executada em um ambiente mais limitado pode-se vir a ter de ignorar o mouse agora inexistente, e usar o teclado, alterando toda a interface anteriormente utilizada.

3.3 Abordagem gráfica do problema

De acordo com a idéia de usar as capacidades gráficas para exprimir com mais clareza uma maior quantidade de informação, o problema deve ser analisado para ajustar-se a esta nova filosofia, sendo desenvolvida uma abordagem que permita sua representação gráfica. Para tanto, deve-se tentar encontrar uma forma de apresentar um polinômio por meio de imagens, e devem ser conhecidas as particularidades e facilidades vindas desta forma de representação. Isto será feito a seguir, dentro dos próximos tópicos.

3.3.1 Transformação do problema

A representação de um polinômio através de uma imagem pode ser feita facilmente para o caso de polinômios com coeficientes reais e variável também real. Para este caso específico, uma representação no plano é suficiente. No problema proposto, entretanto, existe uma situação diferente a ser analisada: o caso de um polinômio de coeficientes complexos e variável também complexa. A representação de imagens deste tipo exigiria quatro dimensões, e certamente confundiria o usuário mais do que o auxiliaria, dada a multiplicidade de vistas e cortes a ser apresentados.

Inicialmente deve ser encontrada uma forma equivalente de representar o polinômio facilitando sua compreensão ou mesmo diminuindo o número de dimensões necessárias à realização das imagens. Torna-se necessário portanto definir uma forma de construir e apresentar imagens a ser usada no decorrer das sessões de trabalho. Em primeiro lugar, a representação complexa de $p(z)$ exige quatro dimensões. Duas destas são usadas para a variável independente z e duas para $p(z)$. De acordo com o objetivo final do sistema, são de especial interesse os zeros de $p(z)$, ou seja, os pontos nos quais $p(z)$ tem módulo 0. Por esta linha de raciocínio, pode-se pensar na alternativa de representar os módulos de $p(z)$ sobre o plano complexo. Sabendo que o módulo de um valor complexo é sempre um número real não negativo, pode-se trocar as duas dimensões que seriam usadas para representar $p(z)$ por uma só, representando $|p(z)|$. De quatro dimensões

originais restam agora três, que podem ser representadas em uma tela por meio de uma única imagem.

Uma superfície deste tipo, função de duas variáveis, pode ser facilmente apresentada através de uma imagem em três dimensões, tarefa implementada sem quaisquer problemas em um computador. Neste momento deve-se discutir a validade desta representação, já que evidentemente as imagens planejadas são tridimensionais pela natureza de seu método de construção, mas existe a possibilidade de efetuar sua apresentação de forma mais simples, tanto algoritmicamente quanto em termos de desempenho. Além disso, uma superfície em três dimensões em geral impede a completa visão de si mesma, encobrindo partes de si própria, além de exigir um série de parâmetros de controle. Isto não se coaduna com os objetivos do sistema, pois o usuário não deve ter de se preocupar com questões menores tais como posições alternativas de observador e outras. Ademais, o custo das projeções usadas para construir a imagem pode vir a ser bastante alto. Portanto, a produção de várias cenas tridimensionais pode ser custosa[2] e deveria ser evitada se possível. Resta a alternativa de gerar imagens que *representem* três dimensões usando apenas as duas existentes na tela e evitando o mapeamento de três para duas dimensões. Isto pode ser feito da seguinte forma: precisa-se de uma maneira de representar um vetor $v = (Re(z), Im(z), |p(z)|)$. Ora, as duas primeiras componentes podem ser facilmente mapeadas sobre os eixos x-y existentes na tela, gerando um pixel $p = (Map(Re(z)), Map(Im(z)))$. O valor de $|p(z)|$ deve ser portanto associado a este pixel de algum modo.

Resta a dúvida de como fazer a apresentação deste valor associado. Usando os recursos que se oferecem, surge a idéia de usar a cor, já que esta facilmente pode ser vinculada a cada pixel. Assim, o mapeamento inicial das três dimensões passa a ser feito apenas com duas, usando-se a cor de cada pixel para representar a terceira dimensão e reduzindo o esforço de mapeamento. Para tanto, precisa-se apenas de uma função de

[2]Deve ser lembrado que os custos de projeção pespectiva e de mapeamento sobre a tela tornariam ainda mais dispendioso o processo de visualização, que já deve ser caro em vista das freqüentes avaliações do polinômio, e adicionalmente exigiriam do usuário uma preocupação com problemas como escolha de posições de observador, ângulos de visualização, abertura de lentes, etc. Evidentemente, tais preocupações devem ser evitadas ao usuário, possibilitando sua atenção completa no problema a ser resolvido.

mapeamento que transforme um valor de $|p(z)|$ em uma cor. Tal transformação será discutida a seguir.

3.3.2 Interpretação visual

Feita a opção por apresentar em duas dimensões o polinômio, deve-se explicar melhor como deve ser esta representação. Como anteriormente explicado, as partes real e imaginária da variável independente z servirão para mapear z para um pixel sobre a tela. No pixel mapeado, uma cor deve ser calculada levando em conta $|p(z)|$. Dependendo da função de mapeamento de cor, pode-se obter diversas imagens diferentes. Evidentemente existem funções mais ou menos adequadas à tarefa de mapeamento, que geram ou não imagens satisfatórias tanto do ponto de vista estético quanto do ponto de vista de sua eficiência na comunicação das informações. Pretende-se escolher uma função de mapeamento que obedeça às seguintes características:

- A função deve ter um domínio positivo, pois os valores de $|p(z)|$ são sempre positivos ($0 \leq |p(z)| < +\infty$).
- A função de mapeamento de cor deve ter como contradomínio o conjunto de inteiros {0,1,2,. . . ,cor_disp-1}. Isto faz com que os resultados do mapeamento de cores sejam dados diretamente como entradas da tabela de cores existente na máquina em uso, evitando processamento posterior. O valor de cor_disp representa o número de cores disponíveis no equipamento usado.
- A função deve ser monotônica, evitando que diferentes valores de $|p(z)|$ conduzam à mesma representação, possibilitando a interpretação errônea dos dados apresentados.
- A função deve possuir algum tipo de parâmetro que permita o controle, por parte do usuário, da *escala* do mapeamento. Evidentemente é impossível fazer um mapeamento *linear* dos valores em $(0 \ldots \infty)$ para as cores disponíveis.

Tabela 3.1: Forma de distribuição das cores em função de w.

$\|p(z)\|$	Cor a ser mapeada
$0 \ldots w$	cor_disp-1 ... (cor_disp-1)/2+1
w	(cor_disp-1)/2
$w \ldots \infty$	(cor_disp-1)/2-1 ... 0

Uma forma de contornar o problema é permitir que o usuário possa controlar o *centro* da escala de cores sendo usado. Este valor do centro permite que se distribuam os valores das cores para mais perto ou mais longe de 0, mas sempre cobrindo o intervalo $(0 \ldots \infty)$. Esta proposta pode ser melhor explicada pela tabela 3.1.

Uma possível função de mapeamento pertinente com as afirmações acima pode ser definida pela expressão a seguir :

$$cor(w, z) = \lfloor cor\_disp - \frac{cor\_disp}{1 + \frac{w}{\sqrt{|p(z)|+\delta}}} \rfloor$$

onde

- cor_disp é o número de cores disponíveis no equipamento;
- w é o valor do centro da escala de cores (ou centro de cores) usado na imagem;
- δ é um valor pequeno, da ordem de 10^{-20}, com a função de evitar uma divisão por zero;
- z é a variável independente.

Pode-se verificar facilmente que o comportamento desta função segue as regras:

1. $cor(x, z) \leq cor(y, z) \Rightarrow x \leq y$;

2. $cor(w, z_1) \leq cor(w, z_2) \Rightarrow |p(z_1)| \geq |p(z_2)|$;

3. $cor(0, z) = 0$;

4. $cor(\infty, z) = cor\_disp$;

5. $cor(w, z) = (cor\_disp - 1)/2$ se $\sqrt{|p(z)|} = w$;

3.4 Exatidão visual ou estatística

A representação visual das avaliações de $p(z)$ em uma dada região pode ser uma maneira bastante interessante de representar informações sobre o comportamento *geral* do polinômio. Usualmente as imagens são compreensíveis e explícitas o bastante para que se percebam com clareza detalhes comportamentais. Existem ainda situações nas quais as imagens permitem que sejam obtidas algumas conclusões que, se obtidas numericamente, seriam incertas ou sujeitas a erro. O caso mais comum encontrado é o de polinômios com raízes múltiplas, pois a existência de raízes de multiplicidade maior do que um faz com que a superfície composta por $|p(z)|$ possua um comportamento diferente na vizinhança da raiz. Este comportamento pode ser assim explicado, de acordo com cada caso:

- Se for feito um corte transversal na superfície formada por $|p(z)|$ perto de uma raiz de multiplicidade um, a borda do corte na superfície será similar a uma das curvas da figura 3.1. Como se pode perceber claramente, todas as curvas formadas pelas bordas são bastante fechadas, ou seja, de forma nenhuma pode-se pensar que uma delas tangencie o plano $z = 0$, onde está o seu ponto mais baixo, a raiz.

- Se o corte transversal for feito perto de raiz de **maior** multiplicidade, a borda da superfície será semelhante a uma das curvas na figura 3.2. Fica bastante

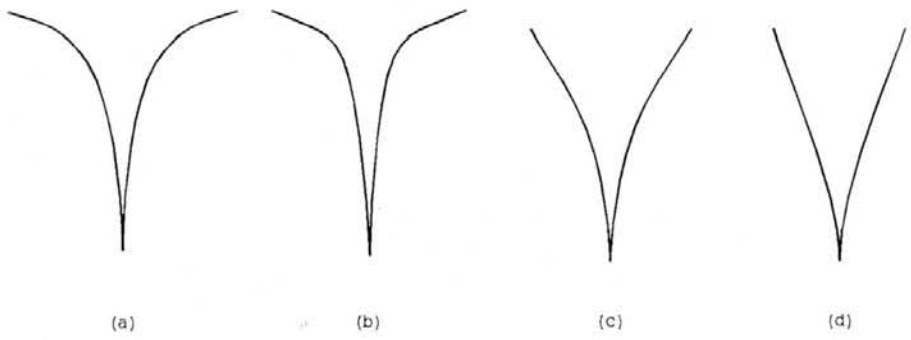

Figura 3.1: Bordas para raízes simples

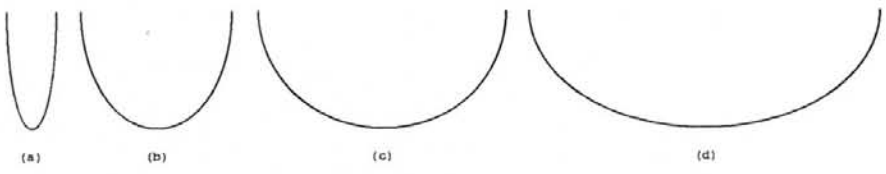

Figura 3.2: Bordas para raízes múltiplas

claro que as curvas deste tipo de raiz são bem mais abertas, dirigindo-se com suavidade até o ponto mínimo. Isto se deve ao fato de que a maior multiplicidade exige que pelo menos a derivada primeira se anule na raiz, fazendo com que a superfície tangencie o plano $z = 0$. A abertura das curvas depende da multiplicidade da raiz em questão, e quanto maior for esta multiplicidade, maior será a abertura.

Se o corte for feito perto de *grupos* de raízes simples, também existirão concavidades semelhantes às feitas por raízes múltiplas, pois a presença de várias raízes simples forma um rebaixamento da superfície de forma similar a uma única raiz múltipla. As concavidades formadas por grupos, entretanto, não são em geral simétricas em sua geometria, ao contrário das geradas por raízes múltiplas, podendo portanto ser diferenciadas entre si.

No caso de raízes múltiplas o surgimento destas concavidades na superfície é sinal de que o valor de $p(z)$ possui variações mínimas na área próxima à concavidade. Em termos práticos, se fossem feitas avaliações *exatas* de $p(z)$ na região próxima à raiz,

Tabela 3.2: Surgimento de erros e multiplicidade de raízes.

Mult.	Intervalo (diâm.)
1	0.0000000000007
2	0.000002
3	0.001
4	0.02
5	0.2
6	0.7
7	1.1
8	2

seriam obtidos resultados muito semelhantes, com pouca diferença entre valores de z diferentes. Entretanto, devido à representação finita que é usada, a semelhança entre os valores faz com que sejam multiplicados os erros contidos na avaliação. Este efeito pode ser mostrado com clareza quando o usuário faz ampliações sobre uma raiz múltipla. Os erros da avaliação são transmitidos às cores, e a imagem passa a ser um pontilhado de diferentes cores, não respeitando qualquer critério de distribuição que seja semelhante às imagens tradicionais.

Em testes realizados com vários tipos de raízes, foram coletados dados referentes ao valor do intervalo sendo representado em relação ao surgimento dos erros verificáveis na imagem. Assim, a tabela 3.2 apresenta o valor de algumas multiplicidades, bem como o tamanho dos intervalos sendo representados quando foi detectada a presença de erros na imagem[3].

As perturbações na representação dos resultados não são uma característica do sistema por si mesmo, mas da implementação da aritmética de ponto flutuante utilizada. As mesmas oscilações seriam obtidas por outros programas fazendo uso da mesma implementação. Entretanto, através da imagem obtida o usuário pode perceber a presença da instabilidade, sendo capaz de tomar a atitude que julgar adequada. Assim, pode-se des-

[3]Dada a existência de dois intervalos (real e imaginário), o valor do diâmetro apresentado na tabela é o valor do maior dentre estes dois diâmetros.

cartar eventuais resultados fornecidos pelo sistema quando este tenta determinar uma raiz em uma zona de instabilidade, ao contrário de outros sistemas onde o resultado fornecido não pode ser verificado diretamente.

3.5 Operações desejáveis

Sob o ponto de vista da interface gráfica, pode-se agora planejar algumas das operações que estarão presentes no sistema, mais especificamente aquelas que agirão sobre as imagens e os dados que as definem. Elas devem ser fornecidas ao usuário, de uma forma ainda a ser escolhida. Aqui estão apresentadas algumas delas, que podem ser consideradas no momento da implementação:

1. Alterações nos extremos reais da imagem, ou seja, alterações na parte real do domínio que está sendo apresentado. Deve ser possível alterar apenas um dos extremos (superior ou inferior) ou ainda ambos.
2. De forma similar, alterações sobre os extremos imaginários.
3. Além da representação das cores nos pixels, deve ser possível a representação através de curvas de nível. Neste caso, o usuário deve poder trocar o tipo de representação e além disso controlar o número de curvas de nível a ser apresentado.
4. Deve ser também possível a representação por curvas de sinal.
5. Uma operação de zoom, selecionando-se uma área da tela para substituir o domínio atual por um menor contido neste.
6. Uma operação para retornar de um nível de zoom para o nível anterior.
7. Uma operação que permita ao usuário consultar os valores de $p(z)$, seja fornecendo o valor de z ou ainda caminhando com um cursor sobre a superfície enquanto os valores de $p(z)$ na posição do cursor são apresentados.

8. Uma maneira de fazer a troca do centro da escala de cores, à vontade do usuário ou de forma predefinida, possivelmente ambas.

9. Uma maneira de alterar o conjunto de cores em uso, trocando-as por novas cores que agradem mais ao usuário.

10. Operação que coloque e retire o desenho dos eixos cartesianos Real e Imaginário.

11. Operações que alterem parâmetros gráficos do sistema. Estes parâmetros ainda não foram definidos neste ponto, mas serão explicados no capítulo sobre desempenho.

12. Uma operação que marque sobre a imagem as raízes já encontradas até o momento.

Estas são somente algumas das operações implementadas, aquelas que tem por função operar sobre a parte gráfica do sistema, e que de alguma forma alteram as imagens produzidas. Outras operações com diferentes finalidades serão apresentadas dentro da descrição dos comandos existentes na ferramenta.

O uso de periféricos como o mouse é desejável, mas não deve ser considerado essencial na construção do sistema, pois eventualmente pode ser preciso fazer uma troca de ambiente, quando então a parte da implementação que gerencia o uso do mouse sofrerá alterações profundas, e logo existirão dois ambientes com interfaces bastante diferentes, uma usando mouse e outra sem este periférico, criando uma certa confusão para o usuário que precisará conhecer dois ambientes distintos.

3.6 Interface: entrada

Tendo sido definidas as operações que estarão na interface a ser usada, resta desenvolver uma forma de especificar as operações a serem feitas, ou seja, prover o usuário

de uma forma de expressar sua vontade. Esta forma deve ser simples mas poderosa o bastante para tornar o sistema uma ferramenta realmente útil no trabalho do dia a dia. Ainda mais, a forma de entrada deve ser facilmente adaptável a outras plataformas de software e hardware, pois o usuário tem o direito de esperar que uma versão futura venha a ser tornada executável em outro ambiente que não o original.

Para ir ao encontro destes requisitos, deve-se descartar ou pelo menos restringir o uso de janelas e recursos muito sofisticados ou dependentes de uma máquina, sistema operacional ou ainda vinculados a um ambiente de janelas, já que tais dependências fazem com que não seja fácil ou mesmo possível transportar o sistema final. A opção parece ser uma forma (qualquer) de linguagem para expressar os comandos, pois esta poderia ser facilmente transportada entre máquinas diferentes com alterações mínimas. Evita-se portanto o uso excessivo de janelas, mouse, etc. Basicamente, deve ser provida uma linha de comandos, similar a outras implementações de sistemas voltados à resolução de problemas matemáticos (por exemplo, veja [WOL91]).

Embora exista uma aparente contradição à afirmativa anterior de que a ferramenta final não deve se aproximar do comportamento de caixas pretas ou sistemas de interface textual, deve ser dito que sistemas como estes, quando em operação normal, baseiam-se *unicamente* em interfaces textuais ou passagem de dados via memória. Ou seja, não fazem uso, como é intenção no sistema sendo agora planejado, de imagens durante *todo* o tempo. Alguns sistemas já existentes fornecem a capacidade de se trabalhar com gráficos, mas de forma eventual pois os problemas que se destinam a resolver pertencem a um espectro muito amplo, e nem todos podem ser representados graficamente. Como o problema proposto já foi trabalhado para obter-se uma forma de apresentação visual, pode ser admitido que esta será usada intensamente.

Adicionalmente, o uso de uma forma de expressão por comandos permite que a interface do sistema permaneça praticamente a mesma, apesar da adição de novos comandos e sinônimos para os já existentes, pois a interface não guarda qualquer relação com os comandos, a não ser por prover uma pequena área na qual eles são digitados. Também o usuário não precisa conhecer as diversas funções de uma série de janelas e

cardápios que lhe seriam apresentados com o fim de substituir a linguagem completa, conhecendo apenas os comandos que lhe são mais usuais. Isto significa por exemplo que, se forem fornecidos 30 comandos, possivelmente o usuário não terá necessidade destes trinta. Precisará, talvez, de 15. Ora, estes comandos ele conhecerá bem, e os utilizará como se fossem todos os comandos do sistema, descartando mentalmente outros que não lhe são úteis. Os comandos descartados continuam existentes e podem ser usados se necessário, mas não aparecem no trabalho cotidiano do usuário, e em geral não existem para ele. Ou seja, o usuário trabalha com os comandos que deseja, sem se preocupar em guardar a função de outros que lhe são desnecessários no momento.

A linguagem de comandos devem ser construída de tal forma que permita maneiras simples de expressão, isto é, comandos de memorização fácil com eventuais parâmetros simples de lembrar e que abranjam todas as operações habilitadas no sistema, quais sejam, operações referentes a leitura e escrita de arquivos de dados, operações sobre as imagens geradas, operações sobre o polinômio, etc.

A interface, portanto, tem um papel importantíssimo na apresentação dos dados, mas seu papel na entrada dos comandos por parte do usuário é bem mais restrito, devido às considerações sobre portabilidade e facilidade de alteração feitas acima.

3.7 Forma de implementação

Uma vez que estão definidas as principais características da ferramenta a ser construída, já é possível começar a trabalhar na sua forma de implementação. Inicialmente, o software deve ser desenvolvido em uma plataforma SUN, com os cuidados necessários a fim de possibilitar uma migração fácil para outro ambiente. Será feito uso de um ambiente de janelas que fornece facilidades de interface gráficas, e que proverá as primitivas sobre as quais o sistema será construído. O sistema deve possuir uma área gráfica extensa, fazendo uso de uma área auxiliar para entrada de comandos e apresentação de alguns resultados e mensagens.

Entre outras, as informações que devem estar permanentemente presentes na tela são:

- O nome do arquivo que contém o polinômio sendo processado;
- Informações sobre os limites reais da imagem;
- Informações sobre os limites imaginários da imagem;
- O número atualmente sendo usado como ϵ no critério de parada;
- O tamanho atual de cada célula (este item será melhor explicado no capítulo sobre desempenho);
- O centro da escala de cores em uso;
- O número de curvas de nível correntemente em uso;
- O nível de zoom atual;
- Uma área para eventuais mensagens ao usuário;

Como a operação será feita sobre uma plataforma que dispõe de mouse, ele será usado para as operações de zoom e consulta de valores, da seguinte forma:

- Para o zoom, o usuário pode dirigir o mouse diretamente para a imagem, marcando a área que deseja ampliar.
- Na consulta de valores, à medida que o cursor se move sobre a imagem, apresenta-se o valor de z, $p(z)$ e $|p(z)|$ na posição do cursor.

Pode ser notado que já existe um conflito no uso do mouse, pois existem dois usos para ele, sem maneira do usuário conhecer qual deles está em operação. Para resolver este empecilho, precisa-se inserir na interface uma seleção de funcionamento do mouse, que pode chavear entre as operações de zoom e de consulta a valores. Uma descrição mais aprofundada destas duas operações bem como de todas as outras operações instaladas será feita no próximo capítulo.

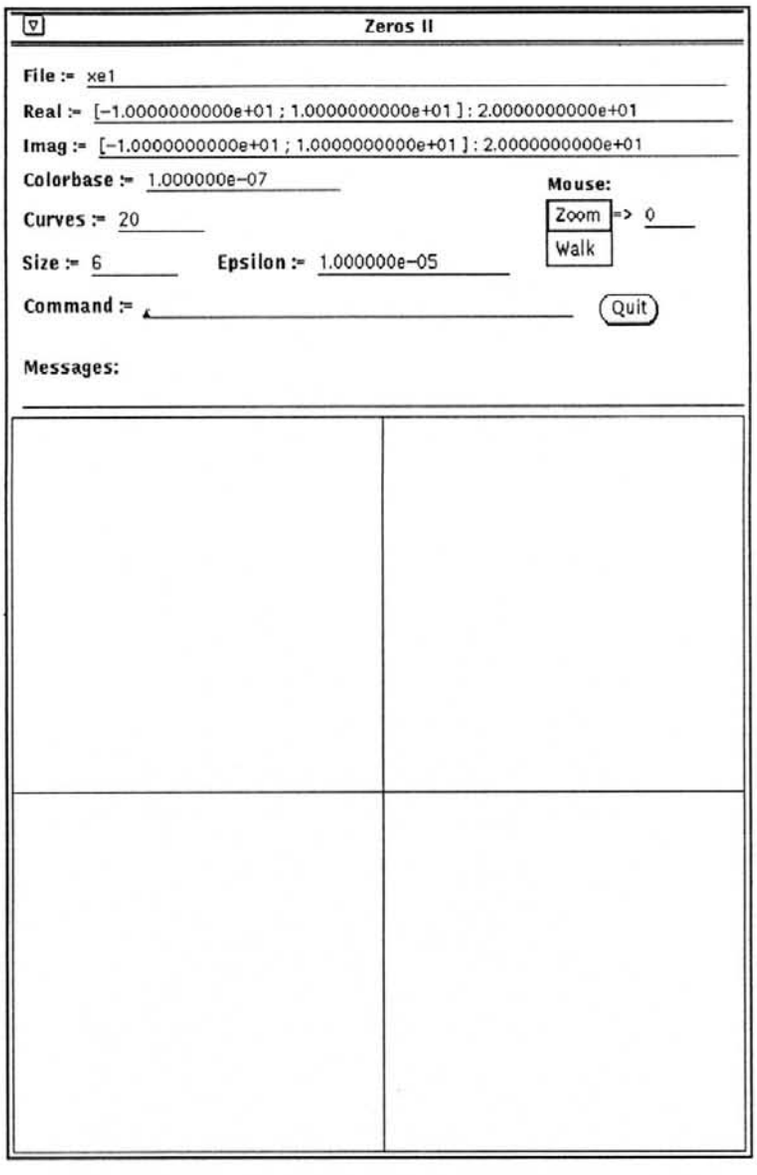

Figura 3.3: Interface proposta.

3.8 Apresentação de dados

A partir dos itens escolhidos para serem apresentados ao usuário, pode-se introduzir a interface apresentada na Fig. 3.3. Basicamente, ela compõe-se de uma grande área gráfica e de campos de texto que apresentam as informações mais úteis sobre a imagem. Nesta primeira proposta estão colocados os seguintes campos:

File

Este campo mostra o nome do arquivo sendo processado no momento.

Real

Este campo apresenta o intervalo real sendo mostrado, ou seja, a porção do eixo das abscissas que pode ser vista. Além disso, mostra também o valor do diâmetro deste intervalo, para que o usuário possa ter mais facilmente uma noção da faixa de valores na qual está trabalhando.

Imag

Mostra os extremos e o diâmetro do intervalo imaginário sendo exibido.

Colorbase

Este campo exibe o valor do centro de cores em uso atualmente.

Curves

Apresenta o número de curvas de nível sendo usado no traçado atual.

Size

Este campo exibe o valor do tamanho de célula sendo usado no momento.

Epsilon

Mostra o valor de ϵ atual.

Mouse

Os dois botões fazem a seleção entre as possíveis funções do mouse. Ao lado do botão Zoom, um valor indica o nível de zoom atualmente usado.

Command

Aqui, o usuário pode digitar os comandos para o sistema.

Quit

Este botão faz com que a execução do sistema seja terminada, e o usuário retorna ao ambiente de janelas.

Messages

Eventuais mensagens do sistema são exibidas nesta área.

Deve ser salientado que o único campo que pode ser diretamente editado pelo usuário é o campo **Command**, pois todos os outros não podem ser alterados. Assim implementado, o sistema somente permite alterações por meio de comandos que afetem o valor apresentado nos campos, e não o conteúdo dos campos. O propósito desta restrição é possibilitar uma portabilidade maior para uma plataforma que não forneça um ambiente de janelas, nem acesso e gerência dos itens apresentados nos campos de texto.

4 LINGUAGEM DE COMANDOS

A definição dos comandos a serem implementados irá ser feita neste capítulo, descrevendo cada comando existente, seus parâmetros e funcionamento. Eles tentam abranger todos os tipos de operações que o usuário possa desejar, e estão abertos a modificações e mudanças, pois, como já dito, é bastante simples fazer alterações na linguagem de comandos já que esta não depende de mudanças trabalhosas na interface.

Existe uma justificativa a mais para a adoção da linguagem de comandos: pode-se facilmente usar sinônimos para os comandos. Por exemplo, define-se todos os comandos iniciais da linguagem em língua inglesa, pois o idioma português tende a ser um tanto prolixo, e o inglês possibilita palavras menores do que suas correspondentes em português. Entretanto, é possível adaptar com facilidade a linguagem para aceitar termos adicionais em português, já que esta é uma alteração muito simples em uma linguagem, embora impossível de ser feita sobre uma interface baseada em cardápios e botões, pois neste caso existiriam os dois idiomas misturados ou somente um deles. Exagerando um pouco, pode-se adicionar comandos correspondentes aos originais em inglês em outras duas ou três línguas, somente adicionando novas palavras-chave para as mesmas ações.

Os comandos apresentados abaixo devem ser digitados na linha de comando apresentada na interface. Após os nomes de cada comando, devem seguir-se seus parâmetros, se existirem. Imediatamente após ser pressionado Enter ao final da linha, o comando é analisado, e, se validado, sua execução é iniciada.

Na lista de comandos dada abaixo, existem comandos que possuem sinônimos, os quais estão listados de forma agrupada. Por exemplo, existem cinco comandos para efetuar a saída do sistema. Dentro da linguagem implementada para a ferramenta, estão definidos os seguintes comandos (agrupados por função):

Os cinco comandos a seguir finalizam o sistema.

end, exit, fim, quit, '.'

Terminam a execução do programa.

Os comandos seguintes operam sobre a parte gráfica do sistema, alterando características visuais ou modos de operação. De forma geral, eles alteram a forma de apresentação dos resultados.

curves, curvas

Faz com que passe a ser usada a representação por curvas de nível. A construção das imagens geradas por esta opção pode ser ligeiramente mais lenta do que o uso de células.

max_curv

Este comando altera o número de curvas de nível usadas na representação por curvas. Se for usado ainda durante a representação por células, automaticamente as curvas passarão a ser utilizadas. Exige um parâmetro inteiro.

cells

Faz com que passe a ser usada a representação por células.

sinal, signal

Este comando passa a usar a apresentação baseada em curvas de sinal, gerando imagens mais rapidamente e com a mesma representatividade e informação das imagens baseadas em cor.

size, cellsize

Altera o tamanho das células usadas, de acordo com a vontade do usuário, exigindo para isso um parâmetro inteiro. Se usado enquanto o sistema está operando em modo Curvas, passa automaticamente ao uso de células.

axes, axis

Faz com que sejam ligados os eixos Real e Imaginário.

noaxes, noaxis

Faz que sejam desligados os eixos Real e Imaginário.

draw

Faz com que a tela atual seja redesenhada, apagando eventuais marcas feitas através de outros comandos. Por exemplo, marcas feitas pelo comando **locate** (ver a seguir).

refresh

Faz com que o modo de desenho automático seja ligado ou desligado (ou seja, inverte o modo corrente). Caso o modo de desenho automático esteja ligado, quaisquer alterações no polinômio ou no domínio farão com que seja iniciado outro desenho da tela. No modo desligado, novas telas serão feitas apenas quando usado algum dos comandos de zoom ou quando for acionado o comando **draw**. Quando da inicialização do sistema, o modo está ligado.

colors, cor

Faz a leitura de um arquivo contendo as novas cores a serem usadas na imagem. Este arquivo se compõe, nas máquinas SUN, de 132 linhas de texto, cada uma contendo 3 valores hexadecimais entre 0 e 255 cada um, separados por um espaço em branco. Este arquivo é lido, e os valores são colocados como entradas RGB (Red, Green, Blue) na tabela de cores em uso. A imagem muda suas cores instantaneamente, embora o resto do ambiente não se modifique. O comando exige um parâmetro, que é usado como o nome do arquivo que contém as cores[1].

center

Este comando possibilita alterações sobre o centro da escala de cores atual, correspondendo ao valor w discutido no capítulo sobre a interface. Este comando altera a escala de cores exigindo um parâmetro em ponto flutuante. Este valor será usado da seguinte forma: suponha que existem 200 cores sendo utilizadas. O centro das cores (isto é, a cor 100) será usado para representar este valor dado. Quando o módulo do polinômio for menor do que o valor fornecido (indo até zero) serão usadas as cores entre 101 e 200. As cores entre 1 e 99 são usadas para valores decrescentes de infinito até o valor fornecido.

[1]Em princípio, não se deve esperar que um usuário leigo saiba descrever ou mesmo construir arquivos especificando cores. No entanto, dada a eventual necessidade de alterar as cores utilizadas (por não serem do agrado do usuário ou para a produção de slides, por exemplo) deve ser fornecida uma forma de determinação de novas cores.

up

Multiplica por dez o valor do centro de cores usado no sistema, facilitando a visualização de cenas complexas e evitando que se tenha de consultar o valor atual.

down

Divide dez vezes o valor de centro de cores usado no sistema, funcionando com as mesmas vantagens do comando anterior.

speed

Este comando exige um parâmetro inteiro, e altera o valor de `scanspeed` sendo usado na imagem atual. A imagem é redesenhada imediatamente. Este comando, bem como o comando **size** serão explicados no capítulo sobre desempenho.

cauchy, cc

Esta ação avalia a cota de Cauchy para as raízes do polinômio, obtendo um valor máximo para o módulo das raízes de $p(z)$. O valor calculado é usado, em seguida, para substituir os extremos atuais da imagem e esta é redesenhada. Assim, a imagem obtida por este comando contém, com certeza, todas as raízes do polinômio. Se, ao ser feita a imagem, elas não forem todas visíveis, provavelmente o valor do centro de cores deve ser ajustado.

Os comandos a seguir operam sobre os valores de parâmetros usados na avaliação de raízes que podem ser alterados pelo usuário.

max_newton

Permite ao usuário alterar o número máximo de iterações usado pelo método de Newton para a determinação de raízes. Exige um parâmetro inteiro.

max_laguerre

Permite ao usuário alterar o número máximo de iterações usado pelo método de Laguerre para o refinamento das raízes. Exige um parâmetro inteiro.

epsilon

Este comando serve para alterar o valor usado internamente pelo sistema como ϵno critério de parada da determinação de raízes. Exige como parâmetro um número real.

Esta categoria de comandos permite que seja alterado o próprio polinômio, ou seus valores sejam consultados ou ainda suas raízes determinadas.

change, altera

Este comando permite que se faça a alteração de um dos coeficientes do polinômio. Para usá-lo, deve ser fornecido um inteiro e um número complexo. O inteiro será usado como o índice do coeficiente a alterar, e o número complexo será o novo valor. Por exemplo:

change 5 3 3.55 $\rightarrow$ irá alterar o coeficiente α_5 para $3 + 3.55i$.

Obs. : o coeficiente do termo de menor grau do polinômio é denotado α_0.

guess, chute

Estes comandos usam como parâmetro um número complexo. Este número serve como ponto inicial para sucessivas iterações do método de Newton, e posteriormente será refinado pelo método de Laguerre. Assim, é fácil determinar qual a raiz que será avaliada a partir de um determinado valor inicial. A raiz encontrada é armazenada para uso e consulta posterior.

zeros

Este comando inicia o processo de determinação de todas as raízes do polinômio. Os valores são mostrados na console, e também podem ser consultados com o comando **answer** (ver abaixo).

value, val

Estes comandos usam como parâmetro de operação um valor complexo, efetuando em seguida a avaliação do polinômio para este valor fornecido e retornando o resultado ao usuário.

answer

Este comando mostra os valores das raízes já encontradas.

Os comandos a seguir operam sobre os extremos real e imaginário da imagem, possibilitando a alteração da área sendo pesquisada.

lo x, lo y, inf x, inf y

Estes comandos exigem um valor de ponto flutuante, e alteram o limite inferior de x ou y para este valor.

hi x, hi y, sup x, sup y

Estes comandos exigem um valor de ponto flutuante, e alteram o limite superior de x ou y para este valor.

x, re

Estes comandos exigem dois valores de ponto flutuante, e alteram os limites reais do domínio de acordo com estes valores.

y, im

Estes comandos exigem dois valores de ponto flutuante, e alteram os limites imaginários do domínio de acordo com estes valores.

As operações de entrada/saída são fornecidas pelos seguintes comandos:

read, load

Estes comandos exigem um nome de arquivo, que conterá um novo polinômio de trabalho. Este novo polinômio substituirá o antigo.

tofile

Refaz a determinação de todas as raízes do polinômio. Os valores não são mostrados, mas escritos em um arquivo cujo nome é exigido como parâmetro neste comando.

write

Este comando exige um nome de arquivo, e salva neste arquivo o polinômio e as raízes já encontradas.

Estes são comandos auxiliares, providos pelo sistema. Eles tem por função prover algumas facilidades que foram consideradas interessantes para o usuário.

who, quem

Este comando mostra qual o polinômio que está sendo utilizado no momento. A saída do comando é feita na janela na qual o sistema foi invocado.

deriv

Este comando exige um número inteiro, o qual designa a derivada a ser mapeada sobre a tela da mesma forma que o polinômio original. Apenas a derivada é desenhada, e não passa-se a operar sobre ela. Assim, se logo depois realizarem-se outras operações, volta-se a trabalhar sobre o polinômio original.

locate

Este comando apresenta sobre a tela a localização das raízes já encontradas, marcando-as sobre o mapa do polinômio.

back

Este comando faz com que se retorne ao nível de zoom anterior.

'-'

Este comando retorna n níveis de zoom de uma única vez. Portanto, este comando exige n, um parâmetro inteiro. O número de níveis de zoom pode ser conhecido pois está entre os dados apresentados na interface. Se o usuário tentar recuar mais níveis do que os atualmente usados, o sistema retorna ao primeiro nível. Deve haver um espaço em branco entre o caracter '-' e o valor inteiro.

way

Tal comando exige dois parâmetros inteiros e dois reais, usados na seguinte ordem: o

primeiro inteiro é usado como o número de interpolações a serem geradas variando-se o coeficiente indicado pelo segundo número inteiro. A variação deste dá-se do primeiro número real até o segundo. As raízes obtidas a cada interpolação são plotadas, obtendo-se a seqüência dos movimentos das raízes. Somente a parte *real* do coeficiente em questão será alterada. A parte complexa continua a mesma durante toda a seqüência de movimentos.

Na interface são refletidas quaisquer alterações sobre os valores nela apresentados. Ou seja, se algum dos comandos efetuar mudanças sobre valores de itens apresentados na tela, o item alterado terá seu valor atualizado, e se necessário, a tela será redesenhada.

Além das ações providas acima, existem na implementação sobre as plataformas SUN duas outras operações: Zoom e Walk. Estas operações fazem uso do mouse e serão descritas a seguir com mais detalhes.

As duas ações permitem que o usuário caminhe com o mouse sobre a imagem. Portanto, existe um conflito referente à escolha da ação a ser realizada. Para solucionar a questão, optou-se por usar dois botões na interface. Estes botões são mutuamente exclusivos, ou seja, somente um deles pode estar pressionado a um dado tempo, e sempre há um deles pressionado. Os dois fazem a seleção entre Zoom e Walk, podendo ser pressionados com a seta do mouse, a qualquer momento. A partir daí, o usuário pode entrar com o mouse na área da imagem e efetuar a operação que desejar.

Zoom

O zoom pode ser feito entrando-se na área de imagem e movendo-se o cursor sobre ela. A seleção da área a ser ampliada é feita por meio de um retângulo, assim marcado: para escolher o primeiro dos cantos do retângulo, deve ser pressionado o botão esquerdo do mouse. A partir deste momento, à medida que o cursor se move passa a ser desenhada uma caixa correspondente à área a ser ampliada. Uma vez escolhida a área com o auxílio da caixa, pressiona-se outra vez o botão esquerdo e o zoom será feito imediatamente. Os parâmetros gráficos da nova imagem continuam os mesmos da imagem anterior, ou seja, tamanhos de célula, centro de cores e outros parâmetros não são alterados. Evidentemente,

alteram-se os valores dos extremos reais e imaginários sendo apresentados. Após ser marcado o primeiro dos cantos da caixa, a operação pode ser abortada pressionando-se o botão direito do mouse.

Adicionalmente, existe uma função associada ao botão central do mouse: se pressionado para marcar o segundo canto da área a ser ampliada, o efeito do zoom será invertido, ou seja, um zoom inverso. Isso significa que a tela atualmente representada será remapeada para dentro da caixa selecionada pelo mouse, e portanto encolhida. Pode-se por conseqüência fazer uma ampliação através do mouse, escolhendo o local para onde a tela atual deve ser transportada.

Walk

A operação de walk inicia-se imediatamente após a entrada do cursor na imagem. A partir deste momento, os valores de z, $p(z)$ e $|p(z)|$ são mostrados na área de mensagens a cada ponto por onde passa o cursor, possibilitando a pesquisa de valores. Adicionalmente, se o botão esquerdo do mouse for pressionado, o valor corrente de z será usado como valor inicial para as iterações de Newton e Laguerre, convergindo para alguma das raízes de $p(z)$. A raiz encontrada será assinalada por intermédio de um pequeno quadrado que piscará por alguns segundos.

5 USO E EXEMPLOS

Para fazer com que a utilização do sistema seja melhor compreendida e seus mecanismos de funcionamento efetivamente utilizados, este capítulo deve fornecer exemplos de uso do sistema, em especial de operações que possuem alguma dificuldade para o usuário iniciante. À medida que estes exemplos forem desenvolvidos, serão exploradas de maneira gradual algumas das operações habilitadas pelo sistema.

5.1 Formatos de entrada

Já que o sistema tenta reduzir o trabalho do usuário, inclui-se aí a possibilidade de trabalhar com arquivos de dados, diminuindo a digitação e a repetição de tarefas desinteressantes a cada vez que se deseja usar o sistema. Para possibilitar a implantação desta idéia de forma mais efetiva, existe um formato para os arquivos de entrada que são usados para definição dos polinômios. Este formato faz com que o arquivo seja composto de uma lista de itens usados para a descrição dos coeficientes.

Ao contrário do que poderia ser esperado, no arquivo não existe uma descrição da primeira região na qual deve ser construída a imagem. Poupando trabalho ao usuário, prefere-se usar a técnica de calcular as cotas de Cauchy, quando então o sistema pode dirigir a imagem imediatamente para uma região que contenha todas as raízes do polinômio. Portanto, a primeira imagem do sistema sempre representará a área determinada pela cota de Cauchy, podendo ser alterada posteriormente de acordo com a vontade do usuário. Adicionalmente, não são fornecidos valores para o tamanho de células a ser usado, nem para o centro de cores inicial ou para o valor de ϵ ao início do programa. Para estes e outros valores são assumidos os seguintes valores iniciais[1]:

[1] estes valores se referem à versão atual do sistema, podendo ser alterados a qualquer instante para que se trabalhe com outros valores mais adequados à tarefa em questão.

Tabela 5.1: Valores iniciais de parâmetros do sistema.

Parâmetro	Valor default
Grau máximo de polinômio	75
Tamanho de célula	2
Centro de cores inicial	1
ϵ (crit. de parada)	0.00001
`scanspeed`	1
Tipo de desenho	Curvas de nível
Num. de curvas de nível	20
Função do mouse	Zoom

O arquivo apresenta os valores dos coeficientes do polinômio, por meio de uma lista de itens onde cada um segue o formato seguinte:

```
( coef_real , coef_imag ) * z ^ p
```

No formato acima, *coef_real* e *coef_imag* são as partes real e imaginária do coeficiente α_p no polinômio, e são representados em valores de ponto flutuante. Podem ser usadas várias vezes referências ao mesmo coeficiente α_p, e a cada nova referência a valores anteriormente definidos para α_p os novos coeficientes fornecidos serão somados aos valores já existentes. Adicionalmente, os coeficientes podem ser fornecidos em qualquer ordem, não precisando ser respeitada a ordem crescente ou decrescente de seu grau, embora seja usual por motivos de organização visual. Para coeficientes ausentes, evidentemente, assume-se o valor zero. Para a geração dos exemplos deste capítulo, foi usada uma ferramenta descrita no Anexo A-1.

5.2 Exemplo de coeficientes

A fim de dar um exemplo de geração de um arquivo de coeficientes, imagine-se um polinômio com as seguintes raízes: $i, -i, 1, 2, 3, 3, 5$. Se forem desenvolvidos os coeficientes deste polinômio, obtém-se :

$$p(z) = z^7 - 14z^6 + 75z^5 - 198z^4 + 287z^3 - 274z^2 + 213z - 90$$

Este polinômio pode ser descrito pelo seguinte conjunto de especificações para cada um de seus coeficientes:

```
(   1 , 0) * z ^ 7
( -14 , 0) * z ^ 6
(  75 , 0) * z ^ 5
(-198 , 0) * z ^ 4
( 287 , 0) * z ^ 3
(-274 , 0) * z ^ 2
( 213 , 0) * z ^ 1
( -90 , 0) * z ^ 0
```

5.3 Exemplo 1

Para iniciar o uso do sistema, é uma boa escolha trabalhar sobre um polinômio conhecido, do qual já são sabidas as raízes. Elas serão: 0, 1, 2, 4, 8, 16. Portanto, um polinômio de sexto grau. A descrição do polinômio pode ser feita como no seguinte arquivo:

```
(    1 , 0) * z ^ 6
```

```
(  -31 , 0) * z ^ 5
(  310 , 0) * z ^ 4
(-1240 , 0) * z ^ 3
( 1984 , 0) * z ^ 2
(-1024 , 0) * z ^ 1
(    0 , 0) * z ^ 0
```

A ordem apresentada acima poderia ter sido diferente, mas por motivo de organização visual é típico manter os coeficientes ordenados, do grau mais alto até o menor. A versão executável do sistema proposto tem o nome de `zeros`. Assim, deve-se chamar `zeros` e o sistema pedirá o nome de um arquivo de descrição. Fornecendo-lhe o nome do arquivo acima, o sistema de janelas é inicializado. A primeira imagem não surge imediatamente, mas deve ser produzida usando-se o comando **draw**.

A imagem inicial está representada na Fig. 5.1. Como o valor do centro de cores tem a função adicional de distribuir as curvas de nível na imagem, isto significa que o valor do centro de cores altera também a concentração das curvas. Uma segunda imagem obtida alterando-se o centro de cores está mostrada na Fig. 5.2. Visivelmente, as curvas de nível estão menos concentradas agora. Uma das falhas mais comuns na geração de uma imagem é o valor inadequado do centro de cores, exemplificado neste caso. Infelizmente, não existem até agora formas automáticas de obter bons valores para o centro de cores, impedindo uma implementação que se ajuste automaticamente a novas imagens.

Nos valores apresentados como extremos do domínio da imagem, pode-se observar o valor da cota de Cauchy calculada. Assim, todas as raízes devem estar contidas na imagem. Isto se verifica facilmente pela imagem da figura.

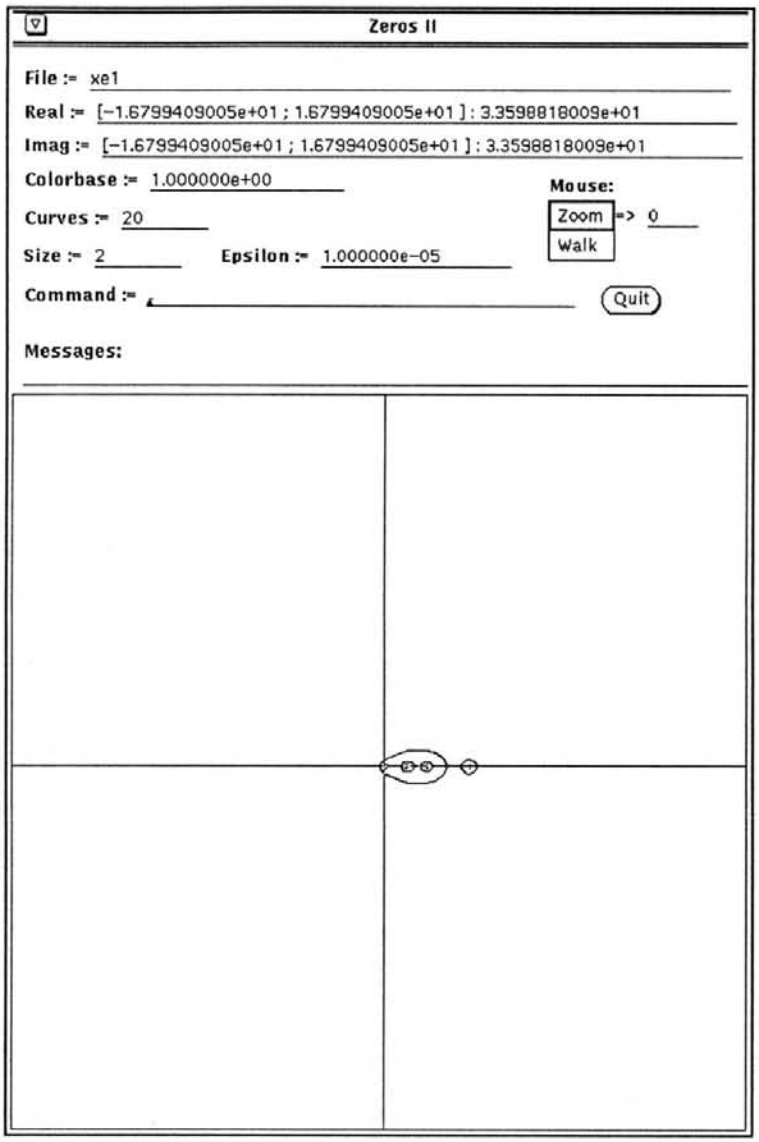

Figura 5.1: Imagem obtida para o exemplo 1.

5.4 Exemplo 2

Neste exemplo, definiremos um polinômio que tem por raízes os valores 0, 2, 4 e mais outras oito raízes imaginárias geradas aleatoriamente. As raízes imaginárias não serão conjugadas, ou seja, não possuirão relação entre si. A lista completa das raízes está fornecida na tabela 5.2:

Os coeficientes do polinômio obtido a partir destas raízes estão apresentados a seguir, e a primeira imagem fornecida para ele pelo sistema está apresentada na Fig 5.3.

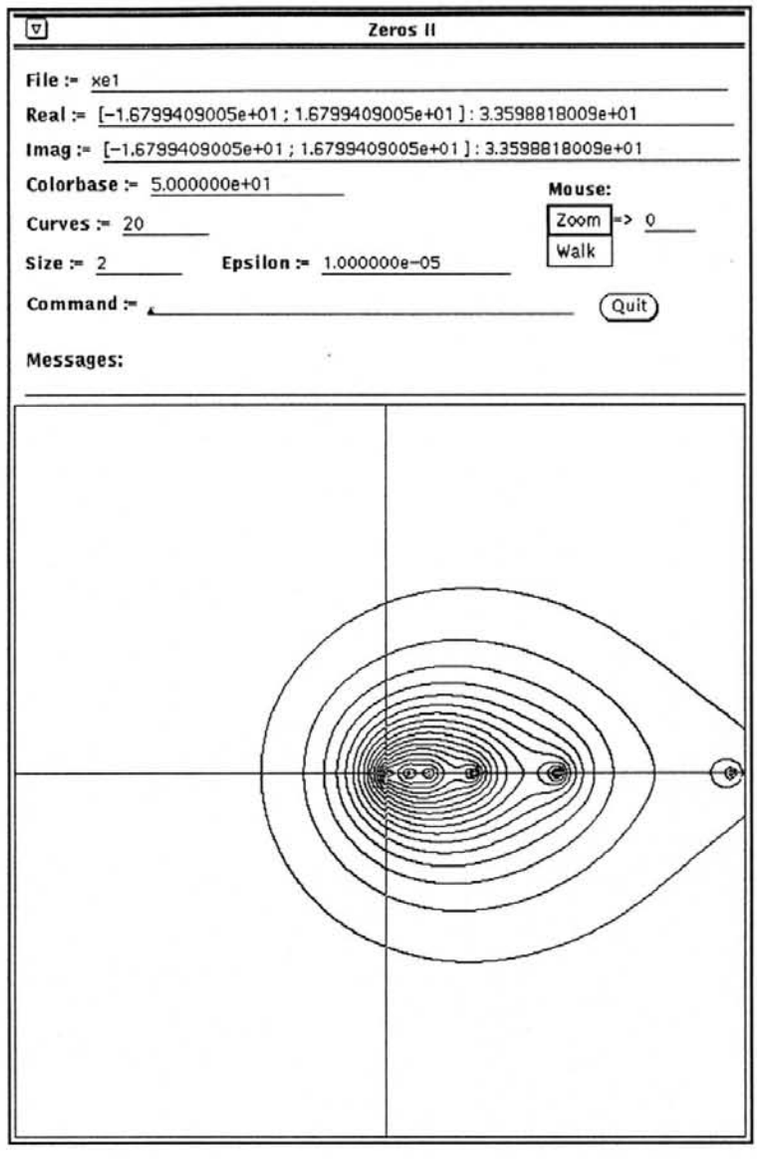

Figura 5.2: Imagem com redistribuição das curvas.

O surgimento de partes imaginárias nos coeficientes do polinômio se deve à presença de raízes não conjugadas.

```
(+1.000000000000e+00, +0.000000000000e+00) * z ^ 11
(-2.461659023288e+00, -1.811163174832e-01) * z ^ 10
(-1.633498477963e+01, -2.758683687222e+01) * z ^ 9
(+2.569511326918e+01, +9.629076467408e+01) * z ^ 8
(-1.950799768671e+02, +3.161827527630e+02) * z ^ 7
(+1.134701970186e+03, -9.029764047440e+02) * z ^ 6
```

Tabela 5.2: Raízes para o exemplo 2.

Parte real	Parte imaginária
+0.000000000000	+0.000000000000
+2.000000000000	+0.000000000000
+4.000000000000	+0.000000000000
+1.850246813079	+1.756946771292
+1.965332057777	+1.787299225008
+1.628246898124	+1.222487001318
+1.498257403028	+1.766072932988
-2.652317987593	-1.618281870903
-2.363856618462	-1.770430240673
-2.557781162931	-1.308687000679
-2.906468379733	-1.654290500867

```
(+1.429098387539e+03, +2.667472456870e+02) * z ^ 5
(-8.359278149772e+03, -4.024577286752e+03) * z ^ 4
(+2.533294195930e+03, +1.721365940919e+03) * z ^ 3
(+4.096820106229e+02, +2.227809851010e+04) * z ^ 2
(+1.035991669119e+04, -2.002370573033e+04) * z ^ 1
(+0.000000000000e+00, +0.000000000000e+00) * z ^ 0
```

5.5 Alterações nas imagens

A imagem da Fig. 5.3 não é muito satisfatória, pois as curvas estão muito concentradas sem cobrir boa parte do domínio. Isto é sinal de que o centro de cores está muito pequeno, concentrando as curvas em torno de valores muito baixos para a região em estudo. Desta forma, altera-se o valor do centro de cores, passando-o para 10, bem maior do que o anterior (de valor 1). A segunda imagem obtida é apresentada na Fig. 5.4. As curvas podem ainda ser consideradas aglomeradas, e portanto experimenta-se um valor de centro igual a 100. O resultado está na Fig. 5.5.

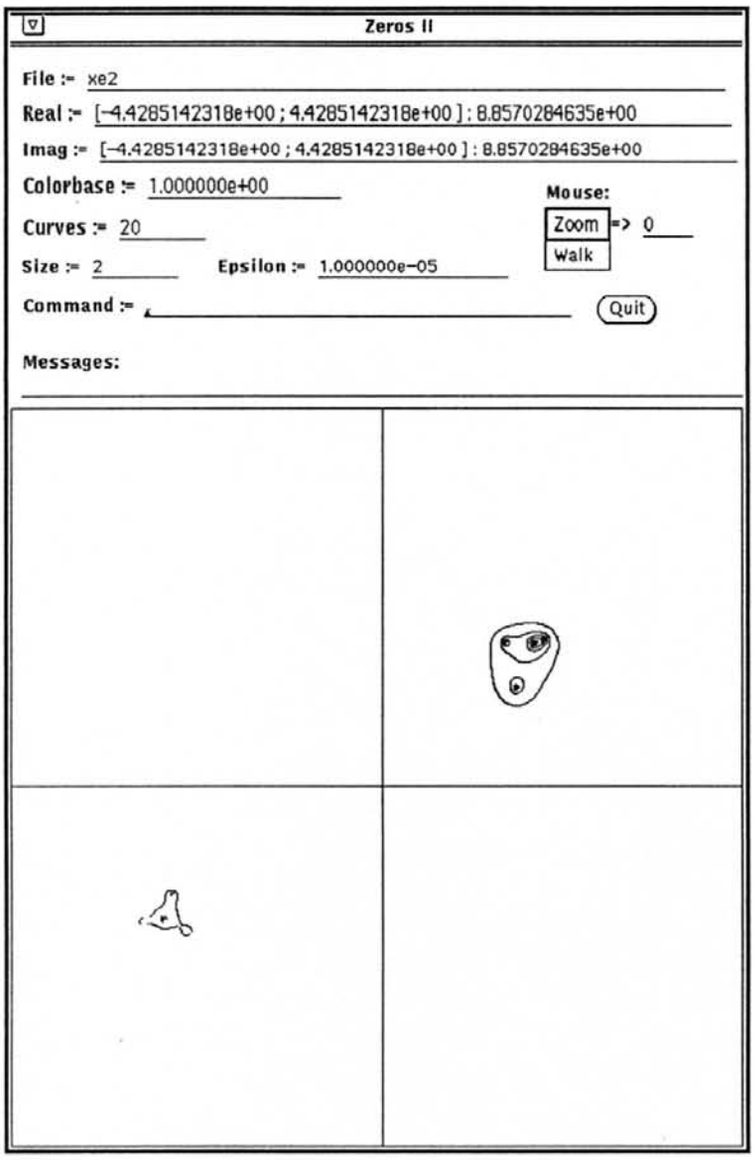

Figura 5.3: Primeira imagem para o exemplo 2.

5.6 Determinação de zeros

A fim de determinar as soluções do polinômio, não é preciso construir todas estas imagens. Simplesmente, a qualquer momento o comando **zeros** pode ser invocado, e então o processo de determinação será iniciado. Depois do fim do processo de determinação de soluções, o usuário pode ver quais foram as raízes encontradas com o comando **locate**, que marca sobre a imagem as soluções encontradas. Com um valor de $\epsilon = 10^{-10}$ para o critério de parada, o sistema encontra os valores citados na Tab. 5.3 para as raízes (comparados com as raízes originais).

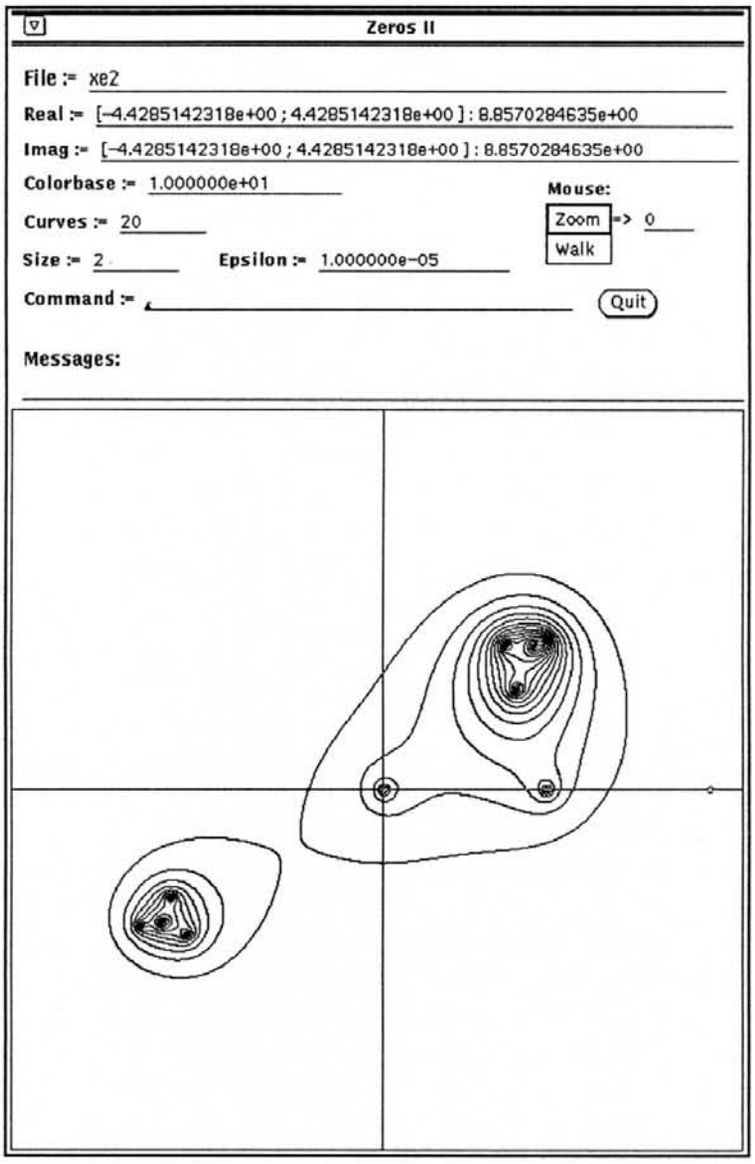

Figura 5.4: Imagem ajustada, centro = 10.

Em outros testes foram usados polinômios de grau mais alto, e os resultados obtidos foram bastante satisfatórios. Usando-se um ϵ igual a 10^{-15}, as respostas obtidas ficaram muito próximas das verdadeiras. No caso de alguns polinômios de grau 32, por exemplo, as soluções obtidas distanciaram-se no máximo 10^{-10} das soluções originais, podendo até coincidir em todas as decimais fornecidas.

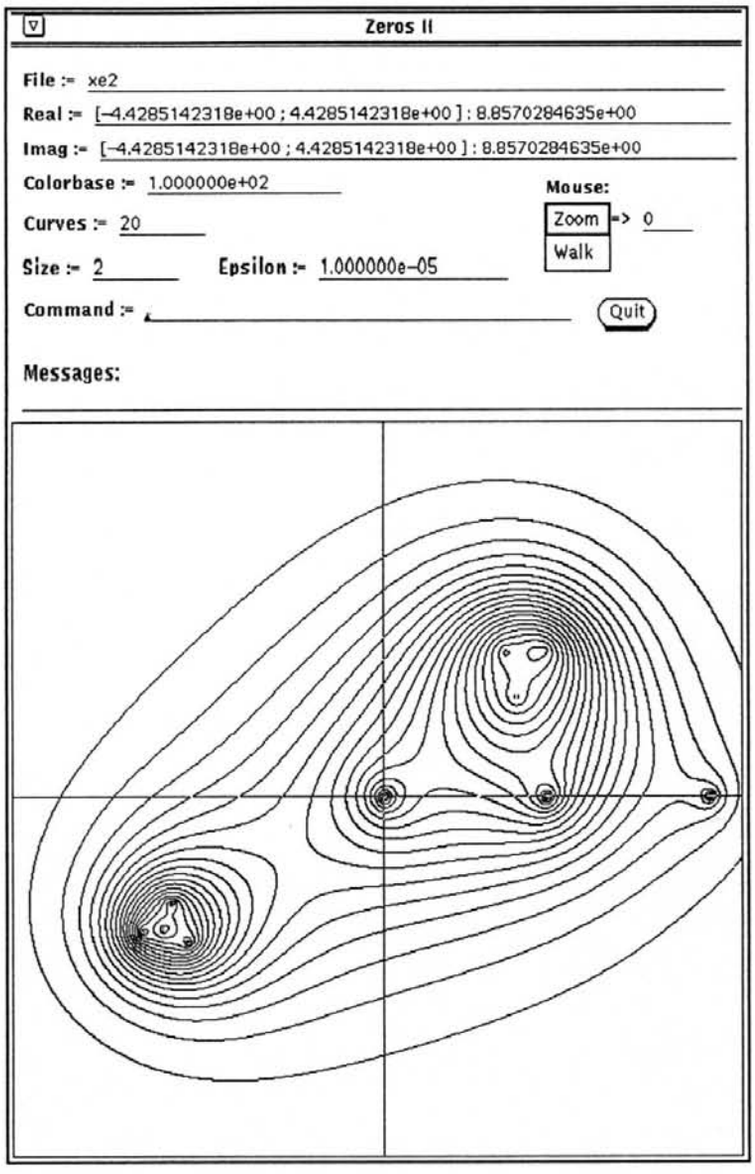

Figura 5.5: Imagem reajustada, centro = 100.

5.7 Operação de zoom

A qualquer momento pode-se usar o mouse para a operação de zoom. Trazendo o cursor por sobre a imagem, um dos vértices da área a ser ampliada deve ser selecionado. Em seguida, escolhe-se o vértice oposto, definindo um retângulo. Esta operação, bem como a imagem da qual ela resulta, está exemplificada nas Figs. 5.5 e 5.6. A zona escolhida mostra em mais detalhe as quatro raízes presentes no terceiro quadrante da imagem.

Para mostrar o erro existente quando o usuário aprofunda em demasia o zoom, encontrando erros de avaliação perto de regiões de instabilidade, pode-se usar um po-

Tabela 5.3: Comparação entre valores originais e obtidos numericamente.

Valores originais	Valores encontrados
+0.000000000000 +0.000000000000 i	-6.54464787e-16 +3.40627061e-15 i
+2.000000000000 +0.000000000000 i	+1.999999999998 -2.61787840e-13 i
+4.000000000000 +0.000000000000 i	+4.000000000000 +1.65748580e-12 i
+1.850246813079 +1.756946771292 i	+1.850246813096 +1.756946771272 i
+1.965332057777 +1.787299225008 i	+1.965332057748 +1.787299225007 i
+1.628246898124 +1.222487001318 i	+1.628246898125 +1.222487001337 i
+1.498257403028 +1.766072932988 i	+1.498257403041 +1.766072932988 i
-2.652317987593 -1.618281870903 i	-2.652317987610 -1.618281870770 i
-2.363856618462 -1.770430240673 i	-2.363856618465 -1.770430240708 i
-2.557781162931 -1.308687000679 i	-2.557781162930 -1.308687000721 i
-2.906468379733 -1.654290500867 i	-2.906468379715 -1.654290500923 i

linômio com raízes duplas, por exemplo. Desta forma, define-se para fins de ilustração o seguinte polinômio:

$$p(z) = z^5 - 20z^4 + 148z^3 - 498z^2 + 747z - 378$$

O arquivo que o descreve é:

```
(   1, 0) * z ^ 5
( -20, 0) * z ^ 4
( 148, 0) * z ^ 3
(-498, 0) * z ^ 2
( 747, 0) * z ^ 1
(-378, 0) * z ^ 0
```

O polinômio acima possui três raízes simples e uma raiz dupla. Elas são: 1, 3, 3, 6, 7. Ao ser feito um zoom sobre a raiz dupla, obtem-se imagens como as da Fig. 5.7. As curvas de nível estão completamente indefinidas, e fazem com que a imagem apareça de forma confusa. Evidentemente, a partir deste tipo de informação um

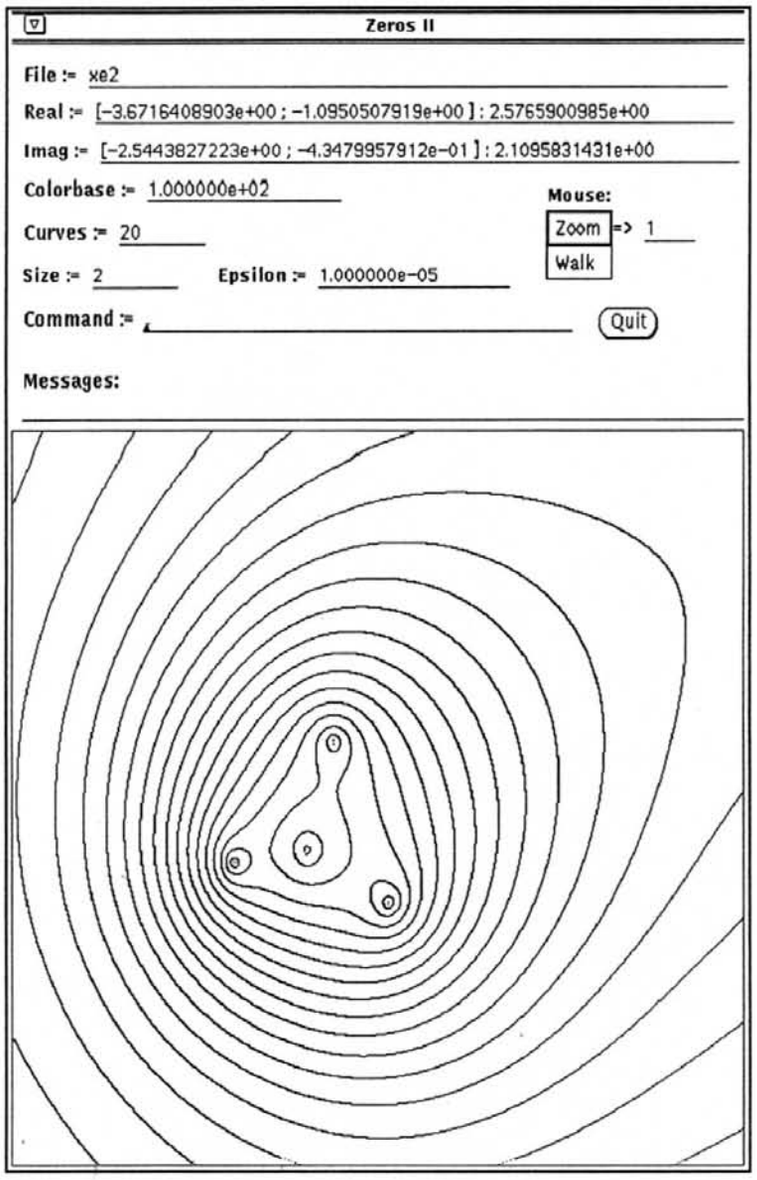

Figura 5.6: Imagem da área selecionada.

processo de determinação de raízes pode fornecer quase qualquer tipo de resposta ao tentar determinar esta raiz, sendo em geral merecedor de pouca confiança. Entretanto, através da apresentação de imagens, pode-se, além de detectar a possível instabilidade numérica, perceber aproximadamente a localização da raiz. Por meio dos dados mostrados na interface pode-se perceber que a área exibida é bastante pequena. Entretanto, se a mesma idéia de fazer ampliações fosse seguida no caso de uma raiz simples e não dupla, este tipo de erro surgiria em uma área aproximadamente 10^{14} vezes menor.

Usando-se operações implementadas na ferramenta, pode-se evitar o uso dos métodos numéricos, que serão provavelmente sujeitos a falhas, aproveitando a existência

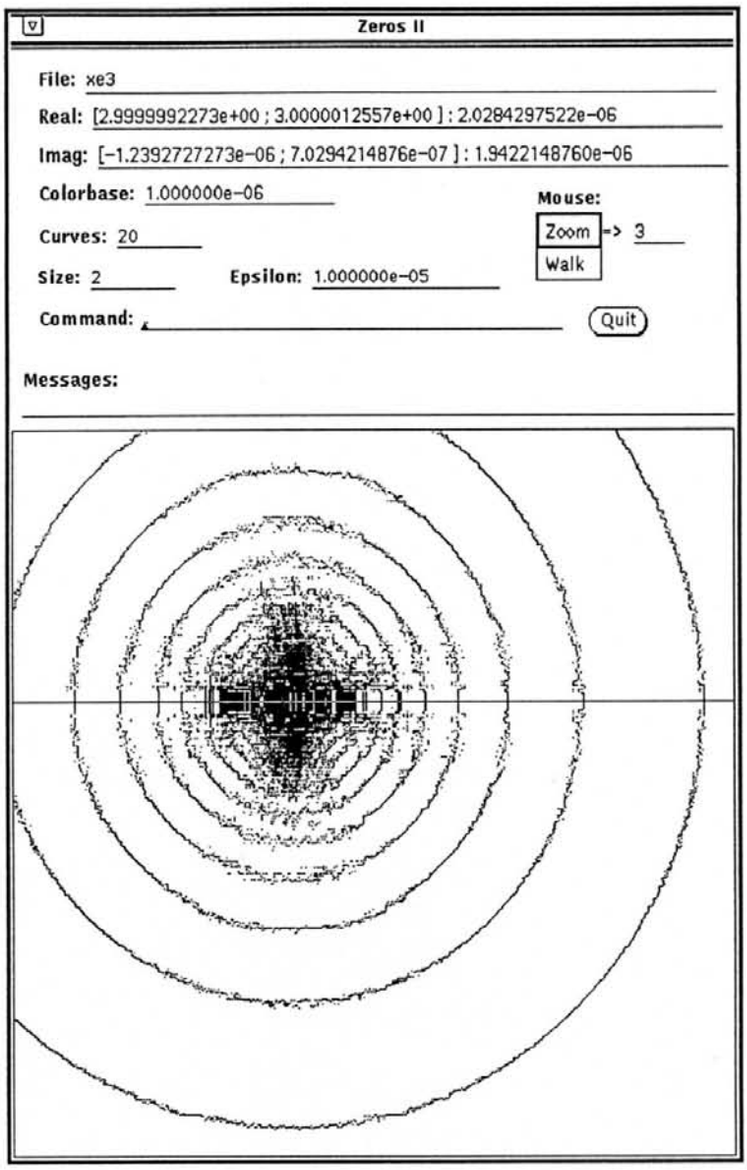

Figura 5.7: Erros próximos a raiz múltipla.

das imagens. A construção das imagens permite que, apesar dos valores numéricos incertos *em geral*, possa-se perceber *aproximadamente* a posição da raiz procurada, pois existe uma coerência estatística na distribuição das cores ou curvas de nível. A existência desta coerência faz com que o olho possa perceber e identificar a posição da raiz. Ou seja, existe um meio puramente visual de identificar uma raiz, que dispensa eventuais avaliações que podem trazer instabilidade. Fazendo uso desta capacidade do mecanismo visual humano, pode-se determinar uma raiz apenas através de ampliações sucessivas e algumas operações bastante simples, que por exemplo caminhem sobre a imagem com um cursor e mostrem o valor de z correspondente à posição do cursor. Assim, um usuário

pode conduzir o cursor até a solução que pode ser identificada visualmente, determinando em seguida o valor de z correspondente.

6 DESEMPENHO E ALGORITMOS ESPECIAIS

No que se refere a detalhes de desempenho o ponto fraco do sistema provavelmente será a rotina de elaboração da imagem, pois ela deve fazer um número muito grande de avaliações do polinômio. Afinal, deve ser feito o preenchimento de uma área de $Res\_x \times Res\_y$ pixels, onde Res_x e Res_y são as resoluções da tela nas dimensões x e y, respectivamente. Sendo assim, tornar-se-ia necessário efetuar $Res\_x \times Res\_y$ avaliações de $p(z)$. Em uma tela de 500×500 pixels, isto significa um total de 250000 avaliações. Evidentemente, tal montante não pode ser considerado pequeno, pois com somente uma pequena fração dele poder-se-ia determinar todos os zeros de $p(z)$. Para tornar o esforço de criar a imagem mais compatível com o esforço de determinar os zeros de $p(z)$, deve-se diminuir o número de avaliações polinomiais usadas na confecção da imagem.

Em princípio, cada pixel da imagem corresponde a uma avaliação de $p(z)$ e a um mapeamento de cor. A primeira sugestão pode ser aumentar o tamanho dos pixels, pois não seria necessária uma quantidade de detalhes tão grande quanto a fornecida por uma avaliação a cada pixel isolado. Implementando esta idéia, pode-se fazer uma avaliação para um grupo de pixels (uma *célula*), e marcar a todos os pixels pertencentes à célula com a cor encontrada. Por exemplo, uma célula poderia ser considerada como um bloco de 3×3 pixels. Sendo feito este agrupamento de pixels em células de lado n, passa-se a ter não mais $Res\_x \times Res\_y$ avaliações, mas $Res\_x \times Res\_y / n^2$. Assim, diminui em muito o total de avaliações necessário à montagem da tela.

A seguir pode ser considerado que a confecção de uma imagem feita varrendo-se simplesmente as células e plotando os pixels correspondentes implica em muito trabalho inaproveitado, pois freqüentemente as avaliações são feitas somente para se constatar que a cor da célula atual é a mesma cor da célula anterior. Uma forma de evitar tal desperdício, fazendo uso de uma provável coerência das células em uma linha poderia diminuir em muito o número de avaliações em células. Após estas considerações, pode ser proposto o seguinte algoritmo para a construção do mapa de $p(z)$:

Dados de entrada : *Res_x*, *Res_y*, cellsize (tamanho das células), scanspeed.

```
int  i, j, oldc, oldi, c, step, base, mid, cmid;

for (j = 0; j < RES_Y; j += cellsize)
{
        oldi = 0;
        step = 1;
        oldc = (int) pixel_color (0, j);

        for (i = cellsize; i < RES_X; i += step * cellsize)
        {
                c = (int) pixel_color (i, j);
                if (c != oldc)
                {
                    i -= step * cellsize;
                    fillrect (oldi, j, i, j + cellsize, oldc);
                    oldi = i;
                    oldc = c;
                    step = 1;
                }
                else
                        step += scanspeed;
        }

        oldc = (int) pixel_color (RES_X - 1, j);

        if (oldc != c)
        {
                base = oldi;
                oldi = i - step * cellsize;
                i = RES_X;
                while (abs (i - oldi) > 1)
                {
                        mid = (i + oldi) / 2;
                        cmid = (int) pixel_color (mid, j);
                        if (c == cmid)
                                oldi = mid;
                        else
                                i = mid;
                }
                fillrect (i, j, RES_X, j + cellsize, oldc);
                fillrect (base, j, i, j + cellsize, c);
        }
        else
                fillrect (oldi, j, i, j + cellsize, c);
}
```

No trecho acima, a função pixel_color() recebe as coordenadas x e y do pixel a ser plotado, converte-as para valores reais para determinar as partes real e imaginária

de z e avalia a cor usada para aquele valor de z, sendo portanto uma adaptação da função cor() anteriormente apresentada. A função fillrect() recebe as coordenadas de dois vértices opostos de um retângulo e uma cor, pintando então o retângulo e seu interior.

O algoritmo apresentado funciona da seguinte maneira: a tela é percorrida por linhas de células, e dentro de cada linha as células são analisadas. Esta análise assim ocorre: a cada célula examinada, verifica-se se sua cor é igual à cor da célula anterior. Se as cores forem iguais, verifica-se a cor algumas células mais adiante, cada vez mais longe. À medida que uma mesma cor se repete dentro de uma linha, o algoritmo vai cada vez mais longe verificar a continuidade de cores. Quando for detectada uma troca de cor, acha-se o pixel exato onde ocorreu a troca e o bloco de células anterior é pintado, ou seja, todas as células que possuíam a cor antiga. Procura-se então pela nova cor encontrada, a partir do pixel onde ela aparece pela primeira vez.

No algoritmo, a variável `scanspeed` regula o avanço quando à procura de células de mesma cor. Por exemplo, se fosse usado um valor de `scanspeed` igual a 3, e o tamanho de cada célula fosse também 3, poder-se-ia ter as seguintes avaliações em uma linha toda de mesma cor (linha de 500 pixels):

Pixel (x) : 0, 3, 9, 27, 54, 90, 135, 189, 252, 333, 393, 492, 499 (fim da tela).

Portanto, neste exemplo treze avaliações em uma linha permitem assegurar com razoável grau de certeza que todas as células são de mesma cor. Esta certeza não é absoluta, entretanto. Existem casos em que este algoritmo apresenta pequenas falhas para certos valores de `scanspeed`, pois pode-se saltar pequenas áreas de cor diferente, e neste caso o valor `scanspeed` deve ser alterado para que haja a correção. Fazendo uso dos comandos **size** ou **speed**, pode-se solucionar facilmente o problema.

Para melhor ilustrar o funcionamento para este exemplo, a Fig. 6.1 mostra os pixels avaliados inicialmente. Os pixels são considerados em grupos de três, correspondendo aos pixels de cada célula. Os pixels 0, 3, 9 e 27 pertencem a diferentes células, e são avaliados da seguinte forma:

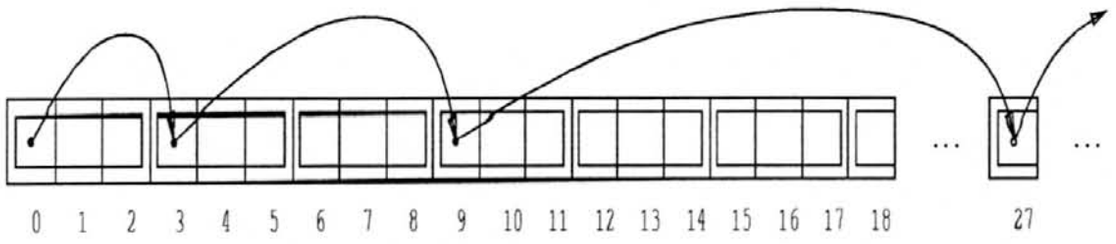

Figura 6.1: Exemplo de avaliação nos pixels.

- Em primeiro lugar é amostrado o pixel 0, pertencente à célula 1.
- Em seguida é amostrado o pixel 3, pertencente à segunda célula. Se a cor desta não for a mesma da célula 1, o algoritmo pinta a célula 0. Se as cores forem as mesmas, o algoritmo passa a fazer uma amostra de cor no pixel 9, pertencente à célula 3. Neste ponto, devido à coincidência das cores das células 0 e 1, a célula 2 não foi analisada. O algoritmo segue assim avançando cada vez mais e evitando cada vez mais células.
- Se entre duas células analisadas houver uma diferença de cores, então o algoritmo faz uma pesquisa entre as células que originaram a diferença e determina a fronteira entre as cores. Ao ser encontrada a fronteira, o bloco de células até ela é pintado com a cor da primeira célula.

Com a finalidade de mostrar a correlação entre os valores de `scanspeed` e o desempenho do algoritmo, foi obtida uma tabela (Tab. 6.1) de resultados para um caso modelo, ou seja, uma imagem que pode ser considerada uma representante típica das imagens do sistema. A escolha desta representante se deve ao problema de que o desempenho depende essencialmente das cores existentes na imagem sendo feita, variando caso a caso. No exemplo, usou-se uma tela de 550 por 550 pixels, ou seja, um total de 302500 pixels. Para diversos valores de `scanspeed` e de tamanho de célula, foi medido o número de avaliações do polinômio usadas na elaboração da imagem. A tabela 6.1 apresenta dados referentes à variação no custo de uma imagem, à medida que é alterado o valor de `scanspeed`.

Tabela 6.1: Comparação de desempenho por tamanhos de célula.

speed	Tam. 1	Tam. 2	Tam. 3	Tam. 5	Tam. 8	Tam. 12
0	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %	100.00 %
1	55.13 %	69.58 %	82.00 %	95.48 %	108.35 %	115.11 %
2	58.42 %	76.34 %	90.62 %	106.84 %	120.75 %	121.96 %
3	63.78 %	83.36 %	98.20 %	117.89 %	123.39 %	123.97 %
4	69.00 %	90.72 %	105.72 %	123.64 %	125.32 %	117.68 %
5	73.86 %	96.95 %	116.24 %	127.65 %	126.85 %	116.15 %
6	78.84 %	102.25 %	122.98 %	128.56 %	119.70 %	116.55 %
7	83.65 %	108.14 %	127.98 %	130.39 %	124.07 %	111.70 %
8	88.49 %	115.48 %	130.47 %	129.65 %	117.06 %	111.89 %
9	93.87 %	122.45 %	132.14 %	127.38 %	117.64 %	107.60 %
10	97.85 %	126.05 %	130.91 %	125.97 %	112.45 %	103.60 %
12	106.97 %	133.28 %	133.68 %	122.95 %	110.77 %	107.79 %

É preferível adotar como parâmetro de desempenho o número de *avaliações* requeridas para a construção de uma imagem, pois os tempos medidos em um sistema operacional multiprocessado tendem a apresentar alguma inexatidão. Na tabela 6.2, comparou-se o número de avaliações de $p(z)$ feitas por imagem com o desempenho de um algoritmo que avalie todas as células da imagem (topo das colunas). Em seguida, os valores foram transformados em porcentagens.

Na tabela 6.2, as comparações foram feitas com o número de avaliações necessário para construir a imagem mais detalhada possível, ou seja, com o tamanho de célula igual a 1 e `scanspeed` igual a 0. Assim, pode-se julgar a economia do algoritmo se comparada com uma varredura e análise de todos os pixels da imagem. A diferença é extrema usando-se tamanhos de células grandes, mas a falta de detalhes da imagem obtida é de tal ordem que impede quase qualquer apreciação dos resultados. Portanto, um equilíbrio deve ser mantido entre o tamanho da célula e o grau de detalhe desejado.

Sobre as imagens geradas durante a obtenção das tabelas acima, pode-se dizer que:

Tabela 6.2: Comparação de desempenho por total de avaliações.

scanspeed	Tam. 1	Tam. 2	Tam. 3	Tam. 5	Tam. 8	Tam. 12
0	100.00 %	28.51 %	13.35 %	5.15 %	2.20 %	1.05 %
1	55.13 %	19.84 %	10.95 %	4.92 %	2.38 %	1.21 %
2	58.42 %	21.76 %	12.10 %	5.51 %	2.65 %	1.29 %
3	63.78 %	23.76 %	13.11 %	6.08 %	2.71 %	1.31 %
4	69.00 %	25.86 %	14.11 %	6.37 %	2.75 %	1.24 %
5	73.86 %	27.64 %	15.52 %	6.58 %	2.79 %	1.22 %
6	78.84 %	29.15 %	16.42 %	6.62 %	2.63 %	1.23 %
7	83.65 %	30.83 %	17.08 %	6.72 %	2.73 %	1.18 %
8	88.49 %	32.92 %	17.42 %	6.68 %	2.57 %	1.18 %
9	93.87 %	34.91 %	17.64 %	6.56 %	2.59 %	1.13 %
10	97.85 %	35.94 %	17.48 %	6.49 %	2.47 %	1.09 %
12	106.97 %	38.00 %	17.85 %	6.34 %	2.43 %	1.14 %

1. À medida que as células são aumentadas, a qualidade decresce sensivelmente. Tamanhos maiores do que 5 geram imagens geralmente grosseiras. Imagens aceitáveis podem ser obtidas, usualmente, com tamanhos de célula abaixo de 5.

2. O ligeiro acréscimo no número de avaliações decorrente no aumento de `scanspeed` se deve ao fato de que o algoritmo passa a avançar *demais* entre as células examinadas, e por isso freqüentemente existem trocas de cores, quando então o local exato da troca deve ser encontrado, exigindo mais avaliações.

3. À medida que o tamanho de célula é aumentado, o valor de `scanspeed` passa a ser de menor relevância, pois todos os valores tendem a se aproximar do valor obtido para `scanspeed` igual a zero, ou seja, *sem* a procura de células repetidas.

4. Em geral, as melhores imagens são obtidas com tamanhos de célula entre 1 e 3, e valores de `scanspeed` também entre 1 e 3. As combinações destes valores costumam fornecer boas imagens em tempos bem mais baixos do que os tempos

de uma varredura total da tela. Normalmente um valor de `scanspeed` igual a 1 apresenta os melhores resultados. Este é o valor default do sistema.

7 EQUAÇÕES NÃO POLINOMIAIS

A construção de ferramentas que operem sobre equações não polinomiais pode apresentar uma série de detalhes e diferenças que não devem ser desprezados ou ignorados. Entre outros, o desconhecimento e a impossibilidade de obtenção de cotas para as raízes, por exemplo, podem dificultar sobremaneira o trabalho de determinação das soluções. Este capítulo pretende detalhar alguns destes problemas que impedem a construção de uma ferramenta segura para equações não polinomiais.

Deve ser notado que o problema de resolver equações (sejam ou não polinomiais) para uma variável complexa é um caso especial da resolução de sistemas de equações não lineares. Mais precisamente, o caso de uma equação e duas incógnitas, pois qualquer número de equações pode ser somado dando origem a uma única equação em duas variáveis a ser resolvida. Portanto, pode-se assumir que a solução desta classe de problemas seja (apesar de ser um caso restrito) tão trabalhosa quanto a resolução de sistemas não lineares.

7.1 Número de raízes

Evidentemente, o grau de um polinômio é um limite para o número de raízes deste. Não só limita, mas indica exatamente quantas raízes são existentes e devem ser encontradas. Ou seja, o conhecimento básico de quanto trabalho deve ser feito é previamente disponível. Não está ainda em discussão a exatidão das soluções obtidas, mas sim seu número, pois o simples fato de saber-se quando o trabalho está terminado e não existem outras raízes ainda à espera de identificação é extremamente vantajoso. Este conhecimento inexiste quando se trata de equações não polinomiais, pois neste caso não se sabe (ou seja, não existe algoritmo capaz de determinar) quantas raízes ainda esperam ser encontradas. Assim, a qualquer momento o sistema é incapaz de saber se está próximo

do fim de sua tarefa. Ou seja, é geralmente impossível decidir se já foram encontradas todas as soluções.

7.2 Impossibilidade de deflação

A cada raiz encontrada em uma equação polinomial, a deflação do polinômio original ajuda a reduzir o problema, além de acelerar os métodos usados nas iterações. Adicionalmente, a deflação faz com que a raiz encontrada seja desconsiderada nas próximas iterações, evitando que esta raiz seja encontrada novamente, e portanto não é desperdiçado tempo em sua determinação e testes sobre sua possível presença entre as raízes conhecidas. Devido à dificuldade de deflação, algoritmos que operam sobre equações não polinomiais operam sempre à mesma velocidade, sem tirar vantagem das raízes já encontradas, e provavelmente encontrando algumas delas mais de uma vez. Uma alternativa a ser testada seria criar um polinômio que tem como raízes as raízes já encontradas para a equação. Este polinômio passa a ser o denominador da equação na qual o usuário está trabalhando, dividindo-a ao final de uma avaliação. Esta estratégia talvez pudesse evitar a repetição de soluções, mas encareceria progressivamente as avaliações da equação sendo resolvida.

7.3 Inexistência de cotas

No caso específico de sistemas que, como este, baseiam-se em uma interface gráfica para conduzir o usuário na realização de sua tarefa, é especialmente difícil trabalhar sem o conhecimento de cotas para as raízes. Isto porque na construção das imagens o sistema deve ser capaz de determinar uma região que englobe *todas* as raízes, para que ao usuário seja fornecida uma imagem que contenha todos os pontos de seu interesse, evitando que ele tenha de procurá-los por si próprio. A impossibilidade da avaliação das cotas interfere no funcionamento do sistema, pois este não pode apresentar corretamente todas as raízes. A existência de cotas solucionaria em parte o problema do desconhecimento

do número total de raízes, pois o usuário teria a possibilidade de procurar as soluções por si mesmo, uma a uma. Esta alternativa seria possível se todas as soluções estivessem contidas em uma mesma imagem e portanto o usuário pudesse eliminá-las à medida que as determina. Entretanto, dada a inexistência de cotas, não é possível nem mesmo garantir que uma dada imagem contém todas as raízes.

7.4 Um algoritmo alternativo

Embora, pelas dificuldades apresentadas acima, seja impossível implementar um sistema que determine todas as soluções de uma equação não polinomial de forma totalmente automática, pode ser oferecida uma solução parcial ao problema. Esta solução pode ser chamada de "solução fraca", pois deve ser dirigida pelo usuário e é em grande parte dependente dele para seu sucesso. A alternativa consiste em fazer os mesmos tipos de imagens usados para equações polinomiais, pois estas imagens podem ser produzidas para qualquer tipo de equação.

Ora, o problema reside essencialmente na determinação numérica das soluções. Se é inviável adotar uma alternativa totalmente automática, pode-se sugerir que o usuário oriente um cursor sobre a tela, aproximando-o da solução que se deseja conhecer. Em seguida, o sistema assume o controle, partindo do ponto do cursor e tentando determinar uma (e somente uma) raiz. Provavelmente, o sistema determinará a raiz mais próxima do cursor, satisfazendo a necessidade atual do usuário. Para operar sobre equações não polinomiais, no entanto, existe a dificuldade de avaliar a derivada primeira da função. Para resolver este problema particular, podem ser adotadas duas alternativas em especial, dentre todas as outras possíveis:

- Usar algoritmos de derivação automática, que façam a avaliação simultânea da função e de sua derivada. Para uma descrição detalhada deste processo, veja [RAL81].

- Adotar algoritmos que evitem a necessidade de avaliação de derivadas, procurando mínimos da superfície originada pela função por meio de avaliações intervalares sobre uma quadtree que divide o domínio inicial, por exemplo. Estes processos, se baseados em aritmética intervalar, garantirão que todas as soluções dentro da imagem de onde foi disparado o processo serão encontradas. Neste caso, seriam achadas todas as raízes contidas na imagem inicial.

8 EXPANSÕES FUTURAS

Uma vez obtida uma ferramenta que implementa uma solução para o problema proposto, pode-se especular sobre novas capacidades a serem incluídas no ambiente implantado, ou modificações sobre operações já desenvolvidas. Nesta fase, torna-se possível traçar novos caminhos a serem seguidos no intuito de aperfeiçoar um sistema existente, dando-lhe uma nova orientação, pois já existe uma experiência sobre o que é factível ou não dentro da proposta inicial. Este é o momento, portanto, de trabalhar novas idéias a desenvolver num futuro próximo.

8.1 Uso de três dimensões

A projeção usada para a tela é bidimensional, e as razões para tanto já foram explicadas anteriormente. No entanto seria possível gerar sob o comando do usuário cenas tridimensionais que representassem a superfície em estudo. Estas imagens, logicamente, seriam mais caras do que as imagens comuns, mas sua utilidade não pode ser descartada. Elas são especialmente úteis com o objetivo de dar uma idéia geral do polinômio, pois a visão em duas dimensões não é sempre bem aceita por todos os usuários. Portanto, uma forma de representação tridimensional seria um avanço bem recebido por parte de alguns usuários mais renitentes em usar a representação 2D.

8.2 Uso de outras aritméticas

A existência de erros na aritmética usada é atualmente incontornável, já que não se pode usar outras aritméticas ou representações numéricas nos compiladores encontrados usualmente. Por exemplo, seria desejável usar a aritmética racional nos cálculos, mas esta não se encontra disponível. Uma possível maneira de solucionar o problema seria

implementar a aritmética que desejamos, criando uma biblioteca de operações a partir daí disponíveis no sistema. Atualmente, as facilidades de programação oferecidas por linguagens orientadas a objetos seriam de vital importância para o desenvolvimento rápido e simplificado desta solução. Por exemplo, seria possível implementar toda a nova ferramenta em C++, obtendo-se as vantagens da programação orientada a objetos a um custo bem mais baixo do que implementando uma biblioteca de rotinas própria. O uso destas "aritméticas particulares" para efetuar os cálculos seria especialmente interessante em um sistema orientado a objetos, pois possibilitaria a construção muito mais rápida de operações bastante complexas sobre polinômios. Da mesma maneira, permitiria a realização de operações que são dificultadas sobremaneira pela presença, em alguns casos, de erros de arredondamento. Por exemplo, a determinação da multiplicidade de raízes poderia ser feita sobre números racionais, usando-se objetos que são os próprios polinômios representados por estes números.

Além do uso de uma aritmética particular, pode-se implementar operações simbólicas que formariam um núcleo bastante poderoso. Estas outras capacidades como divisão polinomial e fatoração, entre outras, facilitariam a avaliação de expressões que podem ser obtidas de um polinômio através destas operações. Se fosse adicionada a capacidade de associar estas expressões a variáveis, seria possível incluir na ferramenta uma forma de comando que computasse expressões algébricas sobre o polinômio.

8.3 Uso de equações não polinomiais

Uma série de problemas se apresentam para o desenvolvimento de um sistema para resolver equações não polinomiais. Estes empecilhos foram explorados no capítulo 7, e se não podem ser completamente resolvidos, podem ser contornados desde que o usuário assuma uma parcela maior de responsabilidade sobre o uso do sistema. Isto significa que se pode ter um "sistema fraco", que não é capaz, por exemplo, de fornecer cotas para as raízes de uma equação. Este sistema poderia no entanto ajudar o usuário a determinar soluções pela inspeção visual, ou seja, por meio de imagens. A grosso modo,

o usuário assumiria a tarefa de determinar visualmente regiões que contivessem as raízes e indicá-las para o sistema, que a partir daí poderia fazer a procura. Nesta proposta, no entanto, o usuário deve assumir um papel de maior responsabilidade, pois dele dependem as estimativas iniciais para as soluções.

Da mesma forma, se o número de raízes é desconhecido, o usuário deve assumir o controle sobre o mecanismo de procura, determinando quantas ou quais das soluções devem ser encontradas e assegurando-se de que existem condições para a convergência do sistema.

8.4 Sistemas de equações

Como já foi sugerido, pode-se adaptar o mesmo tipo de raciocínio usado na obtenção das imagens de $p(z)$ para gerar imagens a partir de um sistema de equações de duas incógnitas. Isto porque quaisquer equações poderiam ser somadas, obtendo-se uma única equação, cujas raízes são as soluções desejadas para o sistema inicial. Usando-se o mesmo tipo de ferramenta, somente deve estar presente o cuidado adicional de verificar todos os resultados obtidos, pois a presença de mínimos na superfície não indica que estes sejam obrigatoriamente soluções do sistema. Por exemplo, o sistema de equações abaixo possui duas soluções:

$$\begin{cases} x^2 + y^2 = 1 \\ x^2 - y = 0 \end{cases}$$

As soluções reais são os pontos onde a parábola $y = x^2$ corta o círculo unitário.

Assim sendo, a soma das equações no exemplo acima originaria a equação

$$2x^2 + y^2 - y = 1$$

que pode ser resolvida para todas as quatro soluções através de uma imagem. Já o sistema

$$\begin{cases} x^2 + y^2 = 1 \\ x^2 - y = -1.1 \end{cases}$$

não possui soluções reais, mas a superfície formada pela soma das duas equações possui pontos de mínimo complexos, que são soluções do sistema. Estes pontos devem ser detectados e identificados como soluções. Neste aspecto, o trabalho em uma ferramenta para resolver sistemas de equações se aproxima do "sistema fraco" sugerido para resolver equações quaisquer, pois exige um grau bem maior de intervenção do usuário. Mesmo assim, baseando-se nos princípios expostos para equações em geral, seria possível a construção de um sistema que dependesse em maior grau do usuário e que o capacitasse a determinar soluções.

8.5 Identificação de multiplicidade

Em sua versão atual o sistema não tenta identificar possíveis raízes múltiplas, deixando estas tarefas a cargo do usuário. Devido às falhas detectadas na implementação dos números de ponto flutuante quando operando em situações limite, esta parece ser a opção mais segura para manter coerentes as informações fornecidas pelo sistema, pois aparentemente não seria difícil encontrar situações nas quais o sistema determinasse as multiplicidades de forma errônea, considerando raízes muito próximas como uma única raiz, por exemplo. Infelizmente tais questões somente poderiam ser sanadas com a instalação de uma aritmética que possua uma representação mais extensa para os números que dela fazem parte.

8.6 Edição de soluções

Uma idéia bastante interessante poderia ser desenvolvida a partir da *edição* das raízes por meio do mouse. Por exemplo, polinômios usados em sistemas de controle e outras áreas afins não devem possuir zeros com parte real positiva. Poderia ser feita uma edição sobre as raízes permitindo que o usuário escolhesse uma raiz por meio do mouse. Em seguida o polinômio original é deflacionado daquela raiz, e o usuário também teria a possibilidade de incluir novas raízes em qualquer local do plano complexo. De forma conjunta, um usuário seria então capaz de remover uma raiz indesejável, transportando-a para outro local de seu agrado. Este sistema de edição não pode ser usado livremente com quaisquer casos que se deseje solucionar, mas pode ser útil para ajustar polinômios já estáveis a fim de ser obtida uma solução melhor ou obter melhores características.

9 CONSIDERAÇÕES GERAIS

A definição e implementação da ferramenta proposta permite que sejam obtidas algumas conclusões interessantes do ponto de vista de adaptações ou de futuras implementações baseadas nos mesmos princípios. Desde considerações sobre desempenho até detecção de falhas na concepção inicial do projeto, passando por detalhes dos algoritmos existentes, pode-se tecer opiniões, baseadas evidentemente no uso e testes da ferramenta proposta. Portanto, neste capítulo serão relatadas algumas das conclusões a que se chegou no decorrer do desenvolvimento e testes deste trabalho.

9.1 Da efetividade das imagens

Para o usuário que se inicia no uso do sistema as primeiras imagens não são de compreensão imediata, fato este confirmado por algumas dúzias de pessoas que muitas vezes se aproximaram para perguntar o que as imagens representavam. Felizmente, depois de alguma explicações sobre o princípio de funcionamento do sistema tais dúvidas eram geralmente sanadas, obtendo-se a partir daí novos colaboradores, críticos e fornecedores de idéias e sugestões. As imagens foram em geral consideradas um meio eficiente de se "apresentar" um polinômio, mesmo por pessoas que por formação teriam preferido uma forma mais ortodoxa de representação. Especialmente, duas características principais do uso de imagens foram consideradas muito interessantes:

- A capacidade de apresentar de uma maneira simples e clara a distribuição das raízes, e mais ainda, fornecer uma idéia geral do comportamento do polinômio através de uma imagem foram considerados muito úteis quando se trata de obter um idéia geral do polinômio sendo estudado. Isto porque não é comum que o usuário tenha à disposição software construído para produzir e operar sobre estas representações que forneçam aspectos gerais do polinômio.

- A capacidade de apresentar os *erros* presentes na avaliação de polinômios usando-se a aritmética de ponto flutuante. Tais imagens permitem que se possa determinar a presença de problemas de avaliação sem que se tenha de avaliar as raízes do polinômio para depois descobrir que elas são incorretas. Ou em outras palavras, é muitas vezes importante obter-se esta noção da *confiabilidade* de uma resposta numérica, e a noção fornecida pelas imagens foi considerada bastante boa, mesmo a um custo relativamente mais elevado do que por outras formas.

9.2 Das operações implementadas

O conjunto de operações fornecidas no sistema parece satisfazer as necessidades de quase todos os usuários. Muitas sugestões de novos comandos foram feitas e várias foram implementadas. Foram sugeridas outras operações que não puderam ser instaladas com facilidade na ferramenta. Estes comandos adicionais exigiriam um trabalho de modificação da ferramenta existente, pois originalmente tais operações não haviam sido previstas e seriam implementadas com dificuldade na versão atual. Entretanto, os novos comandos introduzidos foram geralmente acréscimos ou variações sobre os já existentes, facilitando o uso do sistema. Ou seja, em geral a ação destes comandos poderia ser feita por uma seqüência dos comandos já existentes. Entretanto, é inegável que muitas sugestões recebidas e não instaladas seriam muito úteis, mas exigiriam transformações na estrutura interna do sistema que seriam bastante extensas.

9.3 Da automatização de tarefas

O sistema desenvolvido funciona bastante satisfatoriamente, mas exige do usuário decisões que deveriam ser evitadas, como por exemplo escolhas de centros de cores, seleções dos tamanhos de células e o valor de `scanspeed`. Tais escolhas deveriam

ser feitas de forma automática, permitindo também que o usuário fizesse suas próprias escolhas de forma eventual. Infelizmente, não foram encontradas até o momento na literatura (ou através de algoritmos desenvolvidos no decorrer do trabalho) formas de prover os valores de forma automática. Isto se deve por exemplo aos seguintes fatores, entre outros:

- A extrema variação dos valores usados como extremos no domínio da imagem, pois pode-se ter intervalos com diâmetros de 10^{-18} até em torno de 10^{30}. Evidentemente, estas variações devem ser consideradas quando são calculados os possíveis valores dos parâmetros citados. Não só os extremos da imagem representada podem variar assim, mas mesmo os valores de $|p(z)|$ dentro de uma mesma imagem.

- O grau do polinômio possui influência sobre a escolha dos valores, pois os módulos de polinômios de grau mais alto tendem a crescer muito mais depressa do que os módulos de polinômios de grau baixo. Assim, deve-se ter em mente a possível faixa de valores a ser obtida para uma imagem. Isto poderia ser feito usando-se aritmética intervalar, mas infelizmente tais avaliações tendem a superestimar em muito os valores que podem ser obtidos pelo polinômio, calculando-se valores incorretos para os parâmetros.

- A própria *forma* da superfície apresentada na imagem deve ser conhecida, pois quanto mais plana for a superfície mais próximo de zero deve estar o valor do centro de cores. Isto explica-se porque superfícies quase planas são sempre indicativas de várias raízes agrupadas ou ainda raízes múltiplas, e em geral nestas imagens o valor de $|p(z)|$ está perto de zero. A análise de quão plana é a superfície pode ser trabalhosa, exigindo muito tempo se for feita a cada operação que altera as imagens.

- O sistema deveria ser capaz de decidir que valores tornam a imagem mais representativa, o que é definitivamente uma decisão bastante subjetiva e dificilmente uma máquina poderia tomar decisões sobre estética. Além da dificuldade que

tais conceitos de representatividade ofereceriam para ser colocados em uma máquina, provavelmente seria consumido um tempo razoável em análise dos dados que comporão a futura imagem.

9.4 De outras aritméticas

O uso da aritmética fornecida pelos compiladores correntes é na maior parte das situações confiável, mas existem casos em que não são satisfeitas algumas necessidades de exatidão, especialmente quando se lida com números muito grandes ou muito pequenos. Deve ser enfatizado que a perda de dígitos em multiplicações e os erros encontrados ao se lidar com números de máquina muito próximos foram sempre problemas a serem contornados quando possível na ferramenta. Infelizmente a existência destes problemas está explicada na própria base sobre a qual o sistema foi implementado, ou seja, os números de ponto flutuante. Em qualquer ferramenta ou implementação usando tais números sempre existirão limites para os valores utilizados e portanto sempre haverá problemas em situações próximas a estes limites. Uma alternativa poderia ser implementar *toda* a aritmética sobre a qual se baseia o sistema, se possível sobre números racionais bastante longos. Por exemplo, a implementação usando-se pares de inteiros de 32 bits, ou mesmo formas ainda maiores de representação. Esta alternativa exige entretanto que *todas* as operações sejam feitas nesta aritmética, o que pode tornar-se um problema ao serem colocadas eventuais funções transcendentes. Para resolver o problema e diminuir o trabalho de implementação a melhor alternativa parece ser transportar o sistema completo para uma linguagem orientada a objetos, definindo objetos básicos que seriam os números, as operações sobre eles e assim por diante. A maior vantagem da construção de uma aritmética própria seria a possibilidade de controlar o tamanho e a precisão dos números envolvidos.

Em geral o sistema fornece resultados com exatidão bastante boa, mas existem casos em que a presença de erro é notada, e estas situações podem justificar tal

implementação orientada a objetos. No uso cotidiano, no entanto, o sistema fornece soluções bastante satisfatórias.

9.5 Do desempenho

O ponto mais delicado da ferramenta em termos de desempenho são as rotinas que fazem a construção das imagens, por seu intenso trabalho de avaliação do polinômio. Especialmente sobre polinômios de grau alto (por exemplo, grau 40), a diferença de tempo para polinômios de grau mais baixo pode ser sentida claramente. No entanto, as imagens costumam ser feitas em tempos que variam de 4 a 10 segundos, em geral. A performance nas imagens feitas com células depende, é claro, do grau de detalhe desejado pelo usuário, ou seja, do tamanho das células. As curvas de nível são sempre feitas com o maior grau de detalhe possível, e consomem aproximadamente os mesmos tempos. Já as curvas de sinal são bem mais rápidas mas também dependentes do tamanho das células. Os algoritmos usados para reduzir o número de avaliações de $p(z)$ são bastante razoáveis, mas ainda não foram esgotadas todas as opções para outras implementações. Tarde demais para uma implementação, por exemplo, surgiu a idéia de uma forma de quadtree que calcula os dados que serão usados pelas chamadas recursivas *antes* que elas sejam feitas, reduzindo boa parte da redundância das avaliações. Cálculos iniciais mostraram que esta forma de quadtree seria 3.2 vezes mais barata do que a tradicional, em termos de número de avaliações.

9.6 Das expansões

Dentre as expansões sugeridas anteriormente, todas poderiam ser implementadas em uma versão bastante modificada do sistema. Tal versão deve ser planejada, no entanto, para manter o mesmo tipo de conceitos básicos desenvolvidos na versão atual, ou seja, apresentação de imagens e simplicidade no uso da interface. As alternativas de

desenvolvimento e expansão apresentadas anteriormente são factíveis mas o sistema deve obrigatoriamente passar por uma reformulação, especialmente para possibilitar a adição das operações sobre funções não polinomiais. O caminho parece ser a incorporação destas novas características ao sistema, de forma a compor um ambiente fechado para a solução de equações quaisquer. No entanto, esta versão deve ser planejada de forma bastante cuidadosa, especialmente no que tange a controle de erros e exatidão nas avaliações.

Infelizmente o processo de construção desta segunda versão do sistema deve ser bastante lento se for implementada uma nova aritmética, pois esta deveria ser fornecida para as quatro operações mais uma série de funções transcendentes tais como raiz quadrada, logaritmos, exponenciais e quaisquer outras que sejam necessárias para que se avaliem funções de qualquer tipo, não apenas polinomiais. Garantir a validade e exatidão destas funções não é tarefa simples, e consumiria bastante tempo.

9.7 De outros sistemas existentes

Embora não existam exemplos de sistemas similares produzidos no âmbito do Curso de Pós-Graduação em Ciência da Computação desta Universidade, existem outros sistemas que permitem a obtenção de imagens e resultados semelhantes, embora sejam construídos como ferramentas bem mais poderosas, que não estão restritas somente à determinação de soluções de equações. Talvez o melhor exemplo (e sem dúvida o mais poderoso) seja o conhecido programa *Mathematica* (veja [WOL91]), que faz uma enorme série de operações sobre várias entidades matemáticas, não somente polinômios.

Tal programa, entretanto, difere deste sistema em sua proposta básica por ser de cunho extremamente geral, podendo resolver uma enorme gama de problemas. O sistema aqui proposto opera sobre um conjunto bastante restrito de problemas, o que permite uma interface bastante especializada construída para ser dedicada à questão básica de transmitir as informações sobre os polinômios ao usuário. É inegável que a estrutura interna de *Mathematica* forneceu algumas das idéias para as futuras extensões deste

sistema, especialmente a implementação de uma "aritmética particular". Bom número das capacidades e características de *Mathematica* seriam desejáveis nesta ferramenta. No entanto, a maior parte destas capacidades é de manipulação de dados, pois em termos de interface e diálogo com o usuário *Mathematica* não guarda inovações importantes que representem grande vantagem se absorvidas por esta ferramenta.

Em geral sistemas comerciais são bastante abrangentes em seu escopo de tarefas, permitindo que sejam solucionados muitos tipos diferentes de problemas. Estes sistemas portanto não podem oferecer uma interface totalmente dedicada e direcionada à resolução de um conjunto menor e mais específico de problemas, sendo prejudicados no processo de comunicação com o usuário quando comparados com sistemas especializados. Para obter o mesmo tipo de imagens em um programa de cunho genérico, o usuário é forçado a programar em suas linguagens internas ou a fazer uso de pacotes fornecidos pelos fabricantes do programa.

Sendo assim, é difícil fazer comparações entre sistemas tão díspares, em capacidade, tamanho e versatilidade. Infelizmente, outros sistemas que poderiam talvez ser comparados com a ferramenta definida neste trabalho não se encontram disponíveis para avaliação.

10 CONCLUSÃO

Após o planejamento, desenvolvimento e testes da ferramenta proposta neste trabalho, existem conclusões que podem ser de utilidade quando da elaboração de outros sistemas semelhantes nesta Universidade ou em outros ambientes de pesquisa. Neste capítulo serão detalhadas algumas desta conclusões, às quais podem ser somadas as idéias para futuras expansões relatadas anteriormente. As futuras expansões já referidas são fruto de observações feitas sobre o sistema apontando necessidades ou mesmo deficiências, e conduzem a novos caminhos para o desenvolvimento deste trabalho. Adicionalmente, devemos citar algumas das conclusões sobre a proposta inicial e a ferramenta obtida. Elas são:

- O uso de imagens para fazer a apresentação dos resultados foi uma ótima escolha, sendo obtidos muito bons resultados quando tais imagens são apresentadas. Eventualmente são necessárias algumas explicações adicionais sobre sua obtenção e construção, mas assim que o mecanismo de geração destas é assimilado, as pessoas passam a considerá-las como uma boa forma de apresentação. Especialmente, foi salientada mais de uma vez a importância de uma noção clara sobre o comportamento *global* dos polinômios, obtendo-se uma idéia clara de seu comportamento e distribuição de suas raízes.

- A opção por uma interface baseada em comandos foi considerada válida, pois evita o uso de vários cardápios que eventualmente teriam tantas funções e seriam em número tão grande que confundiriam o usuário. Apesar das vantagens do uso de cardápios, a portabilidade do sistema também seria por demais comprometida com seu uso intensivo. Isto se deve à pouca portabilidade de ambientes gerenciadores de janelas e ademais, a possibilidade de serem produzidos sinônimos para os mesmos comandos é bastante interessante.

- Os algoritmos produzidos para a obtenção das imagens são bastante eficientes e podem ser considerados uma das partes mais interessantes do funcionamento

interno do sistema. Eles proporcionam uma economia bastante considerável se comparados com algoritmos que trabalham por meio de varredura exaustiva da tela. Seu desempenho pode (e deve) ser aumentado ainda mais em uma versão futura, pois já existem novas sugestões a testar.

- Os resultados numéricos fornecidos pela ferramenta são bastante bons, exceto em casos onde existe degeneração nas avaliações ou situações muito próximas dos limites da máquina. Em situações cotidianas ou mesmo não tão comuns (como em raízes múltiplas, por exemplo), o sistema fornece boas estimativas sobre as soluções dos polinômios. Os casos de raízes múltiplas são exemplos simples de instabilidade, pois não se pode confiar completamente nas avaliações do polinômio na região próxima à raiz múltipla[1]. Mesmo sendo pontos de avaliação delicada, existem formas de determinar soluções para as raízes múltiplas. Estes pontos frágeis do sistema poderiam ser contornados por meio da implementação das rotinas que operam sobre ponto flutuante, em substituição às rotinas fornecidas pelo compilador usado.

- A proposta de fazer uso de imagens mostrou-se muito útil e poderia ser adotada em outros tipos de problemas. Seria necessário evidentemente adaptar o problema de maneira semelhante à transformação feita para o caso dos polinômios, a fim de encontrar uma representação adequadamente clara e expressiva. O sistema, em sua versão final, é um bom exemplo de uma abordagem diferente e mais clara a um problema já estudado, fornecendo-lhe novas perspectivas de tratamento. A capacidade essencial de fornecer uma visão abrangente e geral sobre o problema é bastante valorizada, e pode ser propagada a outras ferramentas.

- O desenvolvimento de sistemas semelhantes ou a construção de uma ferramenta genérica de resolução de equações seria uma tarefa muitíssimo interes-

[1] Existem métodos numéricos que permitem a determinação das raízes e de suas multiplicidades, mas estes métodos não foram implementados por causa da aritmética existente, que não era exata o suficiente para sua realização. Perto de raízes múltiplas sempre houveram problemas de avaliação, pois são locais onde geralmente o valor de $|p(z)|$ varia em demasia e de forma descontínua.

sante, especialmente pela abrangência dos tópicos que teriam de ser trabalhados desde a implementação mais básica, a dos números de ponto flutuante, até a interface. Todos os detalhes teriam de ser estudados, para possibilitar um compromisso entre eficiência e comunicabilidade. Possivelmente o assunto mais interessante seria a discussão de quais as operações deveriam ser fornecidas ao usuário e sob que formas, pois uma maneira diferente de apresentar um problema abre enorme campo para que se construam operações sobre esta nova apresentação.

De uma forma geral, este trabalho oferece uma complementação às formas ditas clássicas de determinação de soluções de equações, ou seja, os algoritmos numéricos propriamente ditos. Atingiu-se com sucesso o objetivo de proporcionar uma maneira de visualização que permita uma compreensão melhor e mais fácil do comportamento dos polinômios que são objetivo de estudo, fazendo com que o usuário tenha acesso a informações mais detalhadas e claras, mesmo a um custo superior aos métodos tradicionais. Apesar do custo mais elevado, por meio das imagens podem ser obtidas informações que não estão disponíveis de outras formas, o que torna-se uma vantagem especialmente quando o usuário está operando com casos próximos do limite de exatidão da máquina, ou seja, casos limite.

A adoção da mesma abordagem para equações não polinomiais pode ser feita, desde que se aceitem algumas restrições de comportamento e desempenho que são advindas do desconhecimento de certas informações sobre tais equações, decorrentes de sua própria natureza não polinomial. Mesmo com tais restrições, a proposta continua válida, e pode ser aplicada aos casos não polinomiais.

Apesar dos problemas causados pela aritmética presente nos compiladores usuais, que é bastante restrita e não permite representações muito longas de números em ponto flutuante, resultados numéricos podem ser obtidos, senão como resultados finais, pelo menos como primeiras estimativas para métodos mais sofisticados de determinação de zeros. Nos casos em que as imagens mostram a pouca confiabilidade de métodos

numéricos de determinação de raízes, podem ser adotadas outras formas alternativas de identificação destes zeros por meio de mecanismos gráficos.

Deve ser salientado o fato de que os conceitos de Computação Gráfica que foram utilizados nesta ferramenta foram extremamente simples, de quase nenhuma sofisticação se considerados dentro do enorme conjunto de técnicas de Computação Gráfica hoje existentes. A partir daí, pode-se começar a pensar no impacto que outras técnicas e conceitos mais elaborados teriam se aplicados sobre problemas cotidianos. Espera-se que esta se torne uma prática corrente nos próximos anos, criando-se um novo caminho para o desenvolvimento de programas, baseado em ferramentas para visualização científica.

ANEXO A-1 UMA FERRAMENTA AUXILIAR

Dentro do desenvolvimento deste trabalho foi construída uma pequena ferramenta que permitiu a geração de casos de teste e polinômios especialmente planejados para colocar à prova as capacidades do sistema. Com o auxílio desta ferramenta foi possível testar especificamente os algoritmos de determinação de raízes, já que a criação manual de casos de teste pode ser bastante exaustiva, e esta tarefa se presta sobremaneira a ser automatizada. Este anexo vai explicar o funcionamento desta ferramenta, e, mais importante, sua forma de utilização.

A-1.1 Do problema

Desde o início da utilização em larga escala de computadores percebeu-se a necessidade de controlar e conhecer os erros inerentes à implementação finita dos números reais utilizada nas máquinas digitais. A partir daí, surgiram especialistas denominados Analistas Numéricos, que têm entre suas obrigações conservar a coerência das computações efetuadas em ambientes científicos. Tais analistas enfrentam muitos problemas em seu cotidiano, geralmente por falta de informação suficiente ou adequada sobre o problema a ser resolvido, exigindo por conseqüência estudos do problema, além de conhecimento da implementação que se propõe a resolvê-lo. Entre os fatores que inibem o desenvolvimento de novo software, podemos determinar dois grupos de empecilhos:

- Fatores intrínsecos ao software dedicado à solução do problema, ou seja, erros de máquina, instabilidade algorítmica, questões sobre desempenho, escolha de algoritmos e formas de implementação, estruturas de dados, memória utilizada, etc.

- Fatores vinculados ao problema, tais como mau condicionamento das soluções, problemas de avaliação em determinadas regiões, desconhecimento das soluções exatas, dificuldade de teste e verificação de soluções, etc.

O segundo item acima constantemente impede que sejam feitos testes exaustivos sobre, por exemplo, software para determinação de raízes de polinômios. Isto acontece porque a construção de conjuntos de casos de teste é demorada e feita de uma forma experimental, manual mesmo. É praticamente inviável construir uma série de polinômios com características semelhantes manualmente. Para tanto, exige-se o conhecimento de quais características do software estão sendo analisadas, tais como sua capacidade de detectar raízes muito próximas, sua exatidão em casos de instabilidade numérica (e para isso torna-se necessária a própria *construção* de casos de instabilidade *conhecida*) e o prévio conhecimento de quais são as raízes que deverão ser encontradas. A elaboração de casos de teste que examinem cada uma destas características constitui-se em uma tarefa exaustiva e trabalhosa, quando feita sem o auxílio de algum tipo de dispositivo automático.

Para solucionar este problema no escopo dos polinômios, podemos fazer uso de uma abordagem inversa à construção manual, que pode ser automatizada com facilidade. Esta abordagem baseia-se em fornecer características desejaveis dos polinômios que pretendemos obter, tais como:

- Número de polinômios a ser gerado em uma família.
- Localização desejada das raízes. Por exemplo, podemos dividir este problema de acordo com as seguintes opções:
  - Raízes fornecidas exatamente, através de um valor dado. Com esta opção, podemos forçar o sistema a construir exatamente os polinômios que desejamos.
  - Regiões que devem conter um número preestabelecido de raízes aleatórias. Desta maneira, podemos concentrar um número conhecido de raízes em

uma determinada região, sabendo de antemão quais os valores que deverão ser achados pelo software sendo testado.

- Multiplicidade desejada de cada raiz.

Com estas capacidades, o usuário deverá ser capaz de descrever quaisquer polinômios que deseje. Em seguida, uma ferramenta interpretará esta descrição e produzirá a família de polinômios desejada, obedecendo as características descritas. Ao final, o usuário obterá os polinômios desejados, bem como as listagens das raízes dos mesmos, para fins de comparação com os valores obtidos pela ferramenta de determinação de raízes.

A preocupação em gerar não polinômios isolados, mas séries de polinômios, ou famílias, vêm do fato de que podem acontecer casos de erro nos testes de um polinômio isolado, obtendo-se resultados satisfatórios através de erros que se cancelam. Por este motivo, deve-se testar várias alternativas de polinômios a serem solucionados, todos compartilhando as características que desejamos testar.

Como característica adicional, os coeficientes que são apresentados pelo programa construtor dos polinômios são os mesmos que serão utilizados por outra ferramenta de determinação de raízes, não existindo qualquer erro de arredondamento na leitura destes coeficientes pela segunda ferramenta. Isto é, se fornecessemos manualmente os coeficientes (calculados à mão, e portanto exatos) a um programa de determinação de raízes, este teria de arredondar os valores lidos para a representação usada em ponto flutuante, por vezes alterando os valores iniciais. Evidentemente, as soluções apresentadas por este programa seriam diferentes dos valores obtidos se fossem usados os coeficientes originais sem erro. Assim, usaremos os coeficientes já alterados fornecidos pelo gerador de polinômios, que serão lidos por uma outra ferramenta, quando então poderemos avaliar quão grande é o erro induzido pelas alterações dos coeficientes, através do exame das raízes originais e das raízes encontradas. Não só os coeficientes serão alterados, mas também as raízes originais sofrerão arredondamento, gerando-se um polinômio que possui como raízes estes valores arredondados. Serão estes valores que deverão ser encontrados posteriormente por um software de determinação de raízes.

A-1.2 Linguagem utilizada

Como o usuário deverá descrever as características de seus polinômios através de arquivos lidos pela ferramenta, uma forma qualquer de gramática ou formato de especificação será necessária para possibilitar a comunicação entre usuário e máquina. Esta interface pode ser bastante simples, pois não é necessária uma grande sofisticação para fazer a especificação de uma família de polinômios.

No intuito de possibilitar uma maneira simples de prover a comunicação entre as partes, definimos uma linguagem composta por comandos bastante simples, que, quando colocados em uma seqüência arbitrariamente complexa, podem descrever quaisquer características desejáveis de um grupo de polinômios. Estes comandos são colocados em um arquivo e posteriormente analisados pela ferramenta.

Desta forma, o arquivo poderá conter as seguintes primitivas ou comandos, em qualquer ordem (*a*, *b*, *c*, e *d* representam números reais e *n* representa um inteiro não negativo):

Size *n*

especifica o número de polinômios a ser gerado para aquela descrição. Se forem encontrados vários comandos Size dentro do arquivo de descrição, somente será considerado o último deles. A inexistência de um comando Size indica que será gerado somente um polinômio.

Zero *a b*

especifica uma raiz complexa, através de seu valor explícito: $a + bi$.

Mult *n*

especifica a multiplicidade a ser usada nas próximas raízes a serem produzidas. Se não houver nenhum comando Mult no arquivo, esta será assumida como um. Um comando Mult tem validade até o próximo Mult.

Box *(a,b) (c,d) n*

este comando define um retângulo no plano complexo, contendo os números *u+vi* que possuem valores de *u* em *(a,b)* e valores de *v* em *(c,d)*. Dentro deste intervalo retangular serão criadas *n* raízes, aleatoriamente. Desta maneira, podemos definir qualquer distribuição de raízes que desejarmos. O uso de parênteses é obrigatório neste comando, e deve-se ter $a < b$ e $c < d$. Não existe possibilidade, neste comando, de especificarmos a multiplicidade das raízes. Esta opção seria bastante simples de ser implementada, usando-se mais um argumento, mas optou-se por usar sempre raízes de multiplicidade um pois este é o caso mais comum de uso deste comando.

Unique

esta diretiva indica que, daí para a frente, as raízes complexas colocadas no polinômio serão colocadas sem que se leve em conta o conjugado destas, ou seja, o conjugado do valor inserido **não** será uma raiz do polinômio. Conseqüentemente, os coeficientes do polinômio terão partes imaginárias não nulas.

Nonunique

esta diretiva indica que os conjugados das raízes complexas inseridas no polinômio daí para a frente serão também raízes do polinômio. Este é o modo default do sistema.

Nota: o uso das palavras que designam cada comando é facultativo. O sistema reconhece apenas a primeira letra de cada palavra, e portanto os comandos podem ser substituídos sem problemas por suas letras iniciais.

A-1.3 Geração de resultados

Para processar a descrição obtida usando-se as regras acima, devemos usar uma ferramenta que produzirá os polinômios desejados. Esta ferramenta chama-se `create` e a ela devem ser fornecidos os nomes dos arquivos de descrição e de resultados. Assim, devemos chamar :

`create` *<arqdesc>* *<arqresult>*

Ao final do processamento do arquivo de entrada, *arqresult* terá sido gerado e conterá as descrições dos polinômios e suas raízes, prontas para utilização em outra ferramenta. Caso existam erros no formato do arquivo, as linhas que contiverem tais erros serão ignoradas e um aviso será fornecido.

A-1.4 Formatos de entrada/saída e exemplos

A fim de exemplificar o uso deste aplicativo, pretendemos construir o polinômio que possui raízes reais 1, 2, 3,..., 10. Desta forma, um dos arquivos que faria a descrição deste polinômio seria:

```
Unique
Zero 1 0
Zero 2 0
Zero 3 0
Zero 4 0
Zero 5 0
Zero 6 0
Zero 7 0
Zero 8 0
Zero 9 0
Zero 10 0
```

Deve ser notada a inclusão da diretiva `Unique`, que faz com que cada uma das raízes fornecidas seja colocada somente uma vez no polinômio. Isto acontece porque o sistema usa o default `Nonunique`, fazendo com que o conjugado dos valores fornecidos seja também colocado no polinômio. Como o conjugado de um valor real é o próprio valor real, em conseqüência o sistema colocaria cada valor fornecido *duas* vezes, gerando

um polinômio com raízes de multiplicidade 2. Para evitar este resultado não desejado, devemos incluir a diretiva `Unique`, forçando o sistema a colocar somente as raízes fornecidas, desprezando seus conjugados.

Depois de descrito o polinômio desejado, executamos o sistema usando este arquivo como entrada, e em seguida teremos o seguinte arquivo como saída:

```
(+1.000000000000e+00, +0.000000000000e+00) * z ^ 10
(-5.500000000000e+01, +0.000000000000e+00) * z ^ 9
(+1.320000000000e+03, +0.000000000000e+00) * z ^ 8
(-1.815000000000e+04, +0.000000000000e+00) * z ^ 7
(+1.577730000000e+05, +0.000000000000e+00) * z ^ 6
(-9.020550000000e+05, +0.000000000000e+00) * z ^ 5
(+3.416930000000e+06, +0.000000000000e+00) * z ^ 4
(-8.409500000000e+06, +0.000000000000e+00) * z ^ 3
(+1.275357600000e+07, +0.000000000000e+00) * z ^ 2
(-1.062864000000e+07, +0.000000000000e+00) * z ^ 1
(+3.628800000000e+06, +0.000000000000e+00) * z ^ 0

+1.000000000000e+00 +0.000000000000e+00
+2.000000000000e+00 +0.000000000000e+00
+3.000000000000e+00 +0.000000000000e+00
+4.000000000000e+00 +0.000000000000e+00
+5.000000000000e+00 +0.000000000000e+00
+6.000000000000e+00 +0.000000000000e+00
+7.000000000000e+00 +0.000000000000e+00
+8.000000000000e+00 +0.000000000000e+00
+9.000000000000e+00 +0.000000000000e+00
+1.000000000000e+01 +0.000000000000e+00
```

O arquivo de saída é composto de duas partes, a saber:

1. A lista dos coeficientes do polinômio. Esta lista pode ser usada posteriormente como fonte para um programa que determine numericamente as raízes do polinômio, e em seguida seus resultados podem ser comparados com a segunda parte do arquivo de saída:

2. A lista das raízes do polinômio gerado.

Usando os coeficientes gerados e as raízes originais do polinômio, torna-se possível descrever praticamente quaisquer polinômios que venhamos a precisar, testando em seguida ferramentas de determinação de raízes com facilidade. Eventualmente, a saída dos coeficientes do polinômio pode ser alterada para determinado formato de uma ou outra ferramenta, de acordo com as necessidades do momento.

Em outro exemplo, geraremos um polinômio que possui quatro raízes reais entre 0 e 2. Além destas, possui também seis raízes complexas (simétricas, ou seja, em pares raiz-conjugado) com partes reais entre 2 e 4 e partes imaginárias entre 2 e 3. O seguinte arquivo expressaria o que desejamos:

```
Unique
Box (0,2) (0,0) 4
Nonunique
Box (2,4) (2,3) 3
```

A diretiva `Unique` faz com que as raízes reais sejam incluídas somente uma vez. Em seguida, as raízes complexas são colocadas, desta vez com seus conjugados por causa da diretiva `Nonunique`. Ao final, o arquivo de resultado é constituído por:

```
(+1.000000000000e+00, +0.000000000000e+00) * z ^ 10
(-2.422900427237e+01, +0.000000000000e+00) * z ^ 9
(+2.757891240964e+02, +0.000000000000e+00) * z ^ 8
(-1.905809114738e+03, +5.684341886081e-14) * z ^ 7
(+8.748375478544e+03, +4.547473508865e-13) * z ^ 6
```

```
(-2.750147079427e+04, +0.000000000000e+00) * z ^ 5
(+5.906210025783e+04, -1.818989403546e-12) * z ^ 4
(-8.391862759447e+04, -1.818989403546e-12) * z ^ 3
(+7.401970656658e+04, -7.275957614183e-12) * z ^ 2
(-3.607275815852e+04, +0.000000000000e+00) * z ^ 1
(+7.320697673770e+03, +0.000000000000e+00) * z ^ 0

+1.936142383114e+00 +0.000000000000e+00
+9.565621870368e-01 +0.000000000000e+00
+7.033847564381e-01 +0.000000000000e+00
+1.308871647953e+00 +0.000000000000e+00
+3.024409725808e+00 +2.202018930671e+00
+3.024409725808e+00 -2.202018930671e+00
+3.879954066072e+00 +2.204082270714e+00
+3.879954066072e+00 -2.204082270714e+00
+2.757657857033e+00 +2.793113939833e+00
+2.757657857033e+00 -2.793113939833e+00
```

O desempenho da ferramenta é bastante bom, pois não deve ser considerado um software de uso intensivo, mas apenas usado esporadicamente para gerar casos de teste para outra ferramenta que, esta sim, precisará de um tempo bem mais longo para determinar as soluções dos testes.

Em um teste de desempenho, foram criados 1000 polinômios de grau 20, em um equipamento SUN Sparc 1+. O tempo de criação destes mil polinômios foi de 193s, portanto um tempo médio de 0.193 s/polinômio. Devemos lembrar que este tempo foi distribuído entre os vários processos que estão sob execução em um sistema operacional compartilhado, como é o caso da máquina usada para o teste.

Devido à implementação de ponto flutuante utilizada, foram constatados problemas na geração de resultados com suficientes decimais. Ou seja, valores para raízes

fornecidos com muitas casas decimais não geram coeficientes polinomiais correspondentes àquelas raízes de forma exata. Assim, a construção de polinômios pode ser insegura quando estamos fornecendo raízes muito acuradamente. Por exemplo, geramos um polinômio correspondente às especificações do arquivo abaixo:

```
Unique
Zero 1  0
Zero 0.1 0
Zero 0.01 0
Zero 0.001 0
Zero 0.0001 0
Zero 0.00001 0
Zero 0.000001 0
```

O polinômio gerado teve os seguintes coeficientes:

```
(+1.000000000000e+00, +0.000000000000e+00) * z ^ 7
(-1.111111000000e+00, +0.000000000000e+00) * z ^ 6
(+1.122333221100e-01, +0.000000000000e+00) * z ^ 5
(-1.123445443211e-03, +0.000000000000e+00) * z ^ 4
(+1.123445443211e-06, +0.000000000000e+00) * z ^ 3
(-1.122333221100e-10, +0.000000000000e+00) * z ^ 2
(+1.111111000000e-15, +0.000000000000e+00) * z ^ 1
(-1.000000000000e-21, +0.000000000000e+00) * z ^ 0
```

Infelizmente, devido aos números fornecidos, existe um arredondamento bastante razoável na geração dos coeficientes, perturbando as raízes. Este polinômio foi usado em um programa de determinação de raízes, fornecendo os seguintes resultados:

```
+1.000000000000e+00 - 3.943141028241e-17
+1.000000000000e-01 + 9.054131576898e-20
```

```
+1.000000000000e-02 + 5.639483980202e-19
+1.000000000000e-03 - 1.239255798597e-18
+1.000000000000e-03 - 1.239255798597e-18
+1.000000000000e-03 - 1.239255798597e-18
+1.000000000000e-03 + 6.800101566195e-20
```

O cancelamento fez com que as raízes encontradas fossem perturbadas, ou mais especificamente neste exemplo, as três raízes de menor magnitude foram deslocadas, localizando-se agora sobre a raiz 0.001. Isso leva à interpretação de uma raiz quádrupla, quando na realidade a intenção do usuário que gerou o polinômio era bastante diferente. Tais erros não são corrigíveis de maneira simples, exigindo-se uma nova implementação das operações de ponto flutuante.

A-1.5 Conclusões

A utilização de uma ferramenta simples e poderosa para a geração de exemplos e casos de teste tornou extremamente mais fácil a depuração e testes de alguns ambientes de determinação de raízes que estão sendo desenvolvidos dentro do âmbito do CPGCC, e seu uso deve ser incentivado e ampliado nos próximos meses.

Especialmente, a facilidade de descrição das características desejadas no polinômio deve ser contada como um fator importante na utilização do software, pois propicia uma melhor interação entre usuário e máquina. Não citada especificamente no texto mas presente nas capacidades do sistema, está a possibilidade de se gerar problemas para uso em ambientes de ensino, criando questões e exemplos que podem ser usados em sala de aula.

BIBLIOGRAFIA

[ACT70] ACTON, F. S. **Numerical Methods that Work**. New York: Harper & Row, 1970.

[BAR89] BARBEAU, E. J. **Polynomials**. Berlin: Springer Verlag, 1989.

[HEI84] HEIZINGER, W.; TROCH, I.; VALENTIM, G. **Praxis nichtlinearer Gleichungen**. Wien: Carl Hanser Verlag, 1984.

[MAR49] MARDEN, M. The Geometry of the Zeros of a Polynomial in a Complex Variable. **Mathematical Surveys**, New York, n.3, 1949.

[MAR66] MARDEN, M. **Geometry of Polynomials**. Providence: American Mathematical Society, 1966.

[PRE88] PRESS, W. H.; FLANNERY, B. P.; TEUKOLSKY, S. A., et al. **Numerical Recipes in C**. Cambridge: Cambridge University Press, 1988.

[RAL81] RALL, L. B. **Automatic differentiation: techniques and applications**. Berlin: Springer-Verlag, 1981. (Lecture Notes in Computer Science, 120).

[USP48] USPENSKY, J. V. **Theory of equations**. Bombay: Tata McGraw-Hill, 1948. 353p.

[WOL91] WOLFRAM, S. **Mathematica: a system for doing mathematics by computer**. 2 ed. Reading: Addison-Wesley, 1991.