CONSIDERAÇÕES SOBRE A METODOLOGIA DE DESENVOLVIMENTO DE SOFTWARE NUMÉRICO

por

Dalcidio Moraes Claudio

Tiaraju Asmuz Divério

RP. nº 93 OUTUBRO/88

Trabalho realizado com o apoio do CNPq e FINEP

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

Av. Osvaldo Aranha, 99
90210 - Porto Alegre - RS - Brasil
Telefone: (0512) 21.8499
Telex: (051) CCUF 2680

Correspondência: UFRGS-CPGCC
Caixa Postal 1501
90001 - Porto Alegre - RS - Brasil

RESUMO

Neste artigo pretende-se descrever um Software Numérico Instrucional em seu processo de desenvolvimento, qualidades e ciclo de vida. É apresentado um sistema para ensino desenvolvido na UFRGS e na PUCRS, incluindo uma metodologia de estudo de equações polinomiais. São apresentados exemplos de resolução de equações pelo software SINAI-16.

PALAVRAS-CHAVE:

Software Numérico; ciclo de vida; Metodologia de desenvolvimento, Equações polinomiais; SINAI-16, Fichas Instrucionais; Zeros de polinômios.

ABSTRACT

This paper intends to describe a Numerical Instructional Software on its development process, its qualities and life cycle. We present an educational system which has been developed at UFRGS and PUCRS, including a methodology for study of Polynomial Equations. We also present examples of these equations by the SINAI-16 system.

KEY WORDS

Numerical software; life cycle; development methodology; Polynomial equations; SINAI-16, Instructional cards; Zeros of polynomial.

SUMARIO

LISTA DE FIGURAS

1. INTRODUÇÃO 01

2. O PROCESSO DE DESENVOLVIMENTO DE UM SISTEMA DE SOFTWARE .05
   2.1 Ciclo de vida 05
   2.2 Qualidade do sistema 06
   2.3 Metodologias de desenvolvimento 07
   2.4 As necessidades identificadas por usuários 10

3. SINAI-16 E SUA FILOSOFIA 12
   3.1 O sistema 12
   3.2 Fichas instrucionais 13
   3.3 Implementação 15
   3.4 Exemplo de utilização 16

4. CONCLUSÃO 25

5. REFERENCIA BIBLIOGRAFICA 26

LISTA DE FIGURAS

Figura 1 Menu de segundo nivel 17

Figura 2 Gráfico da cota de Cauchy 17

Figura 3 Gráfico do polinômio P(x) 18

Figura 4 Menu dos métodos de Resolução de raizes 19

Figura 5 Resolução de P(x) pelo método de Newton, valor inicial Xo=3 19

Figura 6 Cálculo das demais raizes de P(x) por Newton ...20

Figura 7 Tela de definição do P(x) (exemplo 2) 20

Figura 8 Gráfico da cota de Cauchy (exemplo 2) 21

Figura 9 Gráfico do polinômio P(x) (exemplo 2) 22

Figura 10 Cálculo das raizes reais por Newton (exemplo 2) ..22

Figura 11 Cálculo da raiz real pelo método de Newton (exemplo 2) 23

Figura 12 Cálculo das raizes complexas por Bairstow 24

Figura 13 Deflação das raizes reais 24

1 INTRODUÇÃO

Usualmente os Softwares Numéricos podem ser agrupados sob as seguintes formas: **Programas Individuais, Conjuntos de Programas ou Pacotes, Bibliotecas de Software e Sistemas de Software.**

Os **programas individuais** são pequenos programas, normalmente desenvolvidos para solucionar um problema especifico. Muitos destes programas têm um ciclo de vida temporário (descartável), como é o caso dos programas desenvolvidos por alunos. Depois de especificados pelo professor, os alunos projetam os algoritmos; os codificam em uma linguagem de programação e corrigem os erros de sintaxe (compilação) e erros lógicos (verificação se o programa realiza aquilo para o qual foi planejado). Após sua execução e entrega para ser avaliado, o programa é, em geral, descartado.

Uma fase natural que se seguiu a essa foi a formação de **conjunto de programas** sobre uma determinada área, originando pacotes de softwares, os quais se desenvolveram e passaram a exigir um pouco mais em termos de qualidade, como por exemplo: robustez, confiabilidade (bem testado), boa documentação e consistência. Em um pacote, algumas convenções devem ser tomadas para facilitar o uso, a documentação e a manutenção do software.

Houveram grupos mais ambiciosos como os que testaram os programas individuais, os documentaram e os organizaram em **bibliotecas de software**, as quais evitaram que se fizessem esforços para desenvolver e testar rotinas que já haviam sido desenvolvidas.

No princípio, nem tudo funcionou como seria esperado pois as primeiras bibliotecas tiveram problemas de não serem flexíveis quanto ao uso e de não serem adequadas à realidade, devido principalmente à inexperiência dos programadores que as desenvolveram. Essas bibliotecas foram compostas por programas desenvolvidos individualmente, sem um projeto de padronização ou verificação.

Após um grande esforço, surgiram bibliotecas confiáveis tais como: IMSL (International Mathematical and Statistical Libraries), SPSS (Statistical Package for the Social Sciences) e NAG(Numerical Algorithms Group), que foram desenvolvidas através de grandes projetos que custaram milhões de dólares, alguns anos de desenvolvimento, envolveram vários doutores, mestres e técnicos e com a cooperação de Universidades, Laboratórios Governamentais e Empresas Privadas.

Para facilitar a utilização dos pacotes e das bibliotecas têm surgido os **Sistemas de Software.** São na realidade gerenciadores e facilitadores das bibliotecas e dos pacotes. Alguns sistemas de software podem ser entidades especialistas que

auxiliam a determinação do melhor método para resolução de um determinado problema, ensinam a utilizá-lo e facilitam seu acesso. Outros se caracterizam por proporcionar um interface entre o usuário e a biblioteca, como é o caso do FORTRAN, que cria uma linguagem pré-processada, a qual está próxima à forma usual de usuário escrever sentenças matemáticas.

Os sistemas de software podem ser orientados por menus ou por comandos. Os sistemas orientados por menus, apresentam aos usuários um conjunto completo de escolhas que eles podem tomar, a cada nivel de atividade. Isto facilita a utilização do sistema, mas o torna mais lento em sua operação, pois para determinada aplicação, o usuário terá que percorrer uma certa quantidade de telas, até que possa executar sua tarefa.

Os menus de tela inicialmente apareciam com as opções associadas a números ou letras, desta forma para utilizarem determinado método, bastava digitar o número ou letra correspondente ao método. Estes menus a cada escolha são refeitos, contendo nova lista de escolha. Os itens de um menu podem ser de três tipos: terminal, intermediário e de retorno.

O item intermediário é uma tela menu que contém outros itens intermediários, terminais ou de retorno. O item terminal é uma tela explicativa mostrando determinado comentário e depois retorna ao nivel anterior, ou um item que ativa um método, proporcionando sua execução e depois retornando ao nivel anterior. Os itens de retorno são as opções que possibilitam o retorno ao menu anterior.

Atualmente existem duas inovações nesta técnica de menus. A primeira é a **pilha de janela** ou **sobreposição de janelas.** As telas menus são empilhadas, umas sobre as outras, através de janelas que são abertas, aparecendo no monitor toda a sequência de telas e opções, o que possibilita ao usuário o "overhead" da execução. A outra inovação é a substituição das letras e números associados aos itens por **botões**, que são regiões da tela delimitadas e, acionáveis pelo dispositivo denominado **rato (mouse)**, um dispositivo gráfico.

Os sistemas orientados por comandos são mais rápidos, mas exigem um maior conhecimento do sistema, para que se possa saber que comandos estão disponíveis em determinado nível de atividade. São necessários sistemas de ajuda para orientar o usuário e ensiná-lo como usar determinado comando.

2 PROCESSO DE DESENVOLVIMENTO DE UM SISTEMA DE SOFTWARE

O processo convencional de desenvolvimento de um sistema de software é composto por três etapas características: a escolha do método básico a ser usado; o projeto do algoritmo e a produção do programa como um item do software. As duas últimas etapas são subdivididas no projeto e análise do algoritmo; construção do software; documentação detalhada; vários testes e distribuição e manutenção do software (como descrito em [CLA87]).

Estas subdivisões estão diretamente relacionadas com as etapas do ciclo de vida convencional de um sistema de software de vida longa ou permanente. Estas etapas são descritas por J.R.Rice (em [RIC85]) e são: Especificação, Projeto, Codificação, Testes, Integração ao sistema, Documentação e Manutenção.

2.1 Ciclo de vida

Na etapa da **Especificação** as funções gerais do sistema são analisadas e transformadas num conjunto concreto de especificações do software. São definidos precisamente os dados de entrada e saída.

No **Projeto**, a estrutura geral do sistema é desenvolvida com um formato especificado para arquivos, representação de dados, etc. Os algoritmos são identificados para serem implementados. O nível de detalhes pode ir até o fluxo de dados de cada método planejado no sistema.

Na **Codificação** é escrito um código preliminar para os programas e realizada uma depuração inicial, a qual é bastante

trabalhosa de ser feita. Os erros de programação são corrigidos na codificação e testes são feitos para verificar se o desempenho está de acordo com a especificação e projeto.

Na **Integração do sistema**, os vários programas são agrupados para compor o sistema, sendo então necessário testar a performance do conjunto e sua confiabilidade.

A **Documentação** é uma atividade que, na realidade, faz parte de cada etapa, pois deve conter informações de cada uma delas: desenvolvimento das especificações, o projeto e a implementação do software. As informações podem ser para pessoas que estão desenvolvendo o software, ou para pessoas que vão realizar a manutenção do software, ou ainda, para os usuários do software.

A **Manutenção do software** é a etapa mais longa da vida de um sistema. Nela são desenvolvidas atividades de correção dos problemas encontrados na especificação ou na implementação, são ainda incorporadas novas facilidades aos sistema, produzindo novas versões, ou adaptando o sistema as novas exigências.

2.2 Qualidade do sistema

A **qualidade** de um sistema de software depende da qualidade das especificações do projeto, implementação, verificação. As especificações são a base do desenvolvimento futuro do software. A qualidade deve ser construída em cada fase do ciclo de desenvolvimento, pois não se pode adicionar qualidade em etapas posteriores e por um pessoal que não

participou do desenvolvimento, sendo, portanto, de responsabilidade de todos os que participam na sua realização ou seu uso.

Para que seja construido um software com a qualidade desejada e se possa medir se o software produto possui essa qualidade, é necessário que se defina a qualidade de software de acordo com seu contexto. Para isto, deve-se analisar a natureza do produto e identificar quais as características consideradas desejáveis. Estas características devem ser possíveis de serem avaliadas, existindo um grau mínimo a ser atingido para que o software seja considerado de boa qualidade. (Estas características e como avaliá-las podem ser encontradas em [BEA87] e em [RIC85]).

A qualidade dos requisitos definidos no início do processo de desenvolvimento é fator básico para que possa ser produzido um software de boa qualidade.

A qualidade é definida em função do usuário. Por exemplo, um software de ensino não exige a mesma qualidade que um software para o projeto espacial da NASA, onde no caso do cálculo do ângulo de entrada na atmosfera terrestre para a nave APOLO faz-se necessário até o uso da Teoria dos Intervalos.

2.3 Metodologias de desenvolvimento

As metodologias mais conhecidas de desenvolvimento são as chamadas "Top down" e "Bottom Up". A metodologia "Bottom up" consiste em desenvolver o sistema na ordem inversa da estrutura,

ou seja, desenvolvem-se os programas, funções e subrotinas e então, vai se agrupando em unidades funcionais ou módulos cada vez maiores, até que se chegue num módulo capaz de resolver todas as especificações do sistema.

A metodologia "Bottom Up" aplicada ao desenvolvimento de softwares numéricos envolve um estudo dos métodos, uma análise matemática e computacional das dificuldades visando transformar o método num programa executável. Identificam-se módulos comuns que são agrupados num mesmo módulo maior. A construção de um software com esta metodologia é semelhante a pensar em caminhar numa árvore das folhas para a raiz.

A metodologia **"Top down"** é a mais utilizada por ser mais natural. Ela consiste em, partindo da especificação do sistema, derivá-la em módulos de acordo com o número de detalhes denominada refinamentos sucessivos.

Nesta metodologia, quando aplicada ao desenvolvimento de softwares numéricos, primeiro se define a estrutura geral, a filosofia e a amplitude do software e então, inicia-se a especificação dos módulos ou pacotes que serão desenvolvidos, a gerência e o inter-relacionamento destes, definindo-se as variáveis globais e de controle do sistema; os dados de entrada e saída de cada módulo e como se dará a passagem de parâmetros. É preciso que se convencione a estrutura das mensagens de erro, para que elas sejam claras e adequadas, identificando a situação anormal, a gravidade e o que realizar para contornar tal situação.

A estratégia Top-down é a melhor estratégia para se controlar a complexidade do desenvolvimento do projeto. Em geral, o desenvolvimento de software dificilmente se dá puramente Top down, pois ao longo do processo de desenvolvimento há necessidade de modificações e estas exigem o uso da estratégia Bottom up. Portanto, no desenvolvimento de softwares numéricos é aconselhável a utilização das duas metodologias, pois elas se complementam e, com elas temos a visão do todo e de cada parte, tendo-se desta forma uma idéia do trabalho computacional para desenvolver o sistema. Isto pode auxiliar também a solução do problema de determinação do número de pessoas que devem trabalhar no desenvolvimento do sistema.

Uma técnica que facilita o desenvolvimento de um projeto é a **prototipagem rápida**, ou **versões sucessivas** que consiste em desenvolver um protótipo do sistema e cada vez mais incorporar a ele novas funções e facilidades, gerando novas versões. Esta técnica possibilita maior interação entre a equipe de desenvolvimento e o usuário final, possibilitando que erros de especificação sejam identificados com antecédencia.

As vantagens da técnica de versões sucessivas é que ela propicia aos usuários uma resposta mais rápida às suas necessidades mais prementes e tem-se a evolução do sistema mais condizente com a necessidade e a capacidade de absorção real do usuário. A desvantagem é que este processo, em geral, é bastante dispendioso porque pode ser necessário que uma parte substancial do código seja escrito várias vezes para que esta se adapte aos novos requerimentos.

Protótipos representam pesquisas recentes e não foi ainda conseguida a demonstração formal de que o esforço dispendido para realizá-los é compensado em termos de custos de desenvolvimento. Uma "análise realista" desta técnica é feita por M.R.S.Borges em [BOR87].

As filosofias do software também influenciam seu desenvolvimento, pois dependendo a quem ele se destina, teremos as suas especificações. Algumas filosofias são: Instrucional (voltada ao ensino); Aplicativa Genérica (poucos métodos de cada assunto, mas uma grande quantidade de assuntos) e Aplicativa Especialista (especifica para um determinado assunto, tendo uma grande quantidade de métodos eficientes).

2.4 As necessidades identificadas por usuários

Para fins deste trabalho, classificamos os usuários de softwares matemáticos em basicamente três grupos que são: alunos, professores e pesquisadores. Para cada um destes usuários tentaremos identificar algumas de suas necessidades.

Os alunos necessitam de softwares que lhes auxiliem na resolução de seus exercícios e que complementem as explicações do professor. Ele são usuários que não exigem muita exatidão e nem se preocupam com velocidade de processamento ou convergência. Para estes, tela-menus contendo explicações, enumerando as possíveis escolhas e tutoriando os movimentos, facilitam o uso do sistema. Eles são exigentes quanto a simplicidade e facilidade de uso, pois caso o sistema imponha restrições, ou sejam necessários

maiores conhecimentos para a entrada de dados iniciais ou passagem de parâmetros, ficam desincentivados de utilizar o sistema.

Os professores são usuários de nível médio de softwares matemáticos, pois em geral têm maiores conhecimentos de computação e se utilizam dos softwares para elaboração de exemplos e exercícios para suas aulas. Muitas vezes, softwares instrucionais são utilizados como um instrumento para correção de provas. Para isto são necessárias facilidades de determinação do número de casas decimais e de acesso a valores parciais. Estes usuários também costumam realizar comparações entre métodos e pesquisas, conseqüentemente são necessários métodos além dos tradicionais, que sejam mais eficientes.

Por usuários pesquisadores entende-se profissionais como engenheiros, que tenham problemas reais que necessitam ser resolvidos com rapidez e alta exatidão. Em geral não requerem métodos elementares, mas métodos eficientes. Em alguns casos especiais, exigem métodos mais elaborados portanto, o sistema tem que ter disponível estes métodos. Um usuário com tal qualificação pode não gostar de telas de menus, por ser obrigado a passar por um grande conjunto de telas para então executar e resolver seu problema, sendo mais aconselhável que o gerenciamento seja feito via comandos de uma linguagem pré-processada. O porte dos problemas é maior, o que necessita a possibilidade de leitura e gravação de dados em arquivos em disco e um melhor aproveitamento da memória.

3 SINAI-16 E SUA FILOSOFIA

3.1 O Sistema

O SINAI-16 (Software Interativo Numérico Aplicativo e Instrucional em micros de 16 bits) é um conjunto de pacotes de software voltado ao ensino de matemática computacional. Sua especificação e projeto fizeram parte do trabalho de mestrado [DIV 86]. Seu desenvolvimento se deu no Laboratório de Programação do Instituto de Informática da PUC RS.

Na sua primeira fase, foi desenvolvido o pacote para resolução de Equações Polinomiais, o qual é capaz de determinar raízes reais e complexas de polinômios de grau maior ou igual a vinte, (este valor foi convencionado pelas limitações de memória do equipamento e por ser um pacote instrucional). Ele permite fazer gráficos em diferentes escalas, calcular o valor de cotas que determinam o intervalo que contém todas as raízes do polinômio e é capaz de deflacionar raízes reais e complexas, rebaixando o grau do polinômio com razoável exatidão.

Para este módulo inicial foram desenvolvidos dois manuais, o manual instrucional do usuário e o manual de manutenção do programa [DIV86a] e [DIV86b]. No manual instrucional estão as informações referentes a instalação e a utilização. Para cada um dos métodos foi desenvolvida uma ficha instrucional composta por um conjunto completo de informações, tanto matemáticas como computacionais. Nelas existem ainda exemplos de execução e uma lista de referências bibliográficas sobre o método em questão.

O pacote foi avaliado por especialistas em educação, no que diz respeito a aspectos instrucionais, como por exemplo as telas-menus que facilitam o uso, as telas explicativas claras e concisas e perguntas claras que possibilitam uma conversação do usuário com a máquina, gerando um estilo conversacional no sistema.

Este pacote foi utilizado como uma ferramenta de auxílio no ensino de Cálculo Numérico durante vários semestres. O parecer dos alunos foi muito satisfatório e ficou comprovado o êxito deste tipo de instrumento no ensino.

Quanto a aplicações, o pacote foi solicitado para solucionar alguns problemas associados à engenharia, os quais foram resolvidos com uma exatidão de cerca de 7 a 8 algarismos significativos corretos. Atualmente este pacote foi cedido a cerca de dez centros universitários no país.

3.2 Fichas Instrucionais

O SINAI-16 introduz uma nova metodologia de ensino, a qual poderá se tornar uma forma de apresentar novos métodos e possibilitar uma catalogação dos métodos existentes. Esta metodologia consiste de uma ficha instrucional específica para cada classe de problema.

A fim de exemplificar a metodologia, apresentamos a ficha instrucional para métodos de resolução de equações algébricas e transcendentes. Ela é composta de vários itens que,

no seu conjunto, constituem todas as informações necessárias para as pessoas que queiram utilizar os métodos de resolução de equações, quer com ênfase prática de utilização do método, ou para quem quiser implementar outros métodos, ou ainda, considerações de ênfase computacional.

Os itens que constituem a ficha instrucional de um método de resolução de equações são:

i) Nome do método;

ii) Classificação do método;

iii) Objetivo do método;

iv) Descrição do método;

v) interpretação geométrica;

vi) Limitações e restrições;

vii) Erro do método;

viii) Ordem de convergência;

ix) Indice de eficiência;

x) Algoritmo;

xi) Implementações disponíveis;

xii) Listagem do programa;

xiii) Mensagem de erro;

xiv) Dados de entrada e saída;

xv) Exemplos de utilização do programa;

xvi) Referência bibliográfica;

xvii) Observações.

No módulo inicial, desenvolvido para o pacote de resolução de equações polinomiais, foram catalogados em fichas

instrucionais e implementados novo métodos, que são: Bissecção, Newton-Raphson; Secante; Newton para raizes múltiplas; Hibrido C; Midrem; Graeffe; Newton para raizes complexas e o Bairstow.

Na verdade, foram identificados 52 métodos existentes na literatura, para os quais estão sendo desenvolvidas as fichas para uma posterior implementação e incorporação ao pacote.

3.3 Implementação

A especificação e o projeto do pacote de resolução de equações foram feitos durante o primeiro semestre de 1985. Foi feito um levantamento e análise dos recursos computacionais existentes no Laboratório de Programação, sendo escolhido um equipamento compatível ao IBM PC.

Nesta ocasião, a única linguagem de programação com recursos gráficos disponível e que continha facilidades para definição de funções era a versão 1.14 do Basic da COMPAQ' Personal Computer. Portanto, este módulo foi desenvolvido em Basic, ocupando cerca de 34 Kbytes de memória.

Atualmente, o pacote de resolução de equações está sendo refeito em PASCAL TURBO. Pretende-se incorporar a ele outros métodos, os quais já foram catalogados. Cada método está organizado em um módulo independente, gerenciado pelas telas de menu.

Pretende-se, ainda, incorporar a este módulo um interpretador de funções para ampliar a classe de problemas que ele é capaz de resolver.

Para cada dependência de máquina foram criadas variáveis, e a elas, atribuídos valores para facilitar a transportabilidade do sistema. Estas variáveis estão identificadas por comentários no início do programa.
Algumas delas são:

menor - representa o menor número de máquina positivo que o sistema de ponto-flutuante permite representar;

graumax - maior grau de polinômio que pode ser resolvido;

ascmax - maior exatidão possível de se obter; este valor é determinado em função da precisão do computador.

3.4 Exemplo de utilização

Para ilustrar a potencialidade do SINAI-16 na resolução de equações e de como poderá ser utilizado como ferramenta no processo ensino-aprendizagem, vamos mostrar a resolução de uma equação polinomial,

$$P(x) = x^4 + 2x^3 - 7.5x^2 - 20x - 11$$

dispensaremos algumas etapas, como por exemplo, as entradas de dados e algumas telas menus contendo as escolhas de itens.

Uma vez introduzido os coeficientes do polinômio P(x), temos a tela-chave de segundo nível contendo o menu mostrado na figura 1.

```
Resolução de Equações Polinomiais

    1. Definição da função
    2. Gráfico da função
    3. Cálculo das cotas
    4. Cálculo das raizes reais
    5. Cálculo das raizes complexas
    6. Deflação das raizes
    7. Voltar a tela-chave de primeiro nivel
```

Figura 1. Menu de segundo nivel - SINAI'16

Quando não se tem nenhuma informação sobre o polinômio P(x), uma primeira tarefa pode ser o esboço gráfico do polinômio. Para isto é necessário determinar o intervalo que contém todas as raizes, o que pode ser calculado pela cota de Cauchy, no item 3 deste menu.

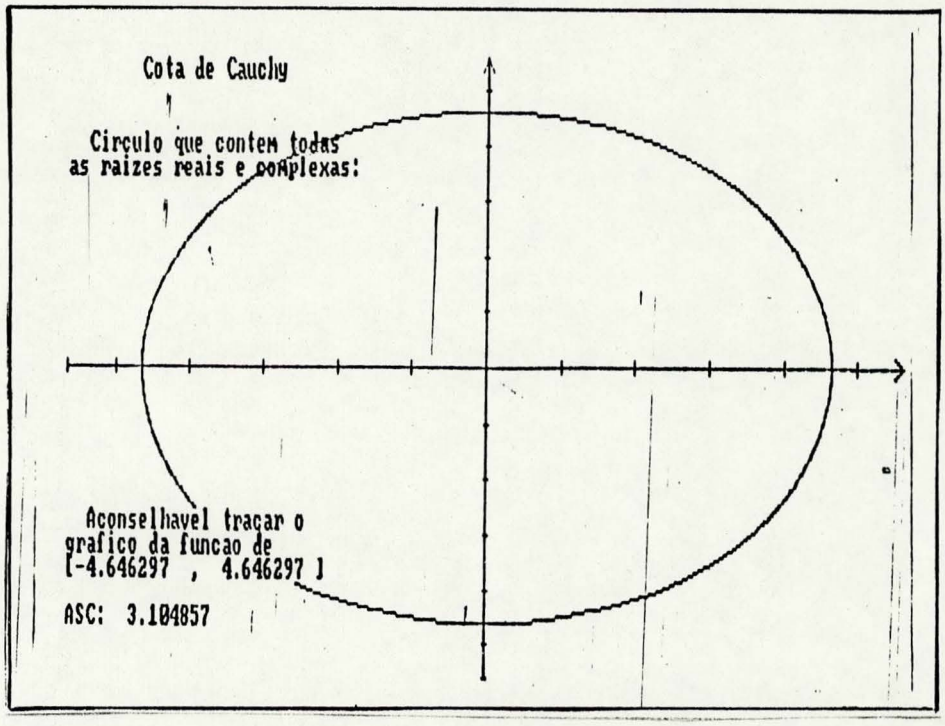


Figura 2. Gráfico da cota de Cauchy

Como pode ser visto na figura acima, temos um círculo centrado na origem, de raio 4.6463 que contém todas as raizes reais e complexas. Pode-se, então, traçar o esboço gráfico no item dois da tela chave de segundo nivel, com a escala no eixo X de -5 a 5 e no eixo Y, de -12 a 12 (poderia ser também de -5 a 5, mas para conter o ponto onde a curva cruza o eixo Y, estamos usando a escala em relação ao termo independente a0 = -11).

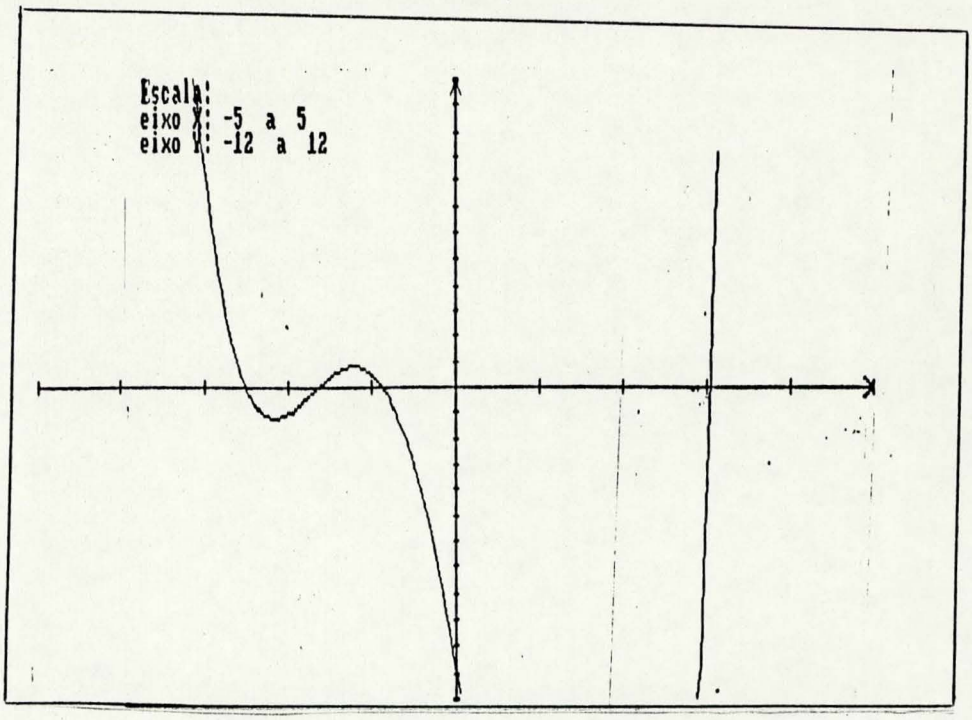


Figura 3. Gráfico da função

Com o gráfico, pode-se observar que a função cruza em quatro pontos o eixo X, de onde se conclui que existem quatro raizes reais, que podem ser resolvidas pelos métodos enumerados no menu mostrado na figura 4.

```
Cálculo de raizes reais

   A) Método da Bissecção
   B) Método de Newton
   C) Método da Secante
   D) Método de Newton para raizes múltiplas
   E) Método Hibrido C
   F) Método MIDREM
   G) Método de Graeffe
   H) Voltar a tela-chave de segundo nivel
```

Figura 4. Menu dos métodos de resolução de raizes

Usando o método de Newton podemos calcular cada uma das raizes. Podemos ter o resultado com valores parciais, como no caso da raiz calculada com o valor inical x0 = 3.0000.

K	Xk	F(Xk)	F'(Xk)	ASC
0	-3.0000000	+8.5000000	-29.0000000	0.0
1	-2.7068966	+2.2040016	-14.7696673	0.7
2	-2.5576717	+0.4214735	-9.3107479	0.9
3	-2.5124043	+0.0328565	-7.8758255	1.4
4	-2.5082325	+0.0002663	-7.7485987	2.5
5	-2.5081981	-0.0000014	-7.7475539	4.6
6	-2.5081983	-0.0000016	-7.7475627	6.8
7	-2.5081985	+0.0000025	-7.7475710	6.8
8	-2.5081982	-0.0000040	-7.7475644	6.6
9	-2.5081987	+0.0000025	-7.7475796	6.4
10	-2.5081984	-0.0000001	-7.7475616	6.6
11	-2.5081984	+0.0000002	-7.7475613	7.8

N' de iterações: 11
Raiz: -2.508198367103168
ASC: 7.806963

Fazer outro calculo? (S/N)

Figura 5. Resolução de P(x) pelo método de Newton, valor inicial Xo=3

Podemos ainda optar, apenas, pelo resultado final, como no cálculo das demais raizes de P(x), onde aparece o valor inicial, o número de iterações, a exatidão e a aproximação da raiz.

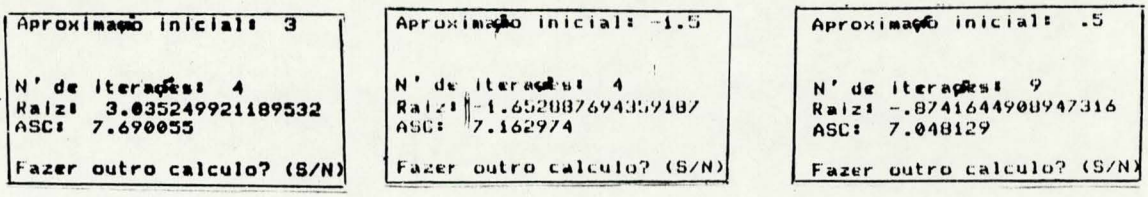


Figura 6. Cálculo das demais raizes de P(x) por Newton

Caso o polinômio tenha raizes complexas, estas podem ser calculadas pelos métodos de Newton para complexas ou por Bairstow, ou se pode deflacionar as raizes reais já calculadas rebaixando o grau do polinômio para o segundo e, então, por Baskara calcular o par complexo. Um exemplo disto está no calculo das raizes do polinômio:

$$P(x) = x^4 + 2x^3 - 7.5x^2 - 11x - 20$$

que, na verdade apenas troca os coeficientes a0 e a1 do polinômio anterior. Optando pelo item 1, definição da função, podemos alterar apenas estes coeficientes e temos então a função definida e mostrada pela tela:

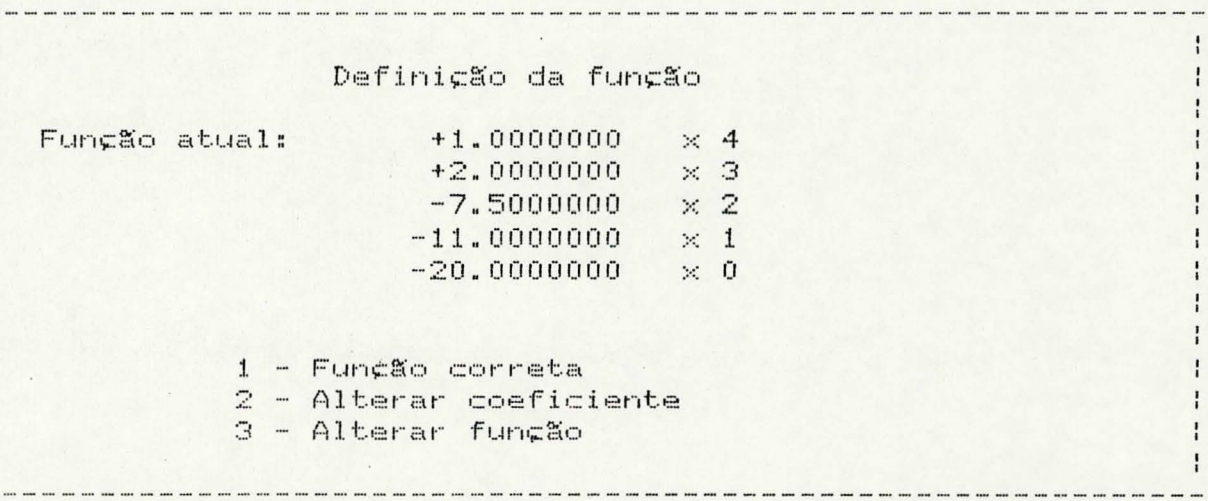


Figura 7. Tela de definição do P(x) (exemplo 2)

Novamente calcularemos a cota de Cauchy para determinar o intervalo que contém as raizes, o qual é de -4.5 a +4.5, como se pode notar no gráfico a baixo. Observa-se, ainda, que esta estimativa tem três algarismos significativos corretos, pois asc = 3.111.

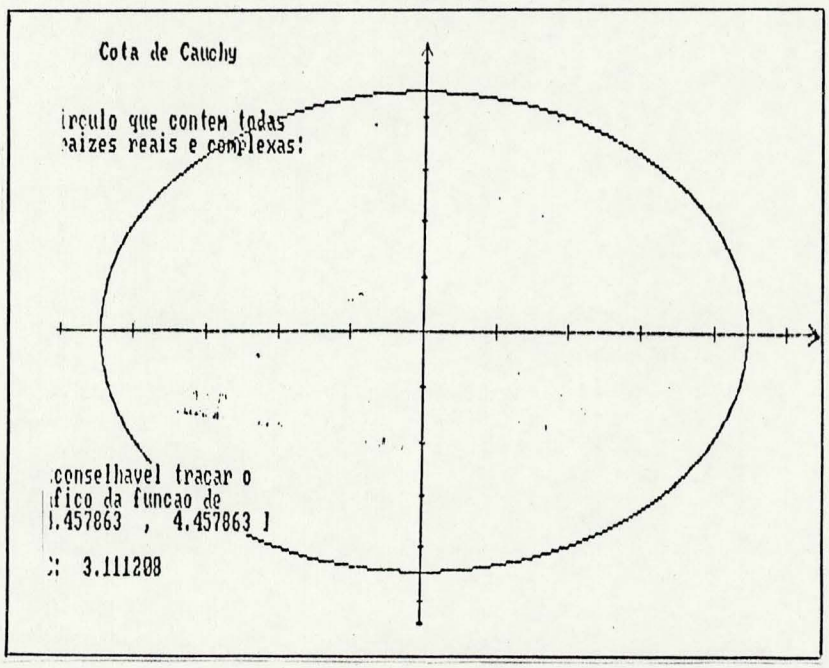


Figura 8. Gráfico da cota de Cauchy (exemplo 2)

Trançando o gráfico na escala do eixo X de -4 a 4 e na escala do eixo Y de -50 a 30, a fim de termos um esboço continuo da função no intervalo citado.

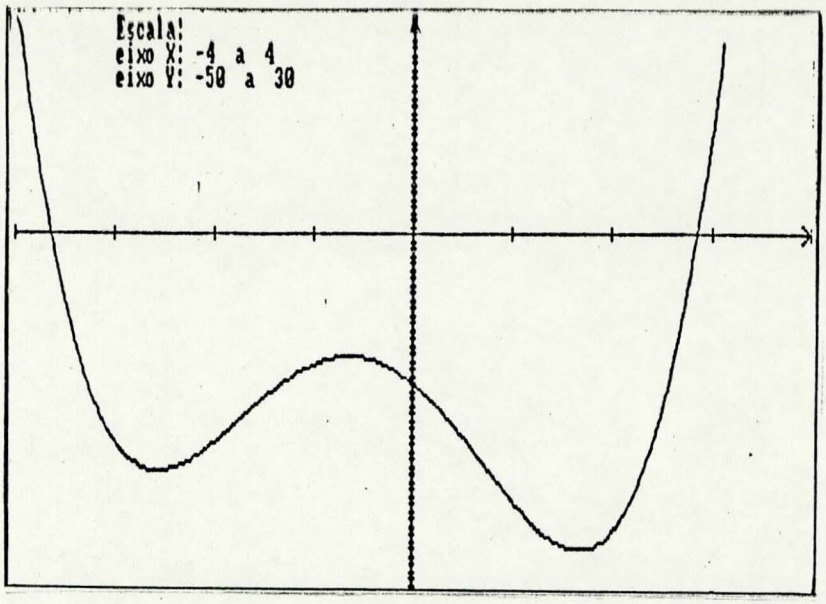


Figura 9. Gráfico do polinômio P(x) (exemplo 2)

Observando o gráfico identificamos duas raizes reais, uma no intervalo [-4 , -3] e a outra entre [2 , 3]. Observa-se ainda que existe um máximo local negativo próximo de x = -1.0, o que determina a existência de raizes complexas.

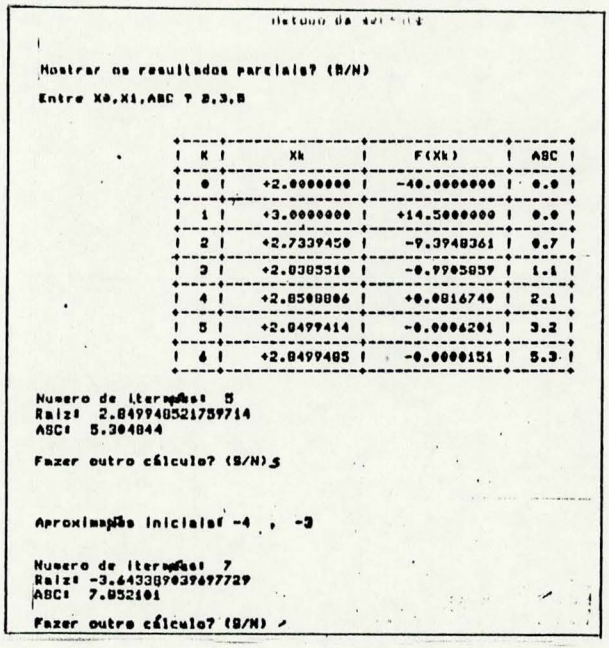
Calcularemos, inicialmente, as raizes reais pelo método de Newton com os valores iniciais x0= -4 e depois com x0= 3. As tabelas abaixo mostram o cálculo com valores parciais.

```
                    Metodo de Newton

Mostrar os resultados parciais? (S/N)
Entre X0, ASC: ? 3.6

+----+--------------+--------------+--------------+------+
| K  |      Xk      |     F(Xk)    |    F'(Xk)    | ASC  |
+----+--------------+--------------+--------------+------+
| 0  |   +3.0000000 |  +14.5000000 |  +106.0000000 | 0.0  |
+----+--------------+--------------+--------------+------+
| 1  |   +2.0632075 |   +1.1714612 |   +89.1294625 | 1.0  |
+----+--------------+--------------+--------------+------+
| 2  |   +2.0500642 |   +0.0101415 |   +87.5890831 | 2.0  |
+----+--------------+--------------+--------------+------+
| 3  |   +2.0499484 |   -0.0000137 |   +87.5754811 | 4.1  |
+----+--------------+--------------+--------------+------+
| 4  |   +2.0499486 |   +0.0000151 |   +87.5755093 | 7.0  |
+----+--------------+--------------+--------------+------+

N' de iteracoes: 4
Raiz: 2.049948552021973
ASC: 4.952345

Fazer outro calculo? (S/N)N
```

```
                    Metodo de Newton

Mostrar os resultados parciais? (S/N)
Entre X0, ASC: ? -4,7

+----+--------------+--------------+--------------+------+
| K  |      Xk      |     F(Xk)    |    F'(Xk)    | ASC  |
+----+--------------+--------------+--------------+------+
| 0  |   -4.0000000 |  +32.0000000 |  -111.0000000 | 0.0  |
+----+--------------+--------------+--------------+------+
| 1  |   -3.7117117 |   +5.0320606 |   -77.2055881 | 0.8  |
+----+--------------+--------------+--------------+------+
| 2  |   -3.6465343 |   +0.2211607 |   -70.4736544 | 1.4  |
+----+--------------+--------------+--------------+------+
| 3  |   -3.6433961 |   +0.0005061 |   -70.1576981 | 2.8  |
+----+--------------+--------------+--------------+------+
| 4  |   -3.6433889 |   -0.0000005 |   -70.1569747 | 5.4  |
+----+--------------+--------------+--------------+------+
| 5  |   -3.6433889 |   -0.0000004 |   -70.1569746 | 8.2  |
+----+--------------+--------------+--------------+------+

N' de iteracoes: 5
Raiz: -3.643388853232132
ASC: 8.245276

Fazer outro calculo? (S/N) S
```

Figura 10. Cálculo das raizes reais por Newton (exemplo 2)

Podemos também calcular os valores das raizes pelo método da Secante nos intervalos dados anteriormente. É dada a tabela de valores parciais para o cálculo da primeira raiz e para a segunda foi dado apenas o valor final da aproximação da raiz.



```
                    Método da Secante

Mostrar os resultados parciais? (S/N)
Entre X0,X1,ABC ? 2,3,5

+----+----------------+----------------+-------+
| K  |       Xk       |     F(Xk)      |  ABC  |
+----+----------------+----------------+-------+
|  0 |   +2.0000000   |   -40.0000000  |  0.0  |
+----+----------------+----------------+-------+
|  1 |   +3.0000000   |   +14.5000000  |  0.0  |
+----+----------------+----------------+-------+
|  2 |   +2.7339450   |    -9.3948361  |  0.7  |
+----+----------------+----------------+-------+
|  3 |   +2.8385510   |    -0.7905857  |  1.1  |
+----+----------------+----------------+-------+
|  4 |   +2.8508806   |    +0.0816740  |  2.1  |
+----+----------------+----------------+-------+
|  5 |   +2.8499414   |    -0.0006201  |  3.2  |
+----+----------------+----------------+-------+
|  6 |   +2.8499485   |    -0.0000151  |  5.3  |
+----+----------------+----------------+-------+

Numero de iterações: 5
Raiz: 2.8499480521759714
ABC: 5.304844

Fazer outro cálculo? (S/N) S


Aproximações iniciais? -4 , -3

Numero de iterações: 7
Raiz: -3.6433899039697729
ABC: 7.852101

Fazer outro cálculo? (S/N) N
```

Figura 11. Cálculo da raiz real pelo método da Secante

Comparando as soluções produzidas pelos dois métodos pode-se verificar que o método de Newton com menos iterações conseguiu melhor exatidão do que o método da Secante.

Para o cálculo das raizes complexas temos algumas alternativas. Uma delas é calcular pelo método de Newton para raizes complexas ou por Bairstow. Calcularemos, para ilustrar, por Bairstow, usando a estimativa inicial z0 = -1 -i, temos os seguintes valores.

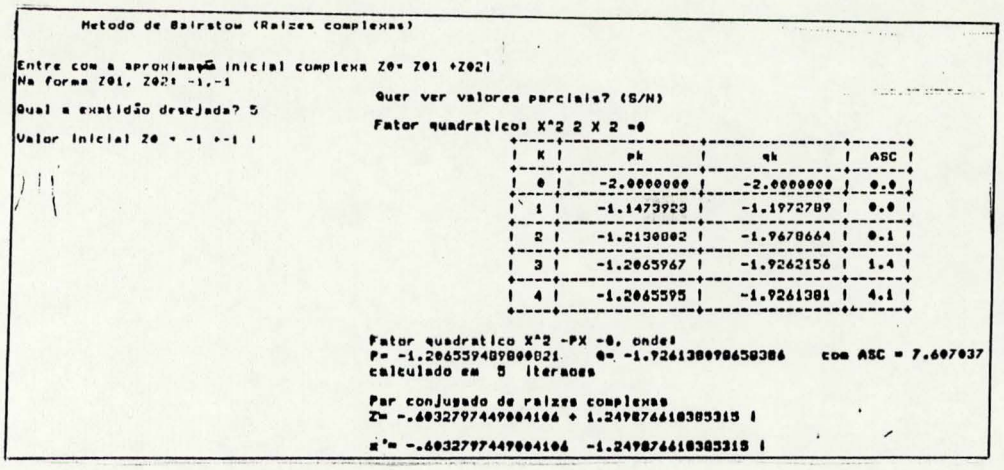


Figura 12. Cálculo das raizes complexas por Bairstow

Outra alternativa, é deflacionar as raizes calculadas por Newton (ou Secante) e depois aplicar a fórmula de Baskara (existente na opção de Bairstow, quando grau do polinômio n=2), no polinômio reduzido de segundo grau para determinar o par complexo.

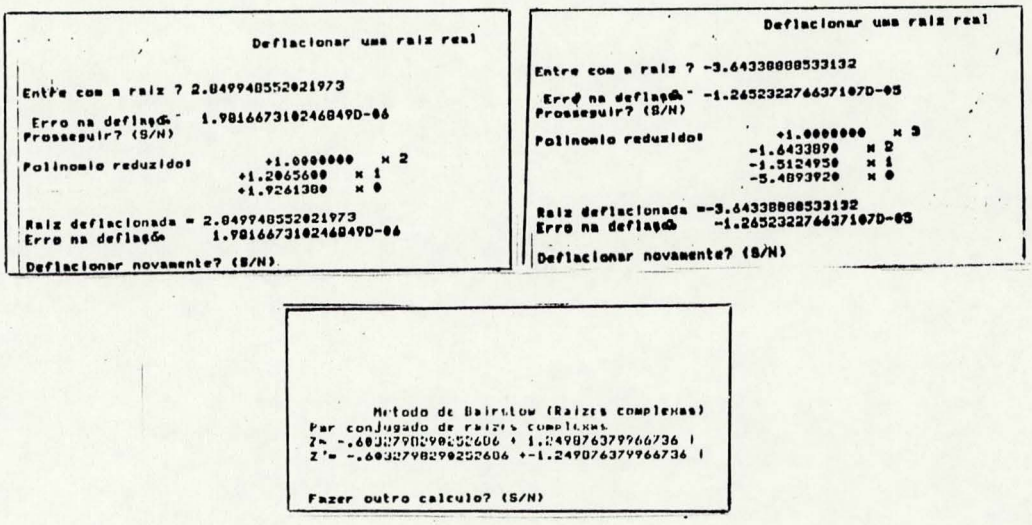


Figura 13. Deflação das raizes reais

Observa-se que apenas cinco algarismos coincidem, nos resultados obtidos pelas duas alternativas.

4 CONCLUSÃO

Este trabalho apresenta contribuições para orientar o desenvolvimento de softwares numéricos, especialmente na metodologia apresentada no sistema SINAI-16.

Estão disponíveis os módulos de resolução de equações polinomiais, transcendentes e está sendo desenvolvido o módulo de Sistemas de Equações Lineares.

Maiores informações e outros exemplos podem ser encontrados no manual instrucional do usuário do pacote de resolução de equações polinomiais.

5 REFERENCIAS BIBLIOGRAFICAS

[BEA87] BEAUFONT,C.E.C. & ROCHA, A.R.C. da. Verificação e validação de software na fase de especificação de requisitos. In: Conferência Latinoamericana de Informática (CLEI PANEL'87), XIII, Bogota, nov.9-13, 1987. pp.280-289.

[BOR87] BORGES, M.R.S. Uma análise realista do enfoque da prototipagem rápida no desenvolvimento de sistemas de informação. In: Conferência Latinoamericana de Informática (CLEI PANEL'87), XIII, Bogota, nov.9-13, 1987. pp.280-289.

[CLA87] CLAUDIO,D.M.,TOSCANI,L.V. and DIVERIO T.A. Fundamentos de Matemática Computacional. Porto Alegre, Sagra, 1987.

[COD74] CODY, W. J. The construction of numerical subroutine libraries. SIAM review, Philadelphia, 16(1):36-46, Jan.1974.

[COD82] CODY, W. J. Basic concepts for computational software. In: INTERNATIONAL SEMINAR ON PROBLEMS AND METHODOLOGIES IN MATHEMATICAL SOFTWARE PRODUCTION, Sorrento, Nov.3-8, 1980. Proceedings. Berlin, Springer Verlag, 1982. p.1-23.

[CRO77] CROWELL, W.R & FOSDICK, L.D. Mathematical software production. IN: MATHEMATICAL SOFTWARE III. New York, Academic Press, 1977.

[DIV86] DIVERIO, T. A. Software Numérico Aplicativo e Instrucional. Porto Alegre, CPGCC da UFRGS, 1986. (dissertação)

[DIV86a] DIVERIO, T. A. SINAI-16 Manual instrucional do usuário. Porto Alegre, CPGCC da UFRGS, 1986.

[DIV86b] DIVERIO, T. A. SINAI-16 Manual de manutenção do programa. Porto Alegre, CPGCC da UFRGS, 1986.

[DIV87] DIVERIO, T.A. Desenvolvimento de software numérico instrucional. In: CNMAC ,X, Gramado, set.20-25,1987. pp.225-229.

[DIV87a] DIVERIO, T.A. & ROMBALDI, V. SINAI-16 - um exemplo de software instrucional.In: CNMAC ,X, Gramado, set.20-25,1987. pp.796-799.

[DIV87b] DIVERIO, T.A. Resolução de equações - uma metodologia de estudo. In: Conferência Latinoamericana de Informática (CLEI PANEL'87), XIII, Bogota, nov.9-13, 1987. pp.190-203.

[DIV87c] DIVERIO. T.A. A problemática do desenvolvimento do software numérico. In: Conferência Latinoamericana de Informática (CLEI PANEL'87), XIII, Bogota, nov.9-13, 1987. pp.398-414.

[JOH82] JONHSTON, R.L. Numerical methods a software approach. New York, John Willey & Sons, 1982.

[RIC71] RICE, J. R. Mathematical software. New York, Academic Press, 1971. (ACM Monograph series)

[RIC85] RICE,J. Numerical Methods, software and analysis. Singapore, McGraw Hill, 1985.

[YOU72] YOUNG, D & GREGORY, R.F. A survey of numerical mathematics. Massachussets; Addilson Wesley, 1972.