UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

CURSO DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

SOFTWARE NUMÉRICO APLICATIVO E INSTRUCIONAL

por

TIARAJÚ ASMUZ DIVERIO

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

Prof Dr Dalcidio Moraes Claudio
Orientador

Porto Alegre, março de 1986.

CATALOGAÇÃO NA FONTE

Diverio, Tiarajú Asmuz
Software numérico aplicativo e instrucional
Porto Alegre, PGCC da UFRGS, 1986.
lv.
Diss. (mestr. ci. comp.) UFRGS-PGCC, Porto Alegre, BR-RS, 1986.

Dissertação: Matemática Computacional:

Desenvolvimento de Software Numérico:

Resolução de Equações: SINAI-16.

AGRADECIMENTOS

Agradeço à CAPES/PICD pelo apoio financeiro;

Ao Corpo Docente, Secretaria e Biblioteca da Pós-Graduação em Ciência da Computação da UFRGS;

Ao Instituto de Informática da PUC RS, em particular, à pessoa da Sra. Diretora Profa. Maria Lúcia Lisboa, pelo incentivo, pelo apoio para a realização deste trabalho, colocando à disposição os equipamentos do Laboratório de Programação (LAPRO), onde realmente foi realizada grande parte deste trabalho;

Ao auxiliar de pesquisa Ernesto Wendt, pela dedicação, pelo excelente trabalho realizado e pela amizade;

Ao meu orientador, Prof Dalcídio Moraes Claudio, pelo interesse e confiança demonstrados desde o tempo em que fui seu auxiliar de pesquisa. Pelo incentivo ao prosseguimento dos estudos, em especial à própria Pós-graduação e pela paciência e conselhos dados para realização deste trabalho;

À Profa. Laira Toscani pelo incentivo, pela confiança, pelas orientações e conselhos dados durante todo o curso;

Aos colegas do Laboratório de Matemática Computacional da UFRGS, Beatriz, Aristóteles e Laranja, pelo apoio e ajuda dados para a realização deste trabalho;

Aos colegas Celso Rodrigues e Margareth Schäffer pela amizade, prestigio, incentivo e consideração;

Aos meus pais, Jamile e Wanner, pelo amor e dedicação, pela formação e pela instrução que me foi dado;

À minha esposa Margarete pelo amor, carinho, compreensão e apoio dados durante todo este tempo;

Aos queridos irmãos da Igreja em Porto Alegre, pelo apoio, amizade e pelas orações.

Dedico este trabalho ao meu querido e amado Senhor e Salvador **Jesus Cristo,** que me tem concedido vida, saúde, sabedoria e amor no pleno conhecimento de seu Eterno Propósito, o qual é cumprido na Sua Igreja, o Seu Corpo Vivo.

SUMÁRIO

LISTA DE ABREVIATURAS 9

LISTA DE SINAIS 10

LISTA DE FIGURAS 12

LISTA DE TABELAS 13

RESUMO 14

ABSTRACT 15

1 INTRODUÇÃO 16

2 ASPECTOS MATEMÁTICOS 19
2.1 Introdução 19
2.2 Níveis de abstração e a abordagem matemática computacional 21
2.3 Terminologia e problemática da matemática numérica 24
2.4 Sistema numérico do computador..... 27
2.5 Natureza, propagação e quantificação de erros 29
2.5.1 Fontes de erros computacionais 29
2.5.1.1 Erro de modelação 29
2.5.1.2 Erro de truncamento 31
2.5.1.3 Erro inerente aos dados de entrada . 31
2.5.1.4 Erro de arredondamento 32
2.5.2 Fórmulas do erro nas operações aritméticas. 34
2.5.3 Uma fórmula computacional para o cálculo da exatidão..... 35
2.6 Tipos de algoritmos 36
2.6.1 Computação discreta 36
2.6.2 Enumeração 36
2.6.3 Iterativo 37
2.6.4 Divisão e conquista 38
2.6.5 Tentativa e erro 39
2.7 Instabilidade de problemas e algoritmos 39
2.8 Critérios de parada de métodos iterativos 42

3 SOFTWARE NUMÉRICO APLICATIVO E INSTRUCIONAL 46
3.1 Introdução 46
3.2 Software numérico - natureza e dificuldades 46
3.3 Níveis de atividades no desenvolvimento de software 48
3.4 O desenvolvedor de software numérico - o matemático computacional 50
3.5 O processo de desenvolvimento do software 53
3.5.1 Projeto e análise de algoritmos 55
3.5.2 Construção do software 56
3.5.3 Uma documentação detalhada 58
3.5.4 Vários testes 59
3.5.5 Distribuição e manutenção do software 60

3.6 Características desejáveis de um software numérico 61
3.6.1 Exatidão e eficiência 62
3.6.2 Confiabilidade e validação 62
3.6.3 Facilidade de uso 63
3.6.4 Boa documentação 63
3.6.5 Flexibilidade e modularidade 64
3.6.6 Robustez 65
3.6.7 Transportabilidade 66
3.7 Bibliotecas de rotinas, conjunto de programas e software numérico 67
3.8 A influência do tipo de usuário no desenvolvimento do software 68

4 MÁQUINAS & LINGUAGENS 72
4.1 Introdução 72
4.2 Abordagem do problema de máquina 74
4.2.1 Nome 74
4.2.2 Fabricante 74
4.2.3 Configuração básica 74
4.2.4 Microprocessador 75
4.2.5 Sistema Operacional 75
4.2.6 Capacidade gráfica 76
4.2.7 Linguagens 77
4.2.8 Compatíveis 80
4.2.9 Aplicações eficientes e aplicativos 80
4.2.10 Custo do equipamento 80
4.3 Descrição de sistemas 80
4.3.1 O sistema ED281 81
4.3.2 O sistema CP500 82
4.3.3 O sistema MAXXI 83
4.3.4 O sistema HP85 85
4.3.5 O sistema I7000PCxt 86
4.4 Conclusões preliminares 89

5 MÉTODOS DE RESOLUÇAO DE EQUAÇOES ALGÉBRICAS 91
5.1 Introdução 91
5.2 A ficha instrucional 91
5.2.1 Nome do método 92
5.2.2 Classificação 92
5.2.3 Objetivo 93
5.2.4 Descrição 93
5.2.5 Interpretação geométrica 94
5.2.6 Limites e restrições 94
5.2.7 Erro do método 94
5.2.8 Ordem de convergência 94
5.2.9 Índice de eficiência 95
5.2.10 Algoritmo 97
5.2.11 Implementações 97
5.2.12 Listagem do programa 97
5.2.13 Mensagens de erro 97
5.2.14 Dados de entrada e saída 98
5.2.15 Exemplos de utilização do programa 98
5.2.16 Bibliografia 98
5.2.17 Observações 98
5.3 A escolha dos métodos..... 98
5.3.1 Método da Bissecção..... 99
5.3.2 Método de Newton 100
5.3.3 Método da Secante 100
5.3.4 Método de Newton para raízes múltiplas 101
5.3.5 Método híbrido C 101

5.3.6 Método MIDREM 102
5.3.7 Método de Newton para raízes complexas 105
5.3.8 Método de Bairstow 107

6 SINAI-16 - UM EXEMPLO DE IMPLEMENTAÇAO 109
6.1 Introdução 109
6.2 Ambiente computacional 109
6.3 Finalidades 110
6.4 Estilo 111
6.5 Métodos implementados 111
6.5.1 Capacidade gráfica 112
6.5.2 Cálculo de cotas 112
6.5.3 Cálculo de raízes reais de equações 112
6.5.4 Cálculo de raízes complexas de equações ...112
6.5.5 Deflacionar de raízes calculadas 113
6.6 Características desejáveis presentes 113
6.7 Documentação 114

7 CONCLUSOES E SUGESTOES 116
7.1 Conclusões 116
7.2 Sugestões 117

ANEXOS 118
Apêndice A : Exemplo de ficha instrucional - método de Newton-Raphson 118
Apêndice B : Itens do Manual Instrucional do Usuário .124
Apêndice C : Itens do Manual de Manutenção do Programa125
Apêndice D : Organização das tela instrucionais do SINAI-16 126

BIBLIOGRAFIA 127

LISTA DE ABREVIATURAS

ACM	Association for Computing Machinery
ASC	Algarismos Significativos Corretos
CAPES	Comissão de Aperfeiçoamento de pessoal de Nível Superior
CPGCC	Curso de Pós-Graduação em Ciência da Computação
CP/M	Control Program / Monitor
Cz$	Cruzado
DOS	Disk Operation System
IMSL	International Mathematical and Statistical Libraries
LAPRO	Laboratório de Programação da PUC
MP/M	Multi-Processing Monitor Control Program
NAG	Numerical Algorithms Group
NATS	National Activity to Test Software
OTN	Obrigação do Tesouro Nacional
POSET	Conjunto Parcialmente Ordenado (em inglês)
PUC RS	Pontifícia Universidade Católica do Rio Grande do Sul
RAM	Random Access Memory
ROM	Read Only Memory
SIGMAP	Special Interest Group on Mathematical Programming
SIGNUM	Special Interest Group on Numerical Mathematics
SINAI_16	Software Interativo Numérico Aplicativo e Instrucional em micros de 16 bits
UFRGS	Universidade Federal do Rio Grande do Sul

LISTA DE SINAIS

R	Conjunto dos números reais
P	Sistema de ponto-fixo
F	Sistema de ponto-flutuante
Pn(x)	Polinômio de grau n
ai	Coeficiente do termo do i-ésimo grau de Pn(x)
f(x)	Função real
f'(x)	Derivada primeira da função
f''(x)	Derivada segunda da função
$f^{<n>}(x)$	Derivada n-ésima da função
x,y	Números reais
z	Número complexo
Qx	Número aproximado
ex	Erro da aproximação Qx
| |	Valor absoluto
x!	Fatorial de x
$\sqrt{\ }$	Raiz quadrada
x^n	Potenciação
Σ	Somatório
{Xn}	Seqüência Xn
pi	Número irracional (pi = 3,14159...)
ASC	Algarismos significativos corretos
m	Unidade de arredondamento
log	Logaritmo decimal
ln	Logaritmo neperiano
:=	Atribuição
+	Operação de adição
-	Operação de subtração

/	Operação de divisão
<	menor que
>	maior que
>=	maior ou igual
<=	menor ou igual
Ox	Arredondamento simétrico
$\bar{x}$	Arredondamento para cima
$\underline{x}$	Arredondamento por corte
max	valor máximo do conjunto
min	valor mínimo do conjunto
Ei	Índices de eficiência
p	Ordem de convergência
ø	Função de iteração
$\lim_{n\to\infty}$	limite quando n tende ao infinito
≠	Diferente

LISTA DE FIGURAS

Figura 2.1 Algoritmo que verifica POSET 21

Figura 2.2 Niveis de abstração & etapas da abordagem 23

Figura 2.3 Sistema de ponto-fixo (uniformemente denso)..... 29

Figura 2.4 Sistema de ponto-flutuante (densidade relativa uniforme)..... 29

Figura 2.5 Método dos trapézios (ilustração do erro) 30

Figura 2.6 Conceito de arredondamento 32

Figura 2.7 Gráfico do erro real 37

Figura 3.1 Algoritmo da avaliação de polinômio por potência 52

Figura 3.2 Algoritmo da avaliação de polinômio por produto 52

Figura 3.3 Algoritmo da avaliação de polinômio por Horner . 52

Figura 5.1 Classificação dos métodos iterativos 93

Figura 5.2 Esquema do algoritmo de Horner de fator quadrático 106

Figura 5.3 Esquema do algoritmo de Horner para cálculo da derivada 106

Figura 5.4 Esquema do algoritmo de Bairstow 108

LISTA DE TABELAS

Tabela 2.1 Tabela das raízes do polinômio perturbado do exemplo 2.2 40

Tabela 3.1 Tabela do número de operações para avaliação de polinômios de grau n 53

Tabela 4.1 Tabela de quadro comparativo de sistema de computadores 88

RESUMO

Este trabalho contém uma descrição de um ambiente matemático-computacional e das condições necessárias para o desenvolvimento de software numérico com finalidades aplicativas e instrucionais. É apresentada também, uma metodologia abrangente e instrucional para o estudo e pesquisa de métodos de resolução de equações algébricas, através de fichas instrucionais. É apresentada ainda uma parte de um software interativo numérico aplicativo e instrucional em micro de 16 bits, o SINAI-16.

ABSTRACT

This work contains a description of a computer-mathemathical enviroment and of the needed conditions for the development of numerical software with aplicable and instructive ends. A comprising and instructive methodology for the study and research of algebraic equations' resolution, through instructive cards is also presented. Besides it contains a part of an aplicable and instructive interactive numerical software in a 16-bits microcomputer, the SINAI-16.

1 INTRODUÇÃO

Bibliotecas de Software matemático podem ser vistas como o elo de ligação entre os usuários do computador e seus problemas. Mas, devido: ao recente desencadear da automação da sociedade; à falta de recursos aplicativos e instrucionais disponíveis no mercado, compatíveis com os microcomputadores existentes no Brasil; à evolução da importância e função de uma documentação consistente e simplificada ; e às diferentes finalidades comerciais (e não científicas) a que se destinam os softwares; os softwares matemáticos não têm acompanhado o desenvolvimento dos demais setores da Ciência da Computação e de setores que dela se utilizam.

Recentemente, com a utilização do computador no ensino e com o desenvolvimento da pesquisa em todas as áreas do conhecimento, tem-se verificado a necessidade de bibliotecas de softwares matemáticos numéricos, que contenham uma variedade de subrotinas, uma documentação consistente e simples, capaz de solucionar a grande parte das dúvidas do usuário imediatamente e uma exatidão computacional incluindo exatidão nos resultados e tempo de processamento. É preciso, ainda, que se tenha um estilo conversacional, possibilitando uma melhor ligação, interação entre o usuário e a máquina.

Este trabalho foi desenvolvido com a finalidade de orientar e encorajar o desenvolvimento de softwares numéricos, tanto com finalidade aplicativa, quanto instrucional. No capítulo 2 são descritos os aspectos matemáticos básicos, incluindo a abordagem matemático-computacional, terminologia e problemática.

Há também uma descrição do sistema numérico do computador, a natureza, propagação e quantificação de erros, bem como uma descrição de tipos de algoritmos normalmente usados, e os problemas de instabilidade de algoritmos e critério de parada de algoritmos iterativos.

O capítulo 3 contém a descrição detalhada do projeto de desenvolvimento de um software numérico, incluindo a natureza, dificuldades, níveis de atividades, a pessoa que desenvolve e o processo de desenvolvimento. São definidas também, as características desejáveis de um software numérico e as finalidades de uso através do tipo de usuário com suas influências no estilo e desenvolvimento do software.

As Máquinas e Linguagens são abordadas no capítulo 4, através de uma descrição caracterizadora de sistemas de computador visando dar condições de escolha do melhor equipamento para determinado uso. São descritos cinco sistemas de computador, os quais estão disponíveis em centros universitários (como na UFRGS ou PUCRS). São descritas ainda, algumas linguagens de programação disponíveis nestes sistemas, procurando estabelecer vantagens e desvantagens.

No capítulo 5 é proposta uma metodologia abrangente e instrucional para o estudo de métodos de resolução de equações algébricas, através de fichas instrucionais, contendo todas as informações dos métodos que poderiam se tornar um instrumento para catalogação dos métodos existentes, bem como auxiliar o estudo e desenvolvimento de novos métodos, como métodos híbridos, e ainda evitar estudos repetitivos que nada trazem de novo.

Por fim, temos o **SINAI-16 (Software Interativo Numérico Aplicativo e Instrucional em micro de 16 bits)**, parte de um software a ser implementado no microcomputador I7000PCxt da Itautec, (compatível com o PC da IBM). Esta parte do software consta de alguns métodos de resolução de equações algébricas, escolhidos de acordo com as finalidades aplicativas e instrucionais, de forma a fornecer um software de acordo com a orientação e recomendações deste trabalho.

São apresentados como anexo, um exemplo de ficha instrucional, os itens do manual instrucional do usuário, os itens do manual de manutenção do programa e um esquema da organização das telas do SINAI-16.

2 ASPECTOS MATEMÁTICOS

2.1 Introdução

A Matemática Computacional tem sido objeto de grandes conquistas no mundo científico. Entende-se por **Matemática Computacional** o ramo da matemática construtiva que estuda algoritmos implementáveis em máquinas digitais. Portanto, a Matemática Computacional trata da resolução construtiva, isto é, algorítmica, de problemas através do uso de máquinas digitais, podendo de uma maneira geral, ser dividida em quatro áreas: Matemática Numérica, Simbólica, Gráfica e Intervalar.

A **Matemática Numérica** trata da problemática da resolução de problemas matemáticos através do computador. Tais problemas não se resumem aos problemas analíticos, tradicionalmente abordados no curso de Cálculo Numérico, mas podem ser problemas de Algebra, como problemas de Grupos, Teoria de números, Relações e outros. Esta é uma ciência de enorme importância atualmente e engloba várias disciplinas tais como: Análise Numérica, Aritmética Computacional, Algebra Numérica e Estatística Numérica.

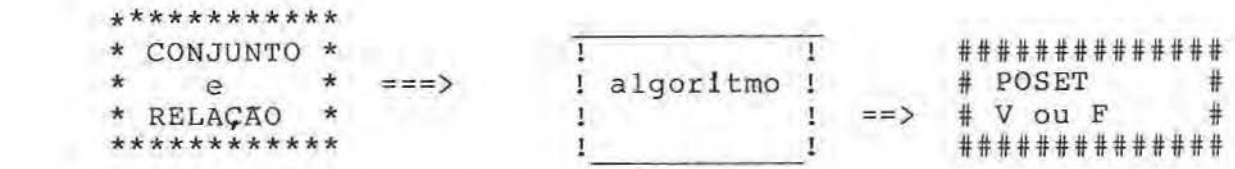
A **Matemática Simbólica** trata dos dados de forma literal, indexados ou não, preocupando-se em obter a solução EXATA para problemas matemáticos. Ela tem se mostrado útil na solução exata de problemas matemáticos, na classificação de estruturas algébricas, e no teste de conjecturas matemáticas.

Atualmente, só uns poucos centros mantêm sistemas capazes de executar cálculos simbólicos de modo eficiente.

A **Matemática Gráfica** trabalha com dados de forma gráfica, entendendo-se por dados gráficos tanto figuras planas e espaciais, como cores e sombra.

A **Matemática Intervalar** trata com dados na forma de intervalos numéricos. As aplicações da Matemática Intervalar ocorrem em diversas áreas, tais como: Geometria, Estatística, Cálculo Numérico, Equações Diferenciais e Lógica Matemática. Com o objetivo de automatizar a análise do erro computacional, a Matemática Intervalar iniciou e trouxe uma nova técnica que permite um controle de erros com limites confiáveis, além das provas de existência e não existência da solução de diversas equações. Ultimamente a Matemática Intervalar tem se desenvolvido rapidamente e já existem pesquisadores na área em diversos países, principalmente nos Estados Unidos e Alemanha Ocidental.

Uma ilustração da importância que a Matemática Computacional tem assumido por simplificar e facilitar demonstrações, é na verificação se uma relação define sobre um conjunto ou não, uma relaçao de ordem (com as propriedades reflexiva, anti-simétrica e transitiva), o que é bem exaustivo. A figura 2.1 descreve um algoritimo que efetua tal verificação. Os dados de entrada são a tabela de valores da relação, produzindo a resposta "verdadeiro", caso seja um conjunto parcialmente ordenado (POSET), e "falso", no caso contrário.




```
1.INICIO
2.Poset= falso;
3.PARA I = 1 incr 1 até N
4. FAÇA SE xi not R xi ENTÃO saia;
5.PARA I=2 até N
6. FAÇA PARA J=1 incr 1 até I-1
7.          FAÇA INICIO
8.                    SE (xi R xj) & (xj R xi) ENTÃO saia;
9.                    SE (xi R xj)
10.                    ENTÃO PARA k=1 incr 1 até N
11.                              FAÇA SE (xj R xk)&(xk R xi)
12.                                    ENTÃO saia
13.                    SENÃO PARA k=1 incr 1 até N
14.                              FAÇA SE (xi R xk)&(xk R xj)
15.                                    ENTÃO saia;
16.               FIM;
17.Poset = verdadeiro;
18.Fim.
```

Fig. 2.1 Algoritmo que verifica POSET

Isto mostra que muitos problemas matemáticos, há muito conhecidos, ganharam importância e mais utilidade com o desenvolvimento da Ciência da Computação.

2.2 Níveis de abstração e a abordagem da matemática computacional

O estudo da Matemática Computacional, como ferramenta de novas tecnologias e na ciência como um todo, têm se tornado cada vez mais útil nestes últimos anos.

Para o bom uso da Matemática Computacional, deve utilizar-se da Teoria da Computação, da Teoria da Informação e da Teoria dos Algoritmos. Além disso, a Matemática Computacional tem sua própria concepção, que difere da Matemática Pura, pois os conceitos de limite, continuidade e infinito, até mesmo de conjunto denso, não são aplicáveis, uma vez que se tem: uma máquina digital com seu sistema de ponto-flutuante (matematicamente discreto), uma memória (finita), uma quantidade finita de cálculos, e uma linguagem de programação com a qual se efetua as tarefas.

A Matemática Computacional, para resolução de problemas, tem sua abordagem especifica, que consiste na formalização precisa do problema, na idealização ou aproximação de um modelo matemático, na análise do problema matemático e no cálculo numérico da solução do problema. O cálculo numérico do problema é feito por um algoritmo numérico eficiente; em uma máquina digital conveniente para se obter o resultado dentro da faixa de aceitação, dentro do critério de exatidão; e com a impressão destes resultados.

Nesta abordagem têm-se diferentes niveis de abstração. Inicialmente o **mundo real**, ao qual pertence o problema genérico. A seguir, na formalização precisa do problema, procura-se levantar **informações relevantes ao sistema**, desconsiderando-se as demais, com a finalidade de estabelecer o modelo matemático que está associado a este problema, ou procurar um modelo que se aproxime, ou simule tal problema.

Uma vez estabelecido o modelo matemático e feita a análise matemática deste problema, é necessário escolher de um algoritmo numérico eficiente, como descrito em [CLA83].

Tem-se, então, o nível dos **Tipos Abstratos de Dados** (TAD), onde não se têm mais os entes reais, e sim abstrações, identificando-se os objetos e o tipo de operações que se pode fazer com eles. Passa-se depois para o nível da **Estrutura Lógica e Algoritmica**, o qual descreve o algoritmo numérico eficiente que resolve o problema. Este algoritmo é independente da máquina. Finalmente, passa-se, para o nível mais baixo, que é o da **Estrutura Física e Programas**, onde se considera a máquina a ser usada e a linguagem de programação comque esta máquina trabalha.

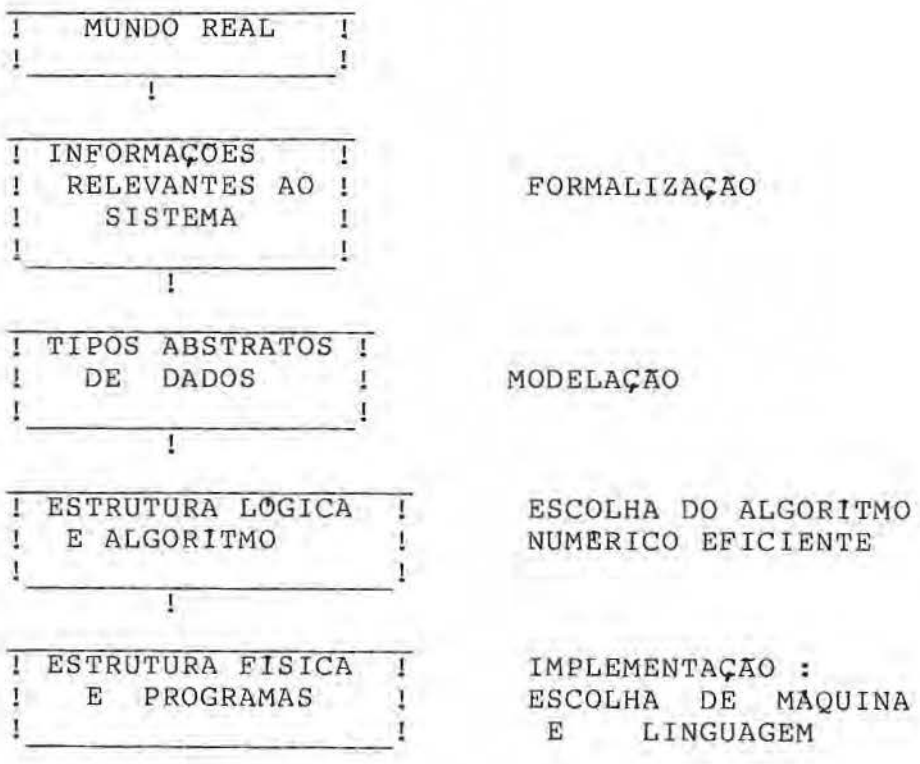


Fig. 2.2 Níveis de abstração & etapas da abordagem

Com o aumento da abstração, há um afastamento da realidade, que pode ocasionar uma perda de contexto ou produzir valores que não tenham sentido para a variável, como por exemplo: a idade de uma pessoa assumir valor negativo ou realizar operações não válidas a um tipo abstrato de dado.

2.3 Terminologia e problemática da matemática numérica

Um **algoritmo** é um procedimento constituído de um conjunto de regras não ambíguas, as quais especificam uma seqüência finita de operações que produzem uma solução para um problema ou uma classe específica de problemas. Um problema é algoritmicamente resolvível se existe um algoritmo (um programa, por exemplo), que para qualquer entrada, se for concedido o tempo e o espaço necessário a sua execução, produz uma resposta correta.

O fato de um problema ser teoricamente resolvível por um computador não é suficiente para o problema ser na prática resolvível. Muitos problemas de aplicação prática são algoritmicamente resolvíveis, mas o espaço ou tempo necessários para a sua execução são muito grandes para que o problema tenha utilidade prática. Um exemplo, é o método de Cramer, calculando o determinante pela definição.Para se resolver com ele um sistema de equações de ordem 20, seriam necessárias de dezenas de milhões de anos. Portanto, os recursos de espaço e tempo requeridos por um programa tem grande importância na prática.

A análise de um algoritmo tem como objetivo melhorar, se possível, seu desempenho e escolher, entre os algoritmos disponíveis, o melhor. Existem vários critérios de avaliação de um algoritmo, tais como: quantidade de trabalho requerido, quantidade de espaço requerido, exatidão de resposta e otimização.

A quantidade de trabalho requerido por um algoritmo não pode ser descrita simplesmente por um número, porque o número de operações básicas efetuadas em geral não é o mesmo para qualquer entrada e depende do "tamanho da entrada". Por exemplo, no método de Cramer para solução de um sistema de equações, o número de operações de adições e multiplicações variam com "n" (número de equações do sistema). A expressão "quantidade de trabalho requerido" é chamada **Complexidade do algoritmo.**

Uma das medidas de complexidade mais importantes é a complexidade de tempo. Existem vários aspectos para se medir a complexidade de tempo de um algoritmo. Um critério pode ser o da medida do tempo de execução requerido por diferentes algoritmos para a solução de um problema em um computador particular. Entretanto, uma medida empírica é fortemente dependente tanto do programa como da máquina usada para implementar o algoritmo. Assim, uma pequena mudança no programa pode não representar uma mudança significativa no algoritmo; mas, apesar disso, pode afetar a velocidade de execução. Logo, se dois programas são comparados, primeiro em uma máquina, depois em outra, as comparações podem dar resultados diferentes. Assim, apesar da comparação de programas reais em computadores reais ser uma fonte de informação importante, os resultados são devidos, inevitavelmente, à programação e às características da máquina.

Uma alternativa útil é uma análise matemática das dificuldades intrínsecas para se resolver um problema computacionalmente.

Na escolha de um algoritmo para se resolver um problema numérico, a **exatidão** é o critério mais importante. Como o computador é uma máquina finita, ele é capaz de representar somente uma aproximação finita do número real. Assim, se não são tomados cuidados especiais, um algoritmo numérico implementado em um computador pode produzir aproximações da solução com pouca ou nenhuma exatidão. Deve-se ter em conta, ainda, que a exatidão pode estar diretamente ligada com o tempo de execução, como é o caso, por exemplo, do algoritmo da Bissecção para determinação de raízes de equações algébricas, onde a convergência, embora garantida, pode ser muito dispendiosa.

Um algoritmo numérico eficiente é aquele que possui as seguintes características: inexistência de erro lógico - há um ponto de partida, mesmo não havendo entradas e há finitude das instruções e continuidade de uma em outra até uma instrução de parada, tendo havido no mínimo uma saída; inexistência de dados dependentes de máquina utilizada; inexistência de erro aritmético; haja uma quantidade finita de cálculos; a existência de um critério de exatidão; seja convergente e por fim, seja eficiente.

Faz-se necessário salientar a diferença, para fins da matemática numérica, existente entre **eficiência** e **eficácia** dos processos de cálculo.

Eficácia é a capacidade do processo em produzir uma resposta correta para o problema dado, enquanto que **eficiência**, além de intuir a idéia anterior, exige que o processo seja econômico nos termos de tempo, volume de dados de referência, volume de memória (computador), exatidão, enfim de custo operacional por quantidade de informação.

Algoritmo eficaz é aquele que produz uma resposta satisfatória, dentro do critério de exatidão, mas pode deixar de ser satisfatório na medida em que é anti-econômico.

Algoritmo eficiênte é aquele, dentre os algoritmos eficazes, que produz a melhor resposta em termos de custo operacional, isto é, produz uma resposta mais vantajosa.

2.4 Sistema numérico do computador

Matematicamente é assumido que todos os cálculos são efetuados no sistema R dos números reais, o qual seria infinito em dois aspectos. O primeiro, no intervalo onde não haveria maior número positivo e nem menor número negativo. O segundo sentido é que seria denso, ou seja, entre dois números racionais, sempre existem infinitos números racionais. Entretanto, quando nos voltamos à realidade computacional, a solução é calculada sem este sistema real infinito, e sim num sistema finito tanto no limitar o intervalo, quanto com uma precisão finita, isto é, os números são representados com um número finito de dígitos.

Na memória do computador, cada número é armazenado numa posição, que consiste em um sinal ("+" ou "-"), mais um número fixo de dígitos. Uma questão importante do projetar a máquina, é como utilizar estes dígitos na representação do número.

Um modelo consiste em considerar em duas partes, uma inteira e a outra fracionária, que é chamado de **Sistema de Ponto-Fixo**, o qual é caracterizado pelo número da base (b), o número de dígitos (n) e o número de dígitos da parte fracionária (f). É representado pela terna-ordenada $P(b,n,f)$.

Este conjunto é uniformemente denso no intervalo de representação e, como conseqüência, qualquer número real x neste intervalo pode ser representado por um elemento fix(x) pertencente a P(b,n,f), com erro absoluto definido por | x - fix(x) | e erro relativo definido por

$$\frac{| x - fix(x) |}{| x |}$$

A maioria dos computadores usa o Sistema de número em Ponto-Flutuante, denotado por F(b,n,l,u), onde "b" é o número da base, "n" a precisão e "l" e "u" são o menor e maior expoente do intervalo.

Qualquer número x, não nulo, pertencente a F tem a forma

$$x := \begin{matrix} + \\ - \end{matrix} \left(\frac{d1}{b^1} + \frac{d2}{b^2} + \frac{d3}{b^3} + \ldots + \frac{dn}{b^n} \right) . b^e$$

ou $x := \pm 0,d1\ d2\ d3 \ldots dn \ . \ b^e$ onde os digitos d1, d2, d3 ... dn da parte fracionária satisfaz:

(a) $1 <= d1 < b$

(b) $0 <= ds < b$ para s:=2, 3 ... n

e o expoente "e" é tal que $(l <= e <= u)$.

Também o zero deve pertencer a F, e é representado por $0 := +0,000...0 \ . \ b^1$.

Isto implica na vantagem do sistema de Ponto-flutuante sobre o sistema de Ponto-fixo: é um intervalo de densidade relativa uniforme.

Para ilustrar estes sistemas, são dadas as representações gráficas dos sistemas P(2,4,2) e F(2,3,-1,2), que são as figuras 2.3 e 2.4, respectivamente.

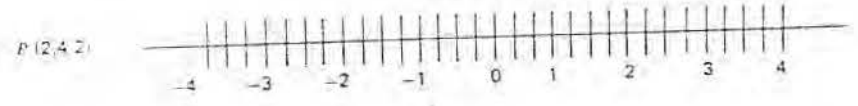


Fig. 2.3 Sistema de ponto-fixo (uniformemente denso)

Fig. 2.4 Sistema de ponto-flutuante(densidade relativa uniforme)

2.5 Natureza, propagação e quantificação de erros

2.5.1 Fontes de erros computacionais

Existem pelo menos quatro fontes de erros que influênciam a resolução de um problema computacionalmente. Estes erros podem ser caracterizados como: erro de modelação, erro de truncamento do modelo, erro inerente aos dados e erro de arredondamento, que colocados juntos se propagam, podendo afetar o resultado final.

2.5.1.1 Erro de modelação

Entende-se por erro de modelação, o erro proveniente de simplificações das situações reais que se faz através de um

modelo, o qual é uma analogia de alguns objetos ou fenômenos do nosso interesse. Sendo o valor do modelo melhor definido como o "grau de que" ele é capaz de responder questões e fazer predicados corretamente.

Em geral, usam-se modelos matemáticos, que são uma caracterização matemática do fenômeno ou processo. Um modelo matemático tem três partes essenciais: um processo ou fenômeno, o qual deve ser modelado; uma estrutura matemática, capaz de expressar as propriedades importantes do objeto a ser modelado e uma correspondência explicita entre estas duas partes. Portanto, um modelo matemático é uma abstração, que associa parâmetros e processos do mundo real com expressões em uma estrutura matemática. Esta abstração admite que se ignore todos aspectos do mundo real, os quais não são de interesse, com isto, um certo erro é introduzido no cálculo, como é o caso do cálculo da área determinada por uma curva f(x), pelas retas x1=a e x2= b e pelo eixo x. Esta área é calculada pela integral definida $I = \int_a^b f(x)dx$, que pode ser calculada pelo método dos Trapézios, onde seria calculada por $I \cong (f(a)+f(b)) \cdot (b-a)/2$. Este modelo introduz certo erro como é mostrado na figura 2.5 .

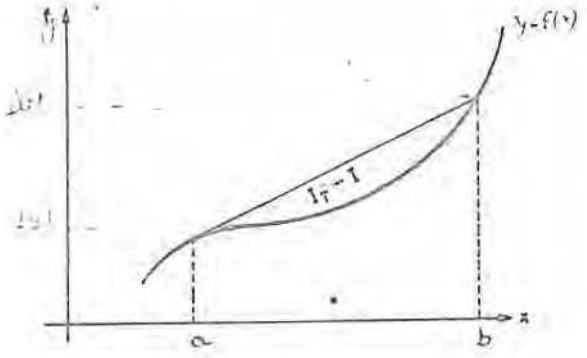


Fig. 2.5 Método dos trapézios (ilustraçao do erro)

2.5.1.2 Erro de truncamento

Entende-se por erro de truncamento de modelo, o erro introduzido pelo procedimento numérico infinito como, por exemplo, as séries infinitas de Taylor (ou de Maclaurin)

$$f(x) = f(x_0) + f'(x_0).(x-x_0) + f''(x_0).(x-x_0)^2/2! + \ldots + f^{<n>}(x_0).(x-x_0)^n/n! + \ldots$$

onde calculamos o desenvolvimento até um número finito de n termos, e desprezamos um número infinito de termos, truncando, portanto, o modelo infinito. Este erro de truncamento pode ser calculado por muitas maneiras, entre elas está o próprio teorema de Taylor.

2.5.1.3 Erro inerente aos dados de entrada

Entende-se por erro inerente aos dados de entrada, o erro nos valores dos dados, causados por inexatidão em medidas como distância, tempo e temperatura, que são medidas por instrumentos imprecisos; por enganos pessoais ou pela natureza necessariamente aproximada da representação com um número finito de dígitos, a qual não pode representar exatamente os números com esta precisão, como é o caso dos números transcendentes, irracionais ou periódicos.

Cabe ainda ressaltar que os números transcendentes e irracionais não têm uma representação finita em nenhuma base numérica. Já os números racionais, podem ter representação finita em uma base e infinita ou periódica em outra.

2.5.1.4 Erros de arredondamento

Por fim, entende-se por erro de arredondamento, o erro resultante da associação de um número real x a um número de máquina Qx do sistema numérico da máquina digital, como ilustra a figura 2.6 .

Existem muitas formas de determinar a associação do número real x a aproximação Qx, estas definem os **tipos de arredondamento.** Entre os tipos de arredondamento mais usados estão o **arredondamento por corte (ou truncamento)**, representado por $\underline{x}$, no qual, para se ter um número com n dígitos, trunca-se o número na posição do dígito n. Este tipo de arredondamento tem, como calculado em [DOR74], um erro relativo máximo de

$$e_{\underline{x}} := b^{1-n} .$$

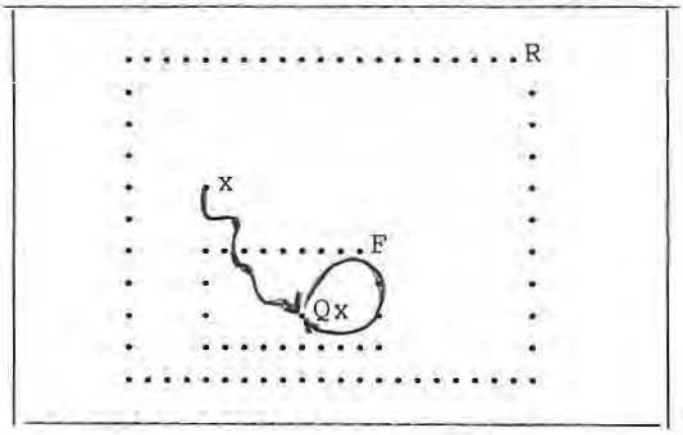


Fig. 2.6 Conceito de arredondamento

O **arredondamento para cima (ou excesso)**, representado por $\bar{x}$, no qual, para se ter um número com n dígitos, trunca-se o número na posição do dígito n, e soma-se uma unidade ao dígito n. O erro relativo máximo é igual ao erro do truncamento.

Outro arredondamento muito usado e conhecido é o **arredondamento simétrico** ou, como é conhecido, arredondamento para número mais próximo de máquina, representado por 0x. Considera-se para se ter uma precisão de n dígitos, o dígito n+1, caso este seja menor ou igual a quatro, trunca-se na posição do dígito n; caso contrário, aplica-se o arredondamento para cima. Neste tipo de arredondamento, o erro relativo é dado por:

$$e_{0x} = 1/2 \cdot b^{1-n} .$$

Comparando-se estes valores do erro relativo com as implementações existentes na maioria dos computadores, verifica-se que, apesar de o erro de truncamento proporcionar um erro maior do que o arredondamento simétrico, ele é o mais utilizado nos computadores. Isto se deve, em parte, ao fato de os computadores serem feitos com finalidade comercial, sem necessidade de uma performance matemática exigente, sendo o arredondamento por truncamento mais econômico em termos de tempo, pois não necessita testar o dígito n+1, como no arredondamento simétrico. Atualmente, apesar do arredondamento por truncamento a maioria dos computadores têm algumas feituras de hardware, como o dígito de guarda, que no momento do arredondamento guarda o dígito n+1 e no momento das operações o torna disponível, compensando parte da perda.

Dentro da Matemática Intervalar, são amplamente utilizados os arredondamentos por corte e o para cima. Um exemplo tem-se no cálculo da solução de uma equação, que se aproxima numericamente a solução, e então, abre-se esta solução em um intervalo $[\underline{x} ; \overline{x}]$ e então, aplica-se as versões intervalares dos métodos para o cálculo da solução da equação.

2.5.2 Fórmulas do erro nas operações aritméticas

Deve-se ainda atentar a questão de quanto estes erros se propagam em cada uma das operações, para isto consideraremos dois números reais x e y, com suas aproximações Qx e Qy e sejam ex e ey seus respectivos erros, ou seja,

$$x = Qx + ex \qquad e \qquad y = Qy + ey$$

temos que o erro da soma é a soma dos erros, dado pela fórmula:

$$e(x+y) = ex + ey,$$

o erro da diferença, é a diferença dos erros, dado pela fórmula:

$$e(x-y) = ex - ey,$$

os erros do produto e do quociente são dados por:

$$e(x.y) = Qx.ey + Qy.ex$$

e

$$e(x/y) = (Qx+ex)/Qy \,.(1 - ey/Qy + ey^2/Qy + \ldots),$$

como é visto e calculado em [DOR79].

Tem-se considerado o erro de arredondamento no resultado de operações aritméticas, mas estas ocorrem muitas vezes na máquina digital, de modo que é necessário considerar a propagação e o efeito final, procurando minimizá-lo. Isto pode ser feito tanto por software, quanto por hardware. Neste último, com o acréscimo do dígito extra na parte fracionária de um número e na unidade lógica e aritmética do computador, podendo assim previnir a perda da exatidão em algumas situações, como no dígito menos significativo. Por software, uma maneira é a utilização de palavra longa no resultado de operações aritméticas, como a multiplicação e divisão.

2.5.3 Uma fórmula computacional para o cálculo da exatidão

Vê-se em [CLA83], o seguinte teorema:

"Seja F(b,n,l,u) um sistema de ponto-flutuante, onde b é a base do sistema, n é o número de algarismos da mantissa, l é o menor expoente e u é o maior expoente. Seja um arredondamento monotônico, transformando números reais em números de F. Então, para todo x real, vale a seguinte fórmula:

$$b^{1-l} <= |x| <= B \text{ implica que } \frac{|Qx - x|}{|x|} <= m$$

onde, $m := \begin{cases} 1/2\ b^{1-n}, & \text{no caso de arredondamento simétrico} \\ b^{1-n}, & \text{nos outros casos.} \end{cases}$

"Sendo m chamado de **unidade de arredondamento**, por ser o erro relativo máximo."

Disto, Dalcídio conclui que, para se obter uma melhor aproximação na região $b^{1-l} < |x| < B$, (que é o intervalo de representação da máquina, uma vez que, b^{1-l} é o menor número de máquina e B o maior número de máquina), deve-se aumentar o número de algarismos da mantissa e, para se obter uma melhor aproximação fora desta região, deve-se aumentar o número de algarismos utilizados na representação do expoente.

Dalcídio também define **Algarismos Significativos Corretos**, denotado por **ASC**, como sendo os algarismos de um valor aproximado Qx, que coincidem com os algarismos do valor exato x, até que ocorra a primeira discrepância. O que pode ser calculado pela fórmula:

$$ASC(Qx) := -(0{,}3 + \log(m + (|Qx-x| / |x|))).$$

Exemplo 2.1: Seja o sistema de ponto-flutuante F(10,10,1,u) com arredondamento simétrico, e seja a aproximação Qpi = 3,141587810 do número irracional pi. Esta aproximação

possui uma exatidão de:

$$ASC(Qpi) = -(\emptyset,3 + \log(\emptyset,5.10^{-9} + |Qpi-pi|/|pi|))$$

$$= 5,512$$

o que significa, que os cinco primeiros algarismos de Qpi estão corretos.

Maiores detalhes do cálculo da fórmula do ASC, encontram-se no apêndice A do livro de Dalcídio ([CLA83]).

2.6 Tipos de algoritmos

Apesar de não existir um conjunto de regras universais para se projetar algoritmos e cada problema necessita nova idealização e uma "modelação aproximadora", existe um número de tipos básicos de algoritmos que podem ser freqüentemente usados para resolver problemas.

2.6.1 Computação discreta

É o tipo de algoritmo onde a resposta EXATA pode ser obtida por uma seqüência de computações elementares. Esta forma de algoritmo é muito aplicada para problemas simples, nos quais a própria descrição do problema já especifica o que necessita ser feito para resolvê-lo.

2.6.2 Enumeração

É o tipo de algoritmo que experimenta TODAS as possíveis "respostas" em ordem para encontrar uma que solucione o problema. Uma pesquisa seqüêncial é um exemplo de enumeração. O desempenho de um algoritmo por enumeração é, em geral, muito lento. Sendo entretanto desejável, a identificação de situações

de casos impossíveis, para se restringir o domínio da busca e se ganhar tempo. Algumas vezes não se tem outro tipo de algoritmo que resolva o problema a não ser a enumeração. Este tipo de algoritmo é a base de muitos programas para problemas não numéricos.

2.6.3 Iterativo

No algoritmo iterativo, uma série de respostas aproximadas, gradualmente precisas são calculadas até que uma esteja "suficientemente perto". Suficientemente perto significa que um critério de parada ou de exatidão seja satisfeito. Esta técnica é geralmente aplicável a problemas numéricos, como no calculo do seno de um ângulo pela série infinita de Taylor ou o método de Iteração linear (de ponto-fixo) para o cálculo do zero de uma função.

Teoricamente, quando existe convergência, tem-se que quanto maior o número de iterações, menor é o erro na solução, mas, computacionalmente, o erro de arredondamento também afeta, tendo um ponto ótimo entre o relacionamento do erro de truncamento do modelo com o erro de arredondamento, como ilustra a figura 2.7 .

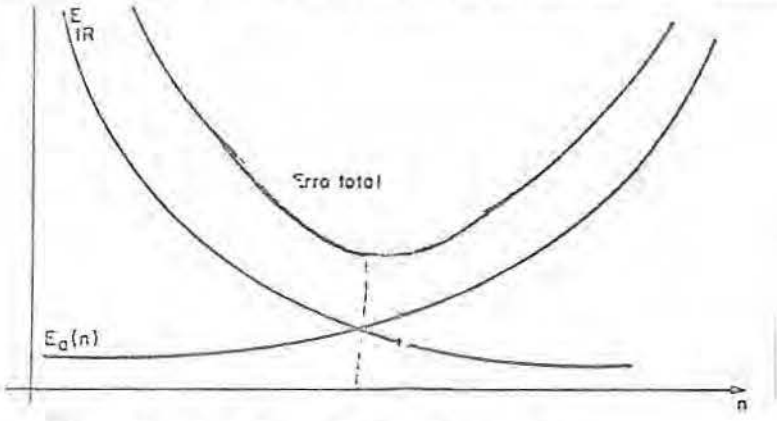


Fig. 2.7 Gráfico do erro real

Na utilização de métodos iterativos, emprega-se a fórmula dos Algarismos Significativos Corretos, com uma

modificação, pois não se conhece o valor real de x, sòmente as aproximações, já que a solução real x é o limite da seqüência {Xn}, quando n tende ao infinito. Então se tem:

$$ASC(xn) \equiv -(0{,}3 + \log(m + |xn+1 - xn| / |xn|)),$$

calculando-se, portanto, os algarismos coincidentes em duas aproximações sucessivas.

2.6.4 Divisão e conquista

Um algoritmo por "divisão e conquista" divide o problema em vários subproblemas similares, mas menores, que podem ser resolvidos diretamente ou subdivididos novamente, usando a mesma técnica, até que possam ser resolvidos. Então, acha-se uma forma de combinar as soluções parciais para se obter a solução para o problema original.

Em geral, os subproblemas resultantes da aplicação do método são do mesmo tipo que o problema original. Neste caso, a aplicação sucessiva do método pode ser expressa naturalmente por um algoritmo recursivo, isto é, dentro de um algoritmo chamado M, com uma entrada de certo tamanho, usamos o próprio M, k vezes para resolver sub-entradas de tamanhos menores. Os algoritmos recursivos são relativamente fáceis de serem escritos e compreendidos, além de serem de fácil análise quanto à repidez e à certificação.

Sabe-se ainda da Teoria da Computação que qualquer algoritmo recursivo pode ser transformado num algoritmo iterativo eqüivalente, embora menos fácil de ser compreendido.

Esta técnica de divisão e conquista para algoritmos é poderosa para resolução de problemas numéricos e não numéricos. Um exemplo não numérico muito conhecido é a pesquisa binária.

2.6.5 Tentativa e erro

Neste tipo de algoritmo há uma espécie de iteração, pois cada aproximação é baseada no grau ou quantidade de erro da aproximação anterior. Um exemplo típico da tentativa e erro é o tradicional jogo de advinhar o número que alguém escolheu entre certo limite, como por exemplo de 1 a 100. A partir da escolha de uma tentativa, para a qual é respondido se acertou e, no caso de ter errado, se é maior ou menor. Do resultado de cada tentativa é feita outra, caso não se tenha acertado ou passado o número de possíveis tentativas. Outro exemplo é o método da Bissecção para o cálculo da raiz de uma função em um intervalo.

2.7 Instabilidade de problemas e algoritmos

Tem-se visto o ambiente do software numérico, com sua problemática e terminologia, bem como a realidade da máquina digital. Caracterizou-se também diferentes tipos de algoritmos. Agora, pretende-se caracterizar o problema da **instabilidade de problemas e algoritmos**, que tanto influênciam o desempenho de um software numérico, resultando em situações indesejáveis.

Diz-se que um problema matemático é **instável, mal condicionado ou mal posto**, se sua solução é muito sensível a pequenas mudanças nos dados de entrada. Como é o caso do exemplo, dado abaixo, de uma equação polinomial de grau 20, onde uma pequena mudança de um dos coeficientes ocasiona que metade das raízes do polinômio se tornem complexas.

Exemplo 2.2: Seja o polinômio $Pn(x) = (x-1)(x-2)..(x-20)$

que é da forma:

$$Pn(x) = x^{20} - 210x^{19} + ... + 20!,$$

onde todas as raízes são inteiras e positivas.

Uma pequena variação da ordem de 10^{-7} é acrescentada ao termo a19 (que vale -210), ocasionando uma modificação nas raízes do polinômio, que são dadas pela tabela 1.1 .

Tabela 2.1 Tabela das raízes do polinômio perturbado do exemplo 2.2

Raízes reais		Raízes complexas
1,00000	6,00001	10,09527 +/- 0,64350 i
2,00000	6,99970	11,79363 +/- 1,65232 i
3,00000	8,00727	13,99236 +/- 2,51883 i
4.00000	8,91725	16,73074 +/- 2,81262 i
5,00000	20,84691	19,50244 +/- 1,94033 i

Esta mudança não é decorrente do método, mas é devido a própria natuereza do problema. O problema em si é sensível, não importando o método de solução usado. A grande dificuldade é detectar a instabilidade de um problema e sua magnitude.

Em contraste, dizemos que um problema é estável, bem-condicionado ou bem-posto, caso pequenas alterações nos dados, sejam acompanhados de pequenas variações na solução do problema.

No exemplo 2.3, tem-se uma mudança no sétimo dígito em um dos coeficientes da equação, o que resultou uma mudança no quarto dígito da solução.

Exemplo 2.3: Seja o polinômio

$P3(x) = X^3 + 96x^2 - 396x + 400 = 0$, que tem por uma das soluções x=2. Suponha que a equação sofra uma modificação no termo a0, resultando na equação

$P3'(x) = x^3 + 96x^2 - 396x + 399,9998 = 0$, cuja a solução agora é x= 2,001.

Um algoritmo é dito **mal condicionado ou instável**, quando o resultado dependerá da maneira pela qual os dados são manipulados. Ele pode ser muito bom algumas vezes e outras não, como é o caso do exemplo 2.4, da conhecida fórmula de Báskara, para resolução de equações do segundo grau.

Exemplo 2.4: Seja a equação polinomial do segundo grau, da forma $P2(x) = ax^2 + bx + c = 0$, com $a \neq 0$. Como é conhecido, pode ser resolvido por Báskara, que se constitui em calcular x1 e x2 por

$$x1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad \text{e} \quad x2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$$

Se quisermos calcular as raízes de uma equação, onde a e b são maiores que zero e b^2 é muito maior do que 4ac, a fórmula da Báskara para o cálculo de x1 conduzirá a um erro significativo, como é o caso da equação $x^2 + 80x + 1 = 0$ (vamos usar uma máquina com 3 dígitos significativos e arredondamento por corte).As raízes desta equação são x2=-79,9 e x1=-0,0125, mas pela fórmula de Báskara temos x1=-0,0500 e x2=-79,9.

Se x1 fosse calculado por $x1 = \frac{-2c}{b + \sqrt{b^2 - 4ac}}$,

obter-se-ia, x1= -0,0125, que é a solução exata. Conclui-se que a fórmula de Báskara é um algoritmo instável.

Muitos algoritmos que são propostos para solucionar certos problemas, seriam estáveis se a aritmética real fosse usada, mas devido a aritmética computacional e aos erros de arredondamento tornam-se algoritmos instáveis na prática.

Uma das causas freqüentes de instabilidade é chamada de "subtração de cancelamento", que consiste em subtrair dois números de mesma grandeza mas de sinais opostos.

Ao testar um novo algoritmo, temos que cuidar se está sendo aplicado a um problema instável. Se o algoritmo introduz erros pequenos ou há erros de arredondamento, a instabilidade do problema irá resultar num grande erro. Estes erros não invalidam o algoritmo, pois o algoritmo pode produzir boas respostas para problemas estáveis. Portanto, para se fazer um julgamento apropriado da estabilidade de um algoritmo, não se pode confundi-la com a estabilidade do problema em que o algoritmo é aplicado.

Duas técnicas para análise da estabilidade de algoritmos são: a da comparação entre a solução exata e a solução produzida do algoritmo, que é chamada de análise posterior do erro; e a outra, considera-se a solução produzida pelo algoritmo como exata, e analisa-se o efeito de uma perturbação no resultado.

2.8 Critérios de parada de métodos iterativos

Vários critérios de parada para métodos iterativos que calculam zeros de polinômios tem sido utilizados, entre estes se tem, para dado polinômio

$$Pn(x) = anx^{n} + an-1x^{n-1} +...+ a2x^{2} + a1x + a0 ,$$

a) critério 1: $$\frac{|\ Xk+1 - Xk\ |}{|\ Xk+1\ |} <= EPS$$

b) critério 2: $$|\ Pn(Xk+1\ | <= EPS\ (\sum_{j=\emptyset}^{n} |\ aj.Xk+1^{j}\ |\ )$$

c) critério 3: $$|\ Pn(xk+1)\ | <= EPS.(\ \max_{paraj=\emptyset,1,2,\ldots,n} \{|\ aj.Xk+1^{j}\ |\})$$

onde EPS é uma constante dependente do número de dígitos da mantissa, num sistema de aritmética de ponto flutuante.

Estes critérios não levam em conta propriedades particulares do polinômio que está sendo avaliado, tais como: existência de raízes múltiplas ou raízes próximas ou ainda, se tais critérios são eficientes para o cálculo de raízes complexas (ou até mesmo, se são aplicáveis quando polinômios possuem coeficientes complexos).

Kahan e Farkas (em [KAH63]) calculam o limite do erro de aproximação para um polinômio real,avaliado em um número real, conforme a descrição resumida de como é determinado o critério, que é dado a seguir.

A fórmula de Horner para avaliar o polinômio Pn(x) é dada por:

$$b\emptyset := an$$
$$bk := x.bk-1 + an-k \quad , \text{ para } k= 1,2,\ldots,n \ ,$$

onde o último termo bn, é a avaliação de Pn(x).

Os erros de aproximação no cálculo de bk são limitados pelos erros da soma e do produto, (S:= u+v e P:= u.v), que são dados por: $$\frac{|\ u+v - S\ |}{|\ S\ |} <= T <= 1/2.b^{1-n}$$ e

$$\frac{|\ u.v - P\ |}{|\ P\ |} <= W <= 1/2.b^{1-n}$$

onde os números T e W usados são limites, os quais influenciam em cada adição e multiplicação, respectivamente, na avaliação de qualquer polinômio. Como já visto, b é a base numérica do sistema de ponto-flutuante e n a precisão (número de digitos da mantissa).

Associado a fórmula de Horner, tem-se outra fórmula recorrente, dada por

$$e0 := \frac{|an|.W}{(W+T)}$$

$$ek := |x|.ek-1 + |bk|, \text{ para } k=1,2,..,n$$

Kahan mostrou que o limite do erro é dado por

$$| Pn(x) - bn | <= (T + W).en - | bn |.W$$

Ele também mostrou um critério proveitoso de aceitação de x como zero do polinômio Pn(x), com limites de erro dado por

$$| bn | <= 2E$$

onde $E := (T + W).en - | bn |.W$.

O fator 2, na fómula acima, garante a existência de pelo menos um número de máquina.

D. A. Adams extendeu este critério para a avaliação de polinômio real, em um número complexo, o que pode ser encontrado em [ADA67].

Igarashi, em [IGA84], para definir seu critério da parada, avaliou o polinômio Pn(x) de duas maneiras. A primeira por Horner, como descrito anteriormente, denominando A(x):=bn, (onde bn é o último termo, o valor da avaliação de Pn(x)); e a segunda maneira de avaliar é dado por

$$B(x) := x.f'(x) - G(x),$$

onde

$$G(X) := (n-1)anx^{n} + (n-2)an-1x^{n-1} +..+ a2x^{2} - a0 = xf'(x) - Pn(x)$$

O critério de parada de Igarashi é dado por:

$$|A(Xk) - B(Xk)| >= \min \{|A(Xk)|, |B(Xk)|\}$$

sendo interpretada sua aplicação em três partes:

a) Se A(Xk)=Ø ou B(Xk)=Ø, então a iteração termina;

b) Se A(Xk).B(Xk)<Ø, então a iteraçao termina;

c) Se A(Xk).B(Xk)>Ø, então o critério é mudado para:

$$\frac{| A(Xk) - B(Xk) |}{\min\{|A(Xk)|, |B(Xk)|\}} >= 1$$

o que significa que a iteração poderá terminar quando

$$2|A(Xk)| <= |B(Xk)| \quad \text{ou} \quad 2|B(Xk) <= |A(Xk)|.$$

Este critério foi testado em [IGA84], e evidenciou eficiência para cálculo de raízes complexas, sendo também aplicável a polinômios de coeficientes complexos.

Considerando que se está preocupado com um software numérico aplicativo e instrucional e que apesar destes limites para erros poderem ser obtidos através de métodos de alcance aritmético ou aritmética intervalar, estes requerem uma grande quantidade de computações, (o que poderia comprometer a performance de software em micros), além de não ser o objetivo deste trabalho. Portanto, será adotado como critério de parada, a já vista na seção 2.5.3, fórmula computacional para o cálculo de exatidão, ASC, e descrita com detalhes em [CLA83].

3 SOFTWARE NUMÉRICO APLICATIVO E INSTRUCIONAL

3.1 Introdução

No capítulo anterior se abordou a problemática na Matemática Computacional, descrevendo o ambiente matemático-computacional, com as fontes de erros computacionais, com fórmulas de quantificação do erro, tipos de algoritmos, instabilidade de algoritmos e critérios de parada.

Neste capítulo, será descrito o software numérico (matemático), considerando-se os níveis de atividade no desenvolvimento do software e os aspectos básicos no processo de desenvolvimento do software, bem como características desejáveis em software numérico. São caracterizadas as "filosofias" de software aplicativo e Instrucional, com o estilo conversacional.

3.2 Software numérico - natureza e dificuldades

O termo "software numérico" ou "software matemático", inicialmente, referia-se a programas com uma performance basicamente matemática na computação de problemas referentes a métodos numéricos que resolvem problemas na Ciência e Engenharia, como é descrito em [CR077].

Software numérico é muito mais do que uma rotina programada de tarefa ou um produto da análise numérica, a qual algumas vezes, situa-se entre a matemática e a ciência da computação, sendo tão aplicada para alguns e irrelevante para outros.

primariamente, o software numérico necessita tolerar erros. A palavra "erro" é inoportuna, porque "erros" numéricos não são erros no sentido usual da palavra, mas diferenças entre aproximações, que podem ser computadas num período finito de tempo.

Software numérico tem dois princípios característicos: o de tratar com aproximações de números reais e o fato de o intervalo de representação destes números varia de acordo com o computador, como já descrito no capítulo anterior.

O fato de que computadores diferem no tratamento de números em ponto-flutuante, significa que se precisa considerar classes de computadores. Há algumas décadas atrás, tratava-se do ambiente de um particular computador, mas hoje, considera-se sobre ambientes em classes de computadores, nas classes de linguaguens. Estas classes de computadores têm diferentes intervalos numéricos, diferentes precisões e diferentes propriedades de arredondamento. Software numérico é escrito para se trabalhar nestes vários ambientes. Ele necessita ser sensível à precisão do computador, para que não se tente calcular maior exatidão do que é possível em uma máquina particular. Ele precisa ser sensível ao intervalo numérico para não permitir overflows e underflows desnecessários, e estar alerta a particularidades de erros de arredondamento.

Software numérico pode ser particularmente difícil de ser projetado, porque mesmo antes da precisão, intervalo e problemas de arredondamento, muitas tarefas numéricas são, em um sentido, insolúveis. Isto não quer dizer que o problema seja teoricamente impossível. Ele pode ser "resolvido", podendo ter como "solução":

a) Um resultado correto e com exatidão requerida ou esperada pelo usuário,

b) Uma parte do programa dá que a tarefa é impossível, por não convergência ou ocorrência de algum erro;

c) Uma solução sem a exatidão desejada.

Os tipos de problemas numéricos podem ser classificados, de acordo com C. W. Gear em [GEA80], segundo três critérios que são as propriedades dos dados, propriedades dos algoritmos e propriedades do erro de arredondamento. A análise e descrição destes tipos de problemas vão além do propósito deste trabalho, embora se constate a dificuldade do projetar softwares numéricos.

Gear, ainda, menciona que grande parte das dificuldades do projeto de software numérico é devido à perda da base científica em ambiente computacional. Não existindo critérios simples que possam ser otimizados, mas um conjunto mal-definido de objetos como a confiabilidade, generalidade e utilidade que se está "tão longe" de falhar em quantificar.

Ele considera também o software numérico como parte da ciência da computação e espera que seja estudado em departamentos de ciência da computação ou informática.

3.3 Níveis de atividades no desenvolvimento de software

Existem alguns níveis distintos de atividades no desenvolvimento de software numérico. O trabalho feito em cada nível particular é um pouco independente do trabalho feito em outro nível, embora as diferenças entre níveis se tornem menos distintas à medida que se ganha mais experiência na preparação de software.

Cody, em seus artigos [COD82] e [COD74] identifica três níveis. O primeiro é o do desenvolvimento da matemática teórica ou análise numérica com seus métodos numéricos teóricos para resolver um particular tipo de problema. A ênfase neste nível está nas considerações teóricas, tais como análise do erro e provas de convergência.

O segundo nível de atividade é o da combinação de um ou mais métodos numéricos teóricos e sua transformação em algoritmo computacional para solucionar um determinado problema particular em um computador. O produto final deste segundo nível de atividade é muitas vezes considerado ser um programa computacional, especialmente quando o trabalho é publicado em uma linguagem algébrica de programação em um jornal profissional. Entretanto, usaremos o termo formal **algoritmo** para descrever o resultado deste nível.

Cody define software computacional como sendo um módulo executável de programação, junto com a documentação dos algoritmos implementados em um ambiente específico.

A palavra-chave na definição de software computacional é **implementação**. Algoritmos na literatura não resolvem problemas em computadores, sòmente implementações de algoritmos resolvem problemas. Um bom programa precisa ser baseado em um bom algoritmo, mas um bom algoritmo não garante um bom programa. Uma implementação imprópria de um ótimo algoritmo pode resultar numa péssima performance, comprometendo a reputação do algoritmo. Então, precisa-se atentar para os detalhes da implementação. Isto introduz o terceiro, e mais importante nível de atividade: a implementação de algoritmos para produzir um software computacional, isto é, o módulo executável de programação, junto

com sua documentação.

Neste terceiro nível da atividade, a transformação de algoritmos computacionais em um software robusto freqüentemente envolve mudanças algorítmicas e inovações, as quais não podem ser feitas sem a ajuda de análise numérica.

3.4 O desenvolvedor de software numerico - o matemático computacional

Existe uma significativa diferença entre o interesse e proximidade do analista numérico com a pessoa que desenvolve o software numérico. Esta pessoa será chamada de **matemático computacional**, para melhor caracterização.

O matemático computacional é um matemático que desenvolve, analisa e cria o algoritmo computacional, para obter uma solução aproximada para um problema matemático. Muitos destes problemas originam-se no campo da engenharia, da física, das ciências biológicas e sociais.

O matemático computacional tem como meta ser um suporte para outros trabalhadores e sua razão de ser é o sucesso do serviço numérico para usuários de computadores, através da criação e organização de bibliotecas de algoritmos numéricos. Eles não são, e nem pretendem ser analistas numéricos. Conquanto, o que eles são, essencialmente, numéricos fluentes, capazes de ler comandos em linguagens de alto nível, com uma apreciação por hardware e sistemas operacionais e devem ter a facilidade de escrever documentações claras e possuir habilidade de organização, somada à sensibilidade para as necessidades de usuários de computadores e ao desejo de vender seus produtos.

Os interesses do analista numérico são bem entendidos, entretanto, estão aquém das necessidades dos usuários de computadores. A força da motivação de um analista numérico é pessoal, estando muitas vezes tão envolvido em seu próprio interesse teórico, para produzir algoritmos ou programas úteis a um grupo de usuários.

Os trabalhos do analista numérico e do matemático computacional estão relacionados e não existe uma clara delimitação entre estas funções, além do que, existem muitos indivíduos, os quais atuam em ambas funções.

Duas atividades intelectuais de importância são a **análise e a síntese**. Analisa-se classes de problemas e classes de métodos, mas depois, os resultados são de pouca utilidade para os analistas. Os resultados da análise precisam ser sintetizados em um programa. O matemático computacional é a pessoa interessada em ambas as atividades. E, segundo Gear, "a análise é a ciência e a síntese é a arte". O resultado será o software numérico propriamente dito.

Na aplicação da análise e síntese para produçao do software numérico, o matemático computacional é, freqüentemente, requerido para desenvolver novos resultados matemáticos e para adaptar novos resultados para uso efetivo do computador digital automático. Os cálculos em algoritmos assim desenvolvidos são feitos por análises baseadas tanto em considerações matemáticas, como em considerações computacionais.

Um exemplo disto é a avaliação do polinômio Pn(x), de grau n, para o qual se tem que planejar (montar) um algoritmo. O método mais óbvio é usar a funçao potência, que é dada pela figura 3.1.

```
1. INICIO
2.    entre {n, an, an-1, an-2, ..., a1, a0, x};
3.    sum := a0;
4.    PARA I = 1  incr 1 até N
5.      FAÇA  sum := sum +ai * x^I;
6.    escreva {sum};
7. FIM.
```

Fig. 3.1 Algoritmo da avaliação de polinômio por potência

Pode-se notar a avaliação das potências x^1, x^2,..., x^N separadamente, o que poderia ser substituído pela avaliação parcial de potências, sendo mantido para próximo cálculo, como ilustrado na figura 3.2.

```
1. INICIO
2.   entre {n, an, an-1, ..., a1, a0, x };
3.   sum := a0;
4.   xI := x;
5.   PARA I = 1 incr 1 até N
6.     FAÇA INICIO
7.         sum := sum + aI * xI;
8.         xI := xI * x;
9.     FIM;
10.  escreva {sum};
11. FIM.
```

Fig. 3.2 Algoritmo da avaliação de polinômio por produto

O algoritmo da figura 3.2, ainda efetua cálculos desnecessários, pois na n-ésima iteração, ainda executará a linha 8, o que produzirá a potência x^N+1. Outro algoritmo para avaliação do polinômio Pn(x) é o conhecido método de Horner, o qual é dado na figura 3.3.

```
1. INICIO
2.   entre {n, an, an-1, ..., a1, a0, x};
3.   b := an;
4.   PARA I = N-1 incr -1 até 0
5.     FAÇA b := aI + x * b;
6.   escreva { b };
7. FIM.
```

Fig. 3.3 Algoritmo da avaliação do polinômio por Horner

Estes três algoritmos resolvem corretamente a mesma classe de problemas (cálculo de polinômios), mas o algoritmo descrito pela figura 3.3 é mais eficiente, pois exige menos operações para avaliar o polinômio, como é mostrado na tabela 3.1.

Tabela 3.1 Tabela do número de operações para avaliação de polinômio de grau n

algoritmo	multiplicações	adições
potência	N(N-1)/2	N
produto	2N	N
Horner	N	N

3.5 O processo de desenvolvimento do software

Para Johnston, em [JOH82], existem três passos básicos na criação de um software, os quais são:

a) A escolha do método básico a ser usado;

b) O projetar do algoritmo;

c) O produzir do programa como um item do software.

Ele define um método como uma fórmula matemática para achar a solução de um problema particular em estudo. Na escolha do método, é necessário que se tenha uma lista dos possíveis métodos a serem usados, o que implica numa pesquisa na literatura para se catalogar os existentes. Então, uma detalhada análise matemática de cada método necessita ser feita, para levantar princípios básicos, nos quais os métodos possam ser comparados, isto, juntamente com certa experiência computacional, a fim de avaliar o trabalho necessário para implementar cada método,

extensões e comparações analíticas, para determinar o método ou combinação de métodos mais apropriada.

O segundo passo é o da descrição detalhada do processo computacional, que é o **algoritmo**, uma fórmula computacional projetada para resolver o problema com a exatidão quanto maior possível em um computador.

O passo final é o de converter o algoritmo em um item do software. Este item do software é uma realização física do algoritmo, isto é, um **programa** do computador. Esta tarefa de produzir o programa, envolvendo diversas questões importantes e difíceis com respeito a estrutura aritmética usada na máquina, alocação de memória, velocidade dos dispositivos e operações do software. O objetivo deste passo é a plena utilização das capacidades computacionais disponíveis do sistema, de uma maneira mais eficiente possível.

Como se vê, a abordagem de Johnston está de acordo com os níveis de abstração descritos no capítuilo 2 e ilustrados na figura 2.2.

Para Crowell e Fosdick, em [CRO77], o processo de produção de um software inicia com a análise algorítmica e prossegue através da construção do software e documentação, para vários testes e, finalmente, para a distribuição e suporte do produto do software. A exigência e o custo necessários são justificados pela utilidade de alta qualidade do software, pela eficiência do produto na sua distribuição e pelos benefícios providenciados à pesquisa em o aplicar.

Dew e James, em [DEW83], concordam com Crowell e Fosdick nas etapas do desenvolvimento do software. Eles partem,

entretanto, de que, já se tenha os métodos, não incluindo, portanto, a importante etapa da escolha do método básico a ser usado, descrita por Johnston. Eles caracterizam o desenvolvimento do software em cinco etapas, que são: o projeto e análise do algoritmo; a construção do software; uma documentação detalhada; vários testes e a distribuição e manutenção do software.

3.5.1 O projeto e análise de algoritmos

Na etapa do projeto e análise de algoritmos para desenvolvimento de um software, é onde são tomadas as decisões que influenciarão o software no que tange ao seu estilo, propósito e finalidade.

A produção de um software numérico não é uma simples extensão de programação básica e tem que ser bem mais do que um conjunto de programas que resolvem uma classe determinada de problemas. Programas que se pretende para distribuição pública necessitam delimitar o intervalo dos possíveis dados de entrada, de uma performance perante adversidades, compiladores e interpretadores, sistemas operacionais e características de hardware.

Nesta etapa é feita a definição de passos tomados para resolver o problema, são feitos convenções de notação, análises dos algoritmos, o relacionamento e o gerenciamento, compondo com estes elementos a filosofia e o estilo do software.

Esta etapa é uma das mais difíceis, pois, além do que já foi citado, é necessário um estudo dos métodos numéricos teóricos, para a tradução em algoritmos computacionais, o que muitas vezes requer reformulações nos métodos.

É ainda nesta etapa que se adota critérios, os quais são as características desejáveis de um software numérico, as quais são descritas mais adiante. Algumas destas são importante existirem em algoritmos a serem implementados, até para usuários sem maiores conhecimentos, tendo apenas a idéia básica do método. Estas são a **estabilidade**, o ser sensível aos erros de arredondamento, que são inevitáveis nos cálculos com números reais em computadores de precisão finita; a **exatidão**, que é a capacidade de resolver problemas com a exatidão pré-requerida na documentação, resolvendo corretamente a classe de problemas para que foi planejado, finalizando, mesmo em caso de insucesso, com mensagens adequadas. Outra característica é a **eficiência**, pois quando implementado o algoritmo não deve gastar muito tempo ou espaço para armazenar ou resolver problemas.

3.5.2 Construção do Software

A construção do software é a codificação do algoritmo, o verdadeiro código do algoritmo, incorporado no software, resultando num **programa e sua documentação.**

Nesta fase de transformação do algoritmo computacional no programa, o qual deve ter capacidade de detectar e contornar situações anormais, sem interromper a computação. O software deve conter facilidades para monitorar erros, a fim de determinar argumentos impróprios e prever problemas computacionais como underflow e overflow, antes que eles ocorram.

Esta transformação do algoritmo computacional num programa com as características acima, em geral, envolvem mudanças no algoritmo. O algoritmo alterado, então, deve ser analisado para garantir sua confiabilidade, pois uma

implementação imprópria até mesmo de um magnífico algoritmo pode produzir resultados insatisfatórios, comprometendo a "reputação" do algoritmo. Ainda, um algoritmo óbvio não é, necessariamente, o melhor algoritmo e sua implementação em diferentes ambientes, máquinas produzem diferentes comportamentos.

Um esforço considerável é necessário para se construir um bom software, e dar a ênfase em ser capaz de transportar programas bem testados de uma instalação para outra, com mínimas modificações. Isto significa, que os matemáticos computacionais devem resistir a tentação dos programadores de explorar os recursos do sistema ou obscuras características especiais da linguagem de programação. Truques deste tipo produzem programas difíceis de serem compreendidos e podem falhar quando implementados em diferentes sistemas.

Deve-se projetar algoritmos com o propósito da **adaptabilidade** e codificá-los com a idéia de transportar o software de um computador para outro. Por exemplo, um teste do tipo:

" IF ABS(ERRO) < Ø.5E-1Ø THEN ..."

não é adaptável, pois, devido a diferente precisão de cada computador, pode não ser possível obter a exatidão necessária em algumas máquinas. O programa, deste modo, irá se comportar diferentemente em diferentes sistemas, podendo falhar em alguns. Testes de erro devem ser expressos relativamente a precisão de cada computador. Uma melhor consideração para este teste seria definir no inicio do programa uma constante, por exemplo, "mi:=Ø.5E-1Ø", que é a precisão do computador e usar nos testes:

"IF ABS(ERRO) < mi THEN ...".

Assim, só seria necessário modificar na passagem de um sistema para outro as constantes dependentes de máquina.

Portanto, num software numérico, alguns pontos, tais como testes de convergência, previsões de underflow e overflow, refletem o esquema de representação dos números no computador. Estas porções do programa, que são dependentes da máquina, devem ser isoladas, tornando fácil sua alteração quando o programa é transportado para outra máquina.

3.5.3 Uma documentação detalhada

A documentação deve ser feita ao mesmo tempo da construção do software, pois é parte deste produto final. A documentação influência o estilo do programa, o qual deve ser escrito para uma clara exibição da organização lógica da computação, localizando e bem definindo as partes dependentes do sistema

A documentação precisa ser clara e bem pensada; organizada de maneira que permita um usuário ocasional obter informações necessárias facilmente.

A documentação deve ser consistente com as operações do programa, para que não haja surpresas no uso do programa.

Do ponto de vista do usuário, a documentação deve ser detalhada, contendo um enunciado claro da classe de problemas que o software, ou programa está designado a resolver, com todos os casos em que ele é particularmente aplicável; uma definição do tipo e propósito de cada parâmetro do programa; as restrições a sua aplicação; um exemplo rodado de aplicação do programa, pois usuários irão freqüentemente procurar se basear no exemplo, adaptando-o ao caso particular que desejam resolver; deve conter uma avaliação de sua performance, isto é, informações detalhadas sobre velocidade de processamento e exatidão das respostas;

enfim, quaisquer características extras usadas no programa devem ser descritas e as principais seções dos programas devem ser destacadas pelo uso de comentários, dando uma estruturação clara, estando a listagem do programa disponível para quem quiser fazer modificações ou extendê-lo.

Resumindo, a documentação do software é a parte entre o que desenvolve e o usuário do produto e, por isto, deve ser completa, pertinente ao serviço a que se destina, com o nível adequado de detalhes. Quanto a que documentação produzir e como a produzir, será discutido mais adiante, mas deve se ter em mente alguns princípios:

a) Ela deve ser adequada e inteligível;

b) Deve-se testar a documentação do mesmo modo como se testa o próprio software;

c) Tem-se que escrever a documentação de modo a satisfazer seus leitores e usuários, não a você mesmo.

3.5.4 Vários testes

A determinação de testes numéricos nos programas matemáticos é uma tarefa complexa e muito demorada, que tem recebido muita atenção nos últimos anos.

Existem dois objetivos principais ao testar uma rotina. O primeiro é o de estabilizar experimentalmente o grau de correção do algoritmo. Para isto, seleciona-se um conjunto de problemas, chamados algumas vezes de "exercício de código", que são projetados para testar o maior número de possíveis caminhos do programa. O segundo objetivo é o de medir a performance relativa do programa em comparação a outros. Uma vez assegurado que o programa é o algoritmo implementado, pois resolve o

planejado, parte-se para a avaliação da performance. É importante entender que se está avaliando não o método numérico abstrato, mas o algoritmo implementado, o programa.

Quase toda avaliação de performance é feita pelo método de **bateria**, um grande conjunto de problemas para os quais o programa foi feito e preparado para resolver. A performance do programa é medida contra a de programas já existentes, registrando, por exemplo, o tempo requisitado para resolver cada problema, ou se foi convenientemente resolvido e quão exata foi sua solução.

Infelizmente, na maior parte das vezes, é difícil, mesmo após vários testes, dizer se um programa é absolutamente melhor que outro, uma vez que um programa pode resolver bem alguns problemas e outros não tão bem, enquanto que o oposto pode acontecer com outro programa. Esta é a razão de existirem bibliotecas de softwares, contendo vários programas para resolver um mesmo problema.

3.5.5 Distribuição e manutenção do software

Nesta etapa, pode-se justificar todo custo e esforço despendido para o desenvolvimento do software, através da distribuição do software, que pode ocorrer através de sociedades profissionais de suporte, como a "Association for Computing Machinery" (ACM), com seus grupos especiais de interesse em matemática aplicada ou numérica, como SIGMAP e SIGNUM, para os quais são enviados artigos, onde são analisados e divulgados.

A divulgação dos softwares está associada a ênfase que o software incorpora, que podem ser utilidade, explorável na pesquisa ou útil no melhoramento da produção de software.

Os projetos de software, em geral, originam-se do esforço de centros universitários, de laboratórios governamentais e de empresas privadas.

O importante, quando se desenvolve um software, é assumir a responsabilidade da manutenção do produto, incluindo correções de erros, modificações, extensões e o suprir novas necessidades do usuário.

Software numéricos de boa qualidade enfrentam a expectativa de usuários, que tem sido significativamente mudada no último quarto de século, incluindo atitudes de facilidades de uso. Um programa, o qual é o mais rápido e mais exato, poderá ser ignorado, se existir mesmo um número pequeno de impedimentos de uso. Os desenvolvedores de software comercial entenderam isto, mas muitos pesquisadores não, e ainda desprendem grande esforço para produzir um programa encorporando o melhor algoritmo, somente para ver o fruto de seu laborioso trabalho ignorado.

3.6 Características desejáveis de um software numérico

O propósito de desenvolver programas de um software numérico é providênciar ferramentas computacionais úteis para programação científica, mas, para isto, estas rotinas necessitam reunir certas características, critérios considerados essênciais, que são exatidão; eficiência; confiabilidade; validação; facilidade de uso; boa documentação; flexibilidade; modificabilidade; robustez e transportabilidade.

Muitas destas características já foram citadas, exemplificadas nas seções anteriores, mas, para uma melhor formalização e caracterização, serão redefinidas, apesar da

dificuldade, da dependência e interrelacionamento destas características.

3.6.1 Exatidão e eficiência

Apesar de os critérios de um bom software tem sido mudados com a experiência e tecnologia, a exatidão e eficiência têm se mantido.

Exatidão é a capacidade do programa resolver problemas com a exatidão pré-requerida na documentação.

O critério de eficiência mudou com o desenvolvimento. No principio, quando os computadores tinham pouca memória e esta era muito cara, um programa eficiente era um que usava a memória eficientemente, sendo o menor possível. Hoje, a memória está tão disponível e barata, fazendo com que este conceito evoluisse para velocidade de processamento; e as máquinas tornando-se cada vez mais velozes, este componente de eficiência deixou de ser tão importante.

3.6.2 Confiabilidade e validação

Confiabilidade é a característica que garante a consistência da operação do programa com sua documentação, resolvendo corretamente a classe de problemas para que foi planejado. Resolver corretamente refere-se a habilidade do programa de obter resultados numéricos desejados eficientemente. Nos casos em que não conseguir encontrar a solução, deve finalizar com uma mensagem de erro adequada.

Validação é um critério muito importante na produção de um software. O propósito da validação é estabelecer que a rotina

cumprirá seu objetivo. Para isto, de inicio, necessita-se estar claro de seu objetivo, da classe de problemas que podem ser resolvidos e o que se entende pela "solução".

O processo da validação de um programa inclui uma detalhada análise matemática do algoritmo, uma prova de que o programa é uma representação fiel do algoritmo e, finalmente, um extensivo teste com representativo conjunto de problemas para testar a maior parte de possíveis caminhos do programa.

Uma apreciação impróprια do ambiente computacional durante a implementação pode degradar a confiabilidade e validação, restringindo o conjunto de problemas que podem ser resolvidos.

3.6.3 Facilidade de uso

O critério facilidade de uso está associado a ter boa documentação, mas também, quanta informação é requerida ao usuário para o uso de tal programa. Um programa fácil de ser usado é aquele que somente requer a entrada (dados iniciais), definições do problema a ser resolvido. Qualquer análise do problema para o propósito de setar os parâmetros do método é feito automaticamente pelo programa.

Um software fácil de ser usado é o que antecipa todas as opções que todos usuários possam querer usar, através de um controle que seleciona a opção desejada.

3.6.4 Boa Documentaçao

A documentação de um programa deve ser clara, conscisa, fácil de entender, explanando como funciona e como usar. Isto é

útil não sòmente para decidir se, usa-se ou não o programa, como também, assistir no diagnóstico de cada dificuldade proveniente do uso. Ela deve ser organizada de maneira que permita um usuário ocasional obter informações necessárias facilmente.

Mensagens de erro são necessárias diagnosticando o acontecido, e sua gravidade. Um exemplo, numa rotina para resolver equações não lineares, pode não ser possível encontrar uma solução com a precisão requerida pelo usuário, em tal caso, a rotina deve apresentar a melhor solução que pode encontrar, dando o número de iterações e a exatidão contida em tal resultado, mais uma indicação de que não foi conseguido a exatidão requisitada.

A documentação vai além de manuais, incluindo a organização do resultado, de forma a documentar resultados, e ainda, a documentação do código, de forma a ser fácil de ser compreendido para manutenção, atualização, alterações e extensões.

3.6.5 Flexibilidade e modularidade

A característica de flexibilidade está associada a característica de facilidade de uso, permitindo que o usuário modifique valores de parâmetros, condições iniciais ou até mesmo de método de resolução do problema.

A modularidade é outra característica muito importante, e influência o estilo da programação, pois separa as tarefas a serem efetuadas em partes, denominadas **módulos**, o que gera um programa estruturado, facilitando o entendimento e manutenção do software. A modularidade é importante para garantir a transportabilidade de um software, uma vez que existem partes dependentes de máquina que necessitam ser modificadas, adaptadas

ao novo ambiente. Estas partes são isoladas e organizadas de maneira que se possa modificar facilmente; esta parte deve ser bem identificada na documentação.

Dentro do conceito da modularidade, alguns autores introduzem a modificabilidade, que é a facilidade de se modificar e adaptar o software a um problema específico que se quer resolver. Esta questão é abordada por R.C. Bushnell, no artigo denominadado: "User modifiable software" (em [RIC71], seção 5.2, p.59-66).

3.6.6 Robustez

A robustez é um indicador da extensão da rotina. Um método numérico envolve alguns parâmetros, os quais podem assumir valores determinados de acordo com as propriedades básicas do problema particular. A análise do processo é geralmente baseado no comportamento do problema modelo. O comportamento da maioria dos problemas não divergem, na prática, do esperado pelo modelo. Mas existem problemas em que a solução se deteriora, afastando-se do modelo. A extensão deste afastamento é uma medida de robustez, da potencialidade do programa.

Robustez se refere também, a habilidade do programa de contornar dificuldades, sem, necessariamente, interrupção do programa. Quando a interrupção se faz necessária, devido a erros, o software tem que providenciar um diagnóstico preciso do erro e sua gravidade. O problema de underflow é um exemplo. Usualmente, quando ocorre, dá-se uma mensagem de erro e considera-se o valor como zero. Se o underflow influenciar consideravelmente o resultado computado é chamado de "destrutivo", caso contrário é chamado de "não destrutivo". O problema com a mensagem de

underflow é que ela não distingue entre o caso destrutivo e o não destrutivo.

Idealisticamente, deve-se reestruturar o programa de modo a evitar underflow; caso ocorra o underflow não destrutivo, o valor é substituido por zero, e caso ocorra o underflow destrutivo, deve ser dada uma mensagem de erro, diagnosticando precisamente o fato.

3.6.7 Transportabilidade

Transportabilidade envolve os conceitos de **portabilidade e adaptabilidade. Portável** é quando o programa pode ser executado corretamente em um novo ambiente sem nenhuma espécie de mudança, o que é, em geral, idealistico na prática. Se um programa pode ser transportado a outro ambiente, mantendo sua performance, com apenas poucas modificações bem documentadas, ele é **adaptável**, ou **transportável**.

Estas características são referentes a mudanças na performance do programa quando o ambiente da máquina é mudado. Isto é uma questão muito difícel, porque existem muitos fatores, os quais afetam a performance.

Idealisticamente, um software numérico deveria ser completamente portável, mas, devido as diferenças de máquinas e seus ambientes, é praticamente impossível atender este objetivo. Existem, entretanto, algumas medidas que podem ser tomadas para ajudar o grande tratamento da uniformidade da performance.

Uma das causas das diferentes performances é a variedade de arquiteturas de máquinas existentes. O remédio é fazer várias versões de rotina, uma para cada tipo de máquina.

Isto não é portável, no sentido da palavra, é adaptável; mas o é para usuários.

Outra atitude para eliminar diferenças na performance é escrever o programa num dialeto "standard" da linguagem de programação, sem explorar características especiais de implementações dependentes de máquinas.

Outra atitude, ainda, é definir valores que são dependentes da aritmética da máquina no início do programa, como constantes, e no tranporte de uma máquina para outra, estes valores devem ser redefinidos de acordo com a aritmética da máquina, para evitar que se tente calcular exatidões maiores do que se pode calcular.

3.7 Bibliotecas de rotinas, conjunto de programas e software numérico

Após esta formalização das características desejáveis de um software numérico, ainda há a necessidade de uma melhor formalização e distinção de uma biblioteca de rotinas, um conjunto arbitrário de programas e um software numérico. A biblioteca de programas ou rotinas, distingue-se de um conjunto de programas por ser de um "capítulo", de uma área específica escolhida, onde todas as rotinas se mantêm numa mesma filosofia, num estilo e uma convenção de nome de variáveis, de indicadores de erro, de ordenação de parâmetros, e entradas e saídas padrões. E ainda, espera-se que constantes de máquinas e do ambiente, sejam parte integrante desta biblioteca.

A biblioteca se constitui de rotinas, que usuários podem utilizar em seus programas, adaptando-as ao problema que eles desejam resolver. Ela deve ser organizada no código de

máquina de acordo com a freqüência de utilização de cada rotina.

Exemplos de bibliotecas de rotinas de grande relevância são descritas em [CRO77], que são: a **NATS** (National Activity to Test Software), a **NAG** (Numerical Algorithms Group) e a **IMSL** (International Mathematical and Statistical Libraries). Elas foram desenvolvidas através de grandes projetos que custaram milhares de dolares, alguns anos de desenvolvimento e cooperação de universidades, laboratórios governamentais e empresas privadas.

O software numérico se constitui não apenas de rotinas, mas é um programa que gerência várias rotinas numéricas, que resolvem certa classe de problemas.

A abrangência do software numérico é decorrente do tipo de usuário a que ele se destina.

Em geral, o software pode ser caracterizado como **instrucional** (quando é voltado ao ensino) ou **aplicativo**. O software aplicativo, como sugere o nome, é o software voltado a aplicações tanto comerciais, quanto científicas (pesquisa).

3.8 A influência do tipo de usuário no desenvolvimento do software

Os usuários de software influênciam o desenvolvimento de softwares, pois quando se desenvolve um software para um grupo de usuários, há limitações na diversidade de interesses e motivações que podem ser abrangidos, devido a conflitos entre conjuntos de interesses, como: entre a pesquisa e o ensino; entre a pesquisa geral e a pesquisa especializada; e entre usuários de uso geral e analistas numéricos.

Um software instrucional, como uma biblioteca de ensino, requer que métodos tradicionais estejam incluídos, uma vez que são ensinados, enquanto que uma biblioteca de pesquisa de uso geral ou um software aplicativo somente requer os métodos considerados como melhores em certa área. Num software instrucional métodos ruins são necessários para que se possa fazer comparações com os bons. Em softwares aplicativos são necessários resultados com grande exatidão, e calculados otimizadamente, enquanto que, em softwares instrucionais, a visualização de cálculos passo a passo são úteis a compreensão do método e a verificação de sua convergência.

Software aplicativo, com finalidade de pesquisa geral ou pesquisa especializada, gera outro conflito, pois grupos de pesquisa especializada, por causa de sua penetração em áreas numéricas particulares, requerem uma abrangência muito maior nestas áreas do que os de finalidade de pesquisa geral.

O próprio comportamento de analistas numéricos que, em geral, efetuam mudanças de conteúdos, códigos e na própria documentação, costumeiramente faz com que o software tenha características de modularidade, para que seja fácil de fazer alterações e expansões; enquanto que, para usuários de uso geral, que apenas utilizam o software de uma forma "estática", ou seja, o software e a documentação permanecem os mesmos para sempre. Isto requer um software fácil de ser usado, com boa documentação e robusto.

A documentação também deve ser adequada a finalidade do software, tanto em qualidade como em quantidade. Para software aplicativo comercial, J D Lomax em seu livro "Documentação de Software" ([LOM83]), descreve a documentação em

três categorias:

a) Informações preliminares;

b) Documentação do usuário;

c) Documentação de manutenção;

constituindo uma "grande coletânea" de documentação.

Para software aplicativo científico, pode ser simplificado e minimizado em documentação do usuário e documentação do programa.

Para software instrucional, a documentação é ainda compensada com um estilo conversacional nos programas, permitindo o usuário fazer inúmeras variações nos dados e parâmetros de entrada. Desta forma, para um mesmo problema o usuário pode, facilmente, alterar um dos parâmetros e observar o efeito da mudança nos resultados, o que lhe permite um controle do comportamento da resolução.

O estilo conversacional proporciona maior interação do usuário com a máquina, dando a ele uma maior clareza dos passos que estão sendo efetuados no processamento, proporcionando, ainda, maior flexibilidade no uso do software, amenizando o "efeito místico", que era resultante de software "caixa-preta", dando assim, maiores informações.

O estilo conversacional proporciona ainda um software fácil de ser usado, desde que em cada nível de tarefa possa ser caracterizado por telas instrucionais que antecipem todas as possíveis tarefas que todos os usuários possam querer requerer. O programa neste estilo deve ser auto-documentado, para proporcionar esclarecimentos a usuários menos instruídos, e que as perguntas sejam claras e definam bem a resposta, não permitindo ambigüidades. Uma técnica usada é "chave de seleção",

onde na tela estão todas opções possíveis, e o usuário escolhe uma delas, através de uma letra, um número ou uma tecla. Isto também é conhecido como "menu".

Maiores detalhes sobre a documentação de software aplicativo e instrucional serão vistos, mais adiante, na documentação do SINAI-16.

4 MAQUINAS & LINGUAGENS

4.1 INTRODUÇÃO

Muitas vezes professores e profissionais em informática são consultados por alunos e conhecidos com a tradicional pergunta: "Eu quero comprar um microcomputador, qual você me aconselha?". A resposta é sempre dada com perguntas do tipo: "O que você quer fazer com o microcomputador?"; "Que tipo de uso você vai fazer?"; "Que sistema você vai rodar?" e até mesmo, "Quanto você quer gastar?"

Portanto, a primeira coisa a ser definida, antes de se pensar na máquina se adotará, é para que finalidade que será utilizada, e que tipos de processamento serão necessários se fazer. Entre as muitas finalidades a que os microcomputadores estão servindo atualmente, pode-se destacar as aplicações científicas, aplicações comerciais, aplicações administrativas, educativas e instrucionais, divertimentos e jogos e até grandes sistemas.

Em aplicações científicas são necessários cálculos com grande exatidão e velocidade de processamento, bem como, o sistema de ponto-flutuante com grande intervalo de representação. Algumas aplicações científicas exigem gráficos. Já em aplicações administrativas e comerciais, não é necessário o sistema de ponto-flutuante, pois, em geral, os cálculos envolvem a manipulação de dinheiro. Em geral, este tipo de processamento exige processamento de texto e para isto, são convenientes, equipamentos capazes de escrever em português, ou seja, que tenham o alfabeto brasileiro. Também se usa, planilhas

eletrônicas para cálculos administrativos e comerciais. Em aplicações educativas ou divertimentos e jogos, não são necessários nem muita capacidade de cáculo, nem de impressão de relatórios, pois, em geral, as atividades concentram-se no momento de uso, sem muito interesse de documentar o que está sendo feito, sendo suficiente gravar os programas, o que algumas vezes é feito atráves de gravadores e fitas. Para monitor de vídeo, muitas vezes é utilizado o próprio aparelho de televisão existente nas casas.

Uma ilustração real pode ser tomada das instituições financeiras governamentais, que estavam utilizando terminais da Scopus e Burroughs e impressoras, para servir seus clientes com saldos bancários. Estes terminais são "inteligentes" e bastante caros para uso simples, deixando cerca de oitenta porcento de sua capacidade ociosa. A solução foi projetada como "terminais do cliente"(TC), que constam de uma unidade com uma pequena tela, um teclado numérico e uma pequena impressora, capazes de requerer o extrato ou saldo e imprimi-lo.

Portanto, é necessário primeiro definir a finalidade de uso, com suas necessidades, para então procurar a solução mais econômica.

Para um software numérico, que visa ensino no terceiro grau e aplicações cientificas, é necessária uma máquina com uma certa precisão e que produza resultados em exatidão satisfatória, pois o processamento se constituirá predominantemente de cálculos. Os resultados terão que ser impressos, para serem documentados.

Tratando-se, ainda, de um soltware que calcula zeros de funções, que são os pontos que interceptam o eixo x, no caso de

zeros reais, faz-se necessário traçar gráficos e até mesmo imprimi-los. E necessário, ainda, que se tenha a capacidade de definir funções transcendentes; para isto são necessárias as mais usuais como seno, cosseno, exponencial e logaritmica, para que destas se definam qualquer outra.

4.2 Abordagem do problema de máquina

O problema da máquina foi abordado do ângulo das necessidades e finalidades, com uma análise das possibilidades disponiveis na Univercidade Federal do Rio Grande do Sul, mas principalmente no Curso de Pós-graduação em Ciência da Computação e também no Instituto de Informática da Pontificia Universidade Católica do Rio Grande do Sul, onde se conta com vários tipos de equipamentos.

Para cada sistema foram considerados vários itens de identificação, que estão descritos a sequir.

4.2.1 Nome

O nome como o equipamento é conhecido e comercializado no mercado.

4.2.2 Fabricante

O nome da empresa, ou grupo que produz e comercializa o sistema.

4.2.3 Configuração básica

Os componentes que constituem o sistema padrão. Em geral inclui o microprocessador, um terminal que combina teclado

e vídeo monitor, um par de drives de disco, uma impressora e a memória disponível.

4.2.4 Microprocessador

É um pequeno e modesto componente eletrônico, sendo o mais importante elemento do sistema. Ele é responsável pela leitura das instruções que compõem um programa e toma as providências devidas em cada instrução. O tipo de microprocessador caracteriza as famílias de computadores. Alguns exemplos importantes de microprocessadores são o Z80 da Zilog, o 6502, e os 8088 e 8087 da Intel.

4.2.5 Sistema operacional

O sistema operacional gerência e poê para trabalhar a máquina física. Os micros gastam muito tempo transferindo informações entre o microprocessador e os vários outros componentes do sistema. Eles precisam, também, controlar as operações destes outros componentes. Tudo isto é feito por este programa, chamado **sistema operacional**.

Eles podem ser **monousuário**, quando o sistema é usado por somente um usuário, ou **multiusuário**, quando o sistema é compartilhado por vários usuários.

Cada sistema de computador tem seu próprio sistema operacional, sendo entretanto, muitas vezes, cópia ou versão compatível de sistemas operacionais consagradas, entre os quais, destacam-se:

a) CP/M, que é um sistema operacional em disco, produzido por uma companhia denominada Digital Research. CP/M se

deve ao nome em inglês "Control Program/Monitor". Muitas versões do CP/M são encontradas para uma grande variedade de microcomputadores de muitos fabricantes. O CP/M-80 pode ser usado em qualquer microcomputador que utiliza o 8080 ou Z80 como processador central e tenha um sistema de "floppy-disk" (disco flexível) de 5 1/4 polegadas ou 8 polegadas. O CP/M-86, outra versão, pode ser usado em qualquer computador que tenha 8086 ou 8088 como processador central e, também, tenha sistema de "floppy-disk";

b) DOS é outro sistema operacional em disco bastante popular em microcomputadores. O nome vem do inglês "Disk Operation System". Ele controla todas as operações relacionadas com o disco. As linguagens de programação, como BASIC, transmitem pedidos ao DOS para qualquer operação onde o disco esteja envolvido. O DOS devolve o resultado ao BASIC. Para utilização do DOS, É necessário no mínimo 16k de memória e o microprocessador 6502, como no caso dos micros da linha apple;

c) O MP/M é um sistema operacional voltado a multiusuários. O nome também vem do inglês, "Multi-Processing Monitor Control Program".

4.2.6 Capacidade Gráfica

A capacidade gráfica implica em que o microprocessador tenha instruções gráficas; no tipo de monitor de vídeo; e no acesso aos pontos do monitor, através da placa adaptadora gráfico/cores que tem dois principais modos gráficos: o modo de baixa resolução e o modo de alta resolução. Nem todos os computadores têm tela gráfica. Outra característica a ser analisada é se há dispositivos para impressão da tela (chamado

"hard copy"), e impressão de gráficos. Os dispositivos de impressão de gráficos pertencem a três categorias principais; impressora de matriz de pontos, plotadores e câmaras gráficas. Uma impressora de matriz de pontos se presta naturalmente à impressão de imagens gráficas exibidas em um monitor, já que a matriz de pontos da impressora corresponde à matriz de pontos da tela.

4.2.7 Linguagens

Para o desenvolvimento de programas ou a manutenção de um existente, é necessário um processador de linguagens de programação. Este processador converte um programa de computador de uma forma que é fácil para um ser humano escrever, para uma forma que pode ser compreendida pelo computador.

As linguagens de programação são geralmente classificadas como sendo de alto e baixo nível. Uma **linguagem de alto nível** permite que se escreva um programa em uma notação próxima à maneira natural de se expressar o problema que se deseja resolver. Uma **linguagem de baixo nível**, entretanto, exige que se escreva um programa em uma notação que esteja próxima às etapas que o computador deve executar para rodar o programa. As linguagens de alto nível são mais fáceis de se aprender que as linguagens de baixo nível. As linguagens de alto nível tendem a ser mais especializadas.

Os processadores de linguagens de programação da alto nível podem ser divididas em dois tipos: compiladores e interpretadores. Um **compilador** lê um programa escrito na **linguagem-fonte** do compilador e o traduz para a **linguagem de máquina**. Pode-se armazenar o programa compilado em um arquivo de disco e executá-lo a qualquer momento como se fosse um comando do

sistema operacional. Um **interpretador** lê um programa que foi escrito e o executa, uma declaração de programa por vez. Nenhuma fase intermediária de compilação é necessária. Simplesmente, insere-se o programa e o interpretador o executa.

A seguir, relaciona-se algumas das principais linguagens de programação, que estão disponíveis a sistemas de computadores, que são:

a) **BASIC**, a linguagem mais popular dos microcomputadores. É uma linguagem de alto nível desenvolvida para grandes computadores de múltiplos usuários nos anos 60 e que se tornou popular com os primeiros microcomputadores, porque era fácil de implementar em máquinas simples com pequenas quantidades de RAM (Random Access Memory). O BASIC é uma linguagem de propósito geral, apropriada para a maioria dos tipos de programação que se conhece;

b) **PASCAL**, uma outra linguagem popular utilizada em uma variedade de áreas de programação. Ela foi desenvolvida como uma ferramenta para o ensino de bons hábitos de programação, tornando fácil escrever programas estruturados e claros, confiáveis e facilmente modificáveis;

c) **Linguagem C**, está se tornando uma das linguagens de programação mais populares, embora, não seja tão fácil de aprender quanto BASIC. Ela possui muitas características que a tornam geralmente melhor para se escrever programas. C é uma linguagem de "nível médio" . Ela combina o controle extensivo sobre como um programa roda com muitas vantagens das linguagens de alto nível como PASCAL. É usada principalmente para escrever sistemas operacionais e programas altamente técnicos, permitindo

programação concorrente, sendo também utilizável para programação científica e comercial;

d) **FORTRAN**, uma linguagem tradicional, originalmente desenvolvida para resolver aplicações matemáticas em grandes computadores, mas também disponível hoje, em microcomputadores;

e) **COBOL**, é outra linguagem tradicional e muito usada comercialmente, mas sendo sem muito valor científico. Para um programa em COBOL, tem-se que escrever muitas linhas de programa para descrição, para, então, se chegar a um resultado. Como muitos programas foram escritos em COBOL e implantados em empresas, o COBOL continua sobrevivendo;

f) **LOGO**, uma linguagem atualmente muito popular para o ensino de programação a crianças. É uma linguagem fácil de aprender e que permite que os estudantes descubram suas próprias idéias de programação. Ela utiliza uma "tartaruga" gráfica que desenha linhas na tela à medida que se move, baseada em suas instruções de programação;

g) **ADA**, uma linguagem patrocinada pelo Departamento de Defesa dos Estados Unidos. É semelhante ao PASCAL, mas possui regras muito mais rigorosas sobre como escrever programas;

h) **LISP**, uma linguagem voltada a inteligência artificial, o que tem se tornado assunto de pesquisa popular, especialmente entre fabricantes japonêses. Como FORTH e APL, LISP possui um formato de programação não-tradicional, que amendronta muitos usuários iniciantes.

4.2.8 Compatíveis

Uma relação de microcomputadores que são compatíveis com este sistema.

4.2.9 Aplicações eficientes e aplicativos

Neste item, relacionam-se os principais tipos de software que estão disponíveis nos sistemas de microcomputadores. Estes aplicativos são, por exemplo: software para processamento de palavras; folhas eletrônicas; sistemas de gerenciamento de banco de dados; programas de contabilidade; programas de planejamento de projetos; programas gráficos; jogos e entretenimentos e aprendendo com auxílio do computador.

4.2.10 Custo do equipamento

Conforme o mercado, o valor do equipamento era dado em ORTNs, devido a grande variação do preço, mas com o congelamento dos preços e extinção da ORTN, o preço vem sendo dado em OTN, a qual vale Cz$ 106.40 (cruzados). Neste item é dado um custo aproximado em OTNs do equipamento em sua configuração basica.

4.3 Descrição de sistemas

A seguir são descritos os cinco sistemas de microcomputadores, escolhidos e existentes nos centros universitários da UFRGS e da PUC RS. Estes sistemas de computadores são: ED281, CP500, MAXXI, HP85 e I7000PCxt. A descrição é de acordo com os itens da seção 4.2, deste capítulo.

4.3.1 O sistema ED281

O sistema ED281 é fabricado pela EDISA - Eletrônica Digital S/A, do grupo Iochpe. A configuração básica do ED281 é um teclado alfanumérico e numérico reduzido; uma impressora; de 1 a 4 drives de disquete de 8 polegadas e até quatro unidades de vídeo. A memória é de 112 Kbyte, sendo expandível até 448 Kbyte. A impressora pode ser tanto matricial como linear. Ele tem disponibilidade de até quatro unidades de disco "Winchester", com capacidade de até 40 Megabytes.

O microprocesador do ED281 é o Z80-A, de 8 bits. O sistema operacional é o MDOS-MB, compatível com CP/M (monousuário), e o MP/M para vários usuários.

O ED281 não tem disponível modo gráfico, tendo apenas o modo texto, com 24 linhas por tela, com 80 caracteres por linha.

As linguagens disponíveis são: o COBOL-MB, compatível ao COBOL-80; e o BASIC-MB, disponível em duas versões, interpretada e compilada.

O ED281 não tem micros análogos, mas está baseado no microprocessador Z80, sendo compatível com CP/M, o que permite que qualquer programa em cima de CP/M seja executável nele. A dificuldade reside no tamanho da disquete, que é de 8 polegadas e não 5 1/4, como na maioria dos micros que usam CP/M.

O ED281 tem aplicações comprovadas eficientes na indústria, comércio e principalmente em instituições financeiras, onde tem sido utilizado para cadastramento de clientes, controle de saldos e contas correntes. Tem sido utilizado para auxiliar a administração de hotéis e empresas, com serviços administrativos e econômicos.

O custo aproximado do ED281, numa configuração da unidade central, dois videos, 112Kbytes de memória, uma impressora e dois drives de 8 polegadas, é de 2000 OTNs.

4.3.2 O sistema CP500

O sistema CP500 é produzido pela Prológica e tem por configuração básica o teclado, dois drives de 5 1/4 polegadas, um video, a unidade central, memória de 48Kbytes, mais saídas para impressora e gravador cassete. O microprocessador é o Z80-A, com freqüência de 2MHz, sendo de arquitetura de 8 bits.

O sistema operacional do CP500 é conhecido por DOS-500, compativel com o TRSDOS, modelo III; podendo ter, opcionalmente, os sistemas NewDOS e CP/M, mas, para isto, necessita modificações de hardware.

A capacidade gráfica do CP500 é de um monitor de video de 12 polegadas, que na configuração básica é formada por 16 linhas de 64 caracteres de largura cada uma, ou 16 linhas por 32 caracteres duplos, no modo texto. O modo gráfico é de 128 pontos na horizontal por 48 na vertical. Através de uma placa de expansão de gráfico em alta resolução, pode-se obter 502 pontos na horizontal, por 192 na vertical.

As linguagens disponíveis para o CP500 são o BASIC interpretado e o ASSEMBLER do Z80 (linguagem de máquina), que são residentes. Ainda se tem o BASIC extendido, através de disquetes, que inclui comandos de acesso a unidades de disco e tratamento de arquivos.

O TRS80 modelo III é compativel com o CP500, o qual tem sido largamente empregado com finalidades profissionais em

empresas, escritórios e consultórios. Destaca-se ainda o emprego em imobiliárias, para administração de condomínios.

O custo aproximado do CP500 é de 460 OTNs.

4.3.3 O sistema MAXXI

O MAXXI é produzido pela Polymax Informática S/A e tem por configuração básica o teclado alfanumérico, monitor de video "preto-e-branco" ou colorido, memória de 48Kbytes, expandível para 64Kbytes e 128Kbytes; interface para gravador ou drives de disquetes de 5 1/4 polegadas e impressora. O microprocessador é o 6502, operando com freqüência de 1MHz e sendo de arquitetura de 8 bits. O sistema operacional é o DOS, o qual exige no mínimo 16Kbyte de memória, pois ocupa aproximadamente 10Kbytes de memória.

No modo texto, o MAXXI divide a tela em 24 linhas de 40 caracteres cada uma, sendo cada caracter formado numa matriz de 5 por 7 pontos. No modo gráfico de baixa resolução, a tela é dividida em 40 pontos horizontais por 48 verticais, ou 40 horizontais por 40 verticais e com 4 linhas de texto. Na alta resolução se tem 280 horizontais por 192 verticais, ou ainda, 280 horizontais por 160 verticais com 4 linhas de texto. Existe a placa "videx" que amplia a capacidade do vídeo de 40 para 80 colunas no modo texto. Os gráficos podem ser gravados em disquetes e depois impressos em impressoras gráficas como a Mônica.

A linguagem disponível é a Poly Soft BASIC, que possui características básicas do padrão BASIC, além de outras implementações como erros de intervalos, sintaxe, indicados imediatamente quando de sua entrada e a permissão de múltiplos

comandos por linha. Permite inteiros variando de -32 768 até 32 767, e em ponto-flutuante variando de 10 elevado a -38 a 38. Tem comandos gráficos e as funções transcendentes. Existe também, uma placa adaptadora para Z80, com sistema operacional CP/M, o que possibilita ao MAXXI operar com qualquer linguagem baseada em CP/M, como por exemplo o PASCAL.

O MAXXI é totalmente compatível com a Apple II Plus, sendo portanto da família ou linha Apple e compatível, também, com o EXATO (da CCE), Microengenho (da Spectrum), Unitron (da Unitron) e com o D8100 (da Dismac).

É um microcomputador para uso pessoal, podendo ser usado para administração de conta bancária, elaboração de imposto de renda, organização de orçamento doméstico e ensino. Ele tem se mostrado eficiente em aplicações educacionais, em aprendizagem de linguagens, desenvolvimento de programas e cálculos. Verifica-se a sua indicação em centros universitários e escolas.

O MAXXI tem aplicativos como processador de textos, que tem grande aplicação em centros de pesquisa, para emissão de artigos e teses, como em escritórios, para impressão de relatórios. As planilhas eletrônicas também permitem uma aplicação científica em engenharia, cálculos de estruturas e de vigas. A capacidade gráfica do MAXXI tem sido usada para projetar plantas e estruturas de concreto. O MAXXI tem inúmeros jogos eletrônicos, sendo uma fonte de entretenimento.

O custo aproximado do MAXXI, com uma impressora, é de 380 OTNs.

4.3.4 O sistema HP85

O HP85 é fabricado pela Hewlett Packard do Brasil Ltda, e possui uma unidade básica constituída de um vídeo, uma impressora térmica, uma unidade de controle de fitas magnéticas, um teclado alfanumérico, com teclas que podem ser definidas. A memória do HP85 é 48 Kbytes, podendo ser ampliada em módulos de 8Kbytes, até mais 48kbytes. Esta unidade básica é do tamanho de uma máquina de escrever, sendo portanto bastante portátil. O microprocessador é próprio do HP85, sendo de arquitetura de 8 bits. O sistema operacional também é exclusivo dele, sendo integrado ao BASIC, os quais são residentes, ou seja, estão permanentemente disponíveis ao se ligar o equipamento.

A tela de vídeo possui dois modos de operação: alfanumérico e gráfico, de fácil comunicação entre si. A tela alfanumérica tem 16 linhas de 32 caracteres de 5 por 7 pontos cada. Mas sua memória é de 64 linhas, sendo disponíveis pelo "rolar" a tela com as teclas de funções especiais. No modo gráfico, a tela é dividida em 256 pontos horizontais por 192 verticais, num total de 49 152 pontos de acesso individual, que asseguram alta resolução gráfica.

A impressora está incorporada ao gabinete básico. Ela é térmica, com controle ajustável de intensidade de impressão, tendo uma velocidade de impressão de 2 linhas de 32 caracteres, o que resulta em aproximadamente 120 linhas por minuto. Através de uma tecla de função especial é permitida a impressão de gráficos, que são impressos no sentido longitudinal do papel.

A linguagem de programação é o BASIC expandido, o que faz o sistema ser simples e potente. Ele está armazenado permanentemente. É uma linguagem mais voltada a aplicações

técnico-científicas, apresentando um grande conjunto de funções matemáticas, como trigonométricas, instruções gráficas e demais instruções de controle de fluxo de programa. Tem possibilidades de expansão com os módulos auxiliares ROM (Read Only Memory), como o módulo matemático com operações entre matrizes.

Os cálculos são feitos com uma precisão de 12 dígitos na mantissa e três dígitos no expoente, na faixa de -499 a 499. Possui instruções gráficas como SCALE, XAXIS e YAXIS, que definem as unidades e o intervalo de variação dos eixos x e y na tela, desenhando-os com ou sem marcas de escala.

O HP85 não é compatível com nenhum outro equipamento, a não ser com as variações HP85-A e HP85-B.

Como o equipamento é mais voltado a aplicações técnico-científicas, existem pacotes de programas em: estatística geral, incluindo análise de amostragem simples, estatística de teste, distribuição e regressão linear e múltipla; análise financeira; análise de regressão; estatística básica e manipulação de dados; análise de circuitos; programação linear e matemática.

O custo aproximado do HP85 é de 1100 OTNs.

4.3.5 O sistema I7000PCxt

O I7000PCxt é fabricado pela ITAUTEC do Brasil, e se apresenta com a seguinte configuração básica: teclado alfanumérico, numérico e com funções especiais; módulo básico que contém a memória de 128Kbytes, podendo ser extendido a 256, ou 512, ou ainda a 640Kbytes; com duas unidades de disco magnético, os quais podem ser substituídos por disco rígido, do tipo "winchester"; video e impressora. O I7000Pcxt pode operar como

I7000 e como PCxt da IBM, para isto, ele incorpora dois microprocessadores, o Z80-B (de 8 bits) e o 8088-2 (de 16 bits); e ainda há a opção do 8087, que é o processador aritmético (também de 16 bits), que aumenta a velocidade nos cálculos. Como conseqüência, ele pode utilizar dois sistemas operacionais: o SIM/M, que é compatível ao CP/M, e o SIM/DOS,que é compatível ao MS-DOS da Microsoft. O SIM/DOS é um sistema monousuário e monoprogramado para micros de arquitetura de 16 bits.

O vídeo pode trabalhar no modo texto e gráfico. No modo texto, a tela é dividida em 25 linhas por 80 colunas. No modo gráfico, a tela é definida como um grande mapa de pontos de acesso individual, podendo ser:

a) 320 linhas por 200 colunas;

b) 320 linhas por 400 colunas;

c) 640 linhas por 200 colunas;

d) 640 linhas por 400 colunas.

A impressora deve ser gráfica, para possibilitar que, a qualquer momento, se possa imprimir a tela, inclusive gráficos, por "hard copy", usando as teclas "Ctrl Esc".

As linguagens disponíveis ao I7000PCxt são todas as que rodam em CP/M (8 bits), como BASIC, PROLOG, PASCAL e LOGO. Como PCxt, estão disponíveis o BASIC, PASCAL turbo, Linguagem C e ASSEMBLER.

A este equipamento são compatíveis todos os que são compatíveis ao IBM PC, como o NEXUS 1600, MAXXI PCxt e MICROCRAFT xt. Outros compatíveis são descritos em [HOF85] e [SAC85].

Este tipo de equipamento tem cada vez mais alargado suas aplicações, alguns exemplos são em automação de escritório

com processamento de textos e cartas, uma vez que contém processador de palavras capaz de escrever em português; agenda eletrônica e arquivos de escritórios; em sistemas internos de empresas com folhas de pagamento; com sistemas de engenharia; com planilhas eletrônicas; simulação e controle de processos.

O custo aproximado do I7000Pcxt é de 800 OTNs.

Tabela 4.1 Tabela de quadro comparativo de sistemas de computadores

	ED281	CP500	MAXXI	HP85	I7000Pcxt
micropro-cessador	Z80-A	Z80-A	6502	HP	Z80-B 8088-2
bits	8	8	8	8	8 e 16
Sistema Operacional	CP/M MP/M	CP/M 2.2	DOS	HP	SIM/M SIM/DOS
tela gráfica	-	128x48	40x48 280x192	256x192	320x200 320x400 640x200 640x400
modo texto	24x80	16x64	24x40	16x32	25x80
hard copy	-	-	via arquivo	tecla copy	Ctrl Esc
aplicações	comerc. admins.	educac. admins. comerc.	educac. cientif. admins.	tecnic. cientif.	comerc. admins. cientif. educac.
linguagens	basic assembler cobol	basic assembler	basic pascal prolog	basic	basic assembler C e Pascal
custo aproximado	2000 otns	460 otns	380 otns	1100 otns	800 otns

4.4 Conclusões preliminares

Com a descrição destes cinco sistemas de computadores e contrastando com as necessidades de um software numérico aplicativo, que são: de fazer gráficos com alta resolução e imprimí-los; de efetuar cálculos com grande precisão, para que se obtenha uma boa exatidão; de termos um software interativo com o usuário, fornecendo informações e requisitando informações; de que este software esteja disponível ao ensino e pesquisa, e portanto, esteja presente a centros universitários, e considerando o custo do equipamento, conclui-se que:

a) O sistema do MAXXI, é o sistema mais econômico e mais comum nos centros universitários, mas o sistema de ponto-flutuante é de pequena precisão e pouca exatidão, o que ocasiona que, para aplicações científicas existam restrições;

b) O sistema HP85 para aplicações científicas é, sem dúvida, o melhor, mas restringe na popularidade, por ser um equipamento muito caro e de linha exclusiva da Hewlett Packard;

c) Os equipamentos da Edisa e Prológica, para aplicações científicas, sofrem restrição por não terem a capacidade do modo gráfico em alta resolução; e a aritmética e o sistema de ponto-flutuante deixam a desejar;

d) O sistema I7000PCxt aparece como uma solução, pois tem um vídeo gráfico e opera em modo gráfico de alta resolução; há a possibilidade de impressão de gráficos e tela a qualquer momento, desde que se tenha a impressora gráfica; tem capacidade de cálculos e o sistema de ponto-flutuante satisfaz as exigências aplicativas científicas e ainda existe a possibilidade de acoplar o microprocessador aritmético 8087, que efetua os

cálculos cem vezes mais rápido. Além disto tudo, ele é totalmente compatível com o PC da IBM, que está se tornando um dos microcomputadores mais vendidos nos Estados Unidos, introduzindo, assim, a tendência de microcomputadores de arquitetura de palavra de 16 bits, como já se verifica entre os fabricantes do Brasil, que estão lançando seus micros de 16 bits.

Estas considerações e conclusões, acrescidas da disponibilidade de micros de 16 bits em centros universitários, como da UFRGS e da PUC RS, levam a escolhe-los como a solução para se implementar software numérico aplicativo e instrucional.

5 METODOS DE RESOLUÇAO DE EQUAÇOES ALGEBRICAS

5.1 Introdução

Em muitos livros de cálculo numérico e análise numérica, encontram-se diversos métodos de resolução de equações. O tempo vem e vai e surgem novos livros de cálculo numérico, apresentando métodos "novos", que na realidade estão sendo redescobertos. Vê-se, também, a falta de metodologia e de organização nesta área, pois não se tem catálogo, ou lista de métodos existentes, com uma quantidade significativa de informações sobre cada método.

Neste capítulo é proposta uma **ficha instrucional** de caracterização dos métodos de resolução de equações algébricas e transcendentes, com a qual se pretende estabelecer uma abordagem, uma filosofia de estudo de métodos. E ainda descrito, neste capítulo, os métodos implementados no pacote SINAI-16. A ficha de cada um destes métodos é encontrada no Manual Instrucional do Usuário. No apêndice A, encontra-se um exemplo de ficha instrucional para o método de Newton-Raphson.

5.2 A ficha instrucional

A ficha instrucional é composta de vários ítens que, no seu conjunto, constituem todas as informações necessárias para as pessoas que queiram utilizar os métodos de resolução de equações, quer com ênfase da análise numérica, ou com ênfase prática de utilização do método, ou quem quiser implementar alguns métodos, ou ainda,considerações de ênfase computacional.

Os itens que constituem a **ficha instrucional** de um métodos são: Nome, classificação, objetivo, uma descrição, a interpretação geométrica, limitações e restrições do método, o erro no método, a ordem de convergência, os índices de eficiência, o algoritimo, implementações existentes, a listagem do programa, mensagens de erro, Dados de entrada e saída, exemplos, bibliografia e observações.

5.2.1 Nome do método

O nome do método como é conhecido na bibliografia.

5.2.2 Classificação

Quanto a natureza, os métodos podem ser:

a) **Métodos de Quebra**, que são métodos onde a idéia básica consiste de, a partir de um intervalo fechado I:=[a,b], que contenha uma raiz da função f(x) (continua no intervalo), determinar um ponto xi que "quebre" o intervalo, de modo que a raiz pertença a um dos subintervalos formados;

b) **Métodos Iterativos**, eles envolvem o conceito de iteração (aproximação sucessiva), que é um dos conceitos mais importantes usados em métodos numéricos para resolução de diversos problemas matemáticos. Em um método iterativo partimos de um valor inicial xØ, e construímos uma seqüência de aproximações gradualmente precisas, até que se satisfaça algum critério de parada. Se existe convergência, existe limite de seqüência, que será a raiz da equação, no caso de resolução de equações. Um passo do método iterativo pode calcular novo valor para um ou mais pontos, reusando ou não informações anteriores, com o que se pode classificar os métodos iterativos em

unipontuais ou multipontuais, podendo ambos ser com ou sem memória.

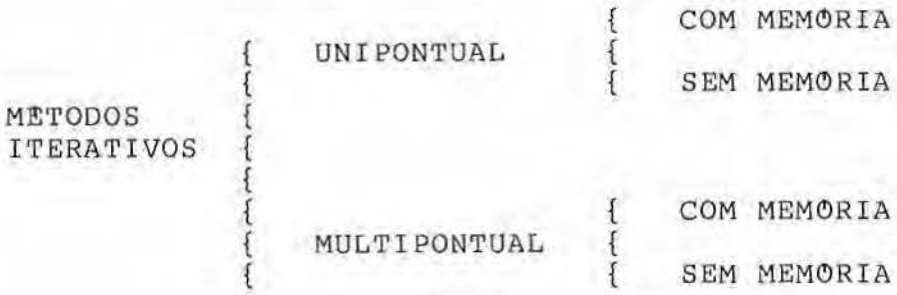


Fig. 5.1 Classificação dos métodos iterativos

c) **Métodos intervalares** são métodos que, em vez de determinar seqüência de números, determinam seqüência de intervalos que convergem para a solução. Para isso, tem-se que ter toda aritimética intervalar;

d) **Métodos hibridos.** Um método hibrido consiste em combinar mais de um método para a determinação de uma raiz. Eles podem ser combinados, ou aplicados isoladamente.

5.2.3 Objetivo

Uma descrição da utilidade do método; em que casos é aconselhável o uso, como o cálculo de raizes reais de um polinômio, ou cálculo de raizes complexas, ou para raizes múltiplas de um polinômio, ou ainda, para o cálculo de raizes reais de equações transcedentes. Isto serve para orientação de um usuario quanto a que método tem que usar para resolver o seu problema.

5.2.4 Descrição

Neste item serão dadas as condições em que o método deverá ou poderá ser usado, contendo um resumo descritivo e as fórmulas utilizadas.

5.2.5 Interpretação geométrica

O gráfico que dá a interpretação do método, para que se tenha o entendimento de como funciona e quando converge o método.

5.2.6 Limitações e restrições

Neste ítem são dados casos de não convergência do método, bem como, situações que alteram o desenpenho normal do método.

5.2.7 Erro do método

A determinação do erro, o cálculo do erro é algo da análise numérica, sendo muitas vezes um trabalho árduo. Neste item, não se tem por finalidade a sua determinação, ou como foi calculado, simplesmente se dará a fórmula do erro , que pode ser encontrado na bibliografia recomendada.

5.2.8 Ordem de convergência

Seja {Xi} := XØ, Xl, ... uma seqüência de valores obtidos a partir de uma função de iteração ø(x), a qual converge para um valor a, tal que f(a)= Ø. Seja ainda, en o erro na iteração Xn, que é en = |Xn - a|.

Se existe uma constante C ≠ Ø,e um número p tal que:

$$en+1 := C\, en^{p} \qquad e \qquad \lim_{n \to \infty} \left\{ \frac{en+1}{en^{p}} \right\} = C$$

então p é chamado de ordem de convergência da seqüência (ou método) e C é a constante assintótica do erro.

Para p = 1, 2, ou 3 a convergência é dita linear, quadrática ou cúbica, respectivamente.

A ordem de convergência é uma medida que indica qual é o ganho em precisão em uma iteração. Supondo C = 1, e que Xn coincide com a solução exata em k dígitos, então Xn+1 se aproxima de a em **p.k dígitos significativos.**

5.2.9 Indice de eficiência

Para se estudar complexidade computacional de métodos, algumas medidas de eficiência de métodos iterativos são apresentadas, com as quais poderá se comparar os diferentes métodos iterativos.

Para melhor caracterização da função de iteração ϕ, e com a finalidade de determinar o custo de uma iteração, introduz-se os parâmetros: e - o número de funções avaliadas por iteração (isto é, f, f', f", ...); m - o número de valores reusados na iteração (isto é, xi-1, xi-2, ...). Seja ainda, p a ordem de convergência vista no item anterior.

Os índices E1 e E2 são conhecidos como o índice de eficiência de TRAUB e o de OSTROWSKI e são dados pelas fórmulas:

$$E1 := p/e \quad , \quad E2 := p^{(1/e)}$$

como é visto em [TRA64].

Estes índices são medidas assintóticas, elas caracterizam um particular método iterativo quando o número de iterações tende ao infinito. Quanto maior o valor de Ei, mais eficiente assintóticamente é o método iterativo. Estas medidas são aplicáveis somente na vizinhança da raiz a, o que resulta, que elas não são sempre adequadas para tratar com problemas

práticos.

Uma medida eficiente ideal e para ser útil à comparação entre métodos deve ser definida em função do número de avaliações de funções e derivadas por interação e o custo destas avaliações; do custo do cálculo da função, que é a combinação dos custos dos elementos da função de iteração; da ordem de convergência p; e do tamanho da constante assintótica do erro.

Em função disto, Feldstein e Firestone, em [FEL69], sugeriram a medida:

$$E3 := p^{(1/H)}, \text{ onde } H := e.(1 + A/B).$$

Sendo p e e a ordem de convergência e o número de funções avaliadas, como já definido. E A é o número de operações aritméticas necessárias em uma iteração, sem incluir a avaliação da função ou derivadas. B é o número das operações requeridas para avaliar a função e derivadas.

Outra medida foi sugerida por Paterson, em [PAT72], como:

$$E4 := (1/M).\log_2 p$$

onde M é o número de divisões e multiplicações requeridas para cálculo da função de iteração.

Já Kung e Traub, em [KUN73], sugeriram a medida E5, como:

$$E5 := \frac{\log_2 p}{\sum ni \, . \, V(f^{<i>}) + c(\emptyset)}$$

onde, ni é o número de avaliações de $f^{<i>}$ usados na função de iteração $\emptyset$; $V(f^{<i>})$ é o número de operações aritméticas

requeridas para uma avaliação de $f^{<i>}$; c(ø) é o número mínimo de operações aritméticas requeridas para combinar $f^{<i>}$ para formar a função de iteração ø.

Existem outras medidas de eficiência, mas o estudo destas medidas foge ao objetivo deste trabalho, podendo entretanto, ser acrescentadas nas fichas instrucionais.

5.2.10 Algoritmo

Deve ser dado um algoritmo em uma linguagem lógica, ou seja, de descrição, caracterizando as etapas da resolução do problema pelo método em questão. Este algoritmo deve conter testes e verificações de convergência, de exatidão e se satisfaz os critérios de parada.

5.2.11 Implementações

Este item diz respeito a uma tabela de referência a implementações existentes, deste método, em equipamentos usuais, contendo o nome do programa, ou rotina e a linguagem de programação que foi implementado.

5.2.12 Listagem do programa

A listagem do programa (fonte) de alguma das implementações que estejam disponíveis nos equipamentos usuais.

5.2.13 Mensagens de erro

Deve ser dada a relação das mensagens de erro, que podem ocorrer neste método, caracterizando a gravidade e, se possível, como contornar.

5.2.14 Dados de entrada e saída

Todas as possíveis informações necessárias para que se utilize o programa, qual resultado e de que forma são emitidos.

5.2.15 Exemplos de utilização do programa

Exemplos que caracterizam a utilização do método, pelo programa da listagem, não só para ilustração e teste do programa, mas também para base de aprendizagem de usuários.

5.2.16 Bibliografia

Consta de uma relação de referências bibliográficas de livros de cálculo numérico, análise numérica e métodos computacionais, que contenham informações do método em questão.

5.2.17 Observações

Nas observações, devem ficar informações que não se encaixem nos itens anteriores, mas que se ache importante referir.

5.3 A escolha dos métodos

A escolha dos métodos deve ser baseada na filosofia e nos objetivos do software que se está desenvolvendo. No SINAI-16, onde se trata de um software com finalidades aplicativas (científicas) e instrucionais, no ensino de cálculo numérico e métodos computacionais, foram escolhidos, inicialmente, os métodos básicos, como o método da bissecção, o método de Newton e o método da Secante. Como se trata de um exemplo de software, que se constitui de uma parte de um software numérico, foram

escolhidos alguns métodos que tratam com o cálculo de raízes múltiplas, como o método de Newton para raízes múltiplas. Para o cálculo de raízes complexas, foram escolhidos a modificação de método de Newton para raízes complexas e o método de Bairstow. Ainda, foram escolhidos exemplos de métodos híbridos, como o método C e também, um novo método denominado MIDREM. Para os tais é dada uma breve descrição, podendo ser encontradas, entretanto, as fichas instrucionais de cada método no Manual Instrucional do Usuário.

5.3.1 Método da Bissecção

O método da Bissecção é um método de quebra e é útil para o cálculo de zeros de funções, desde que para a função f(x), em um dado intervalo J:=[a , b] que contenha uma raiz (simples), ou seja f(a)*f(b) < 0, divide-se o intervalo ao meio e, calcula-se o valor da função neste ponto intermediário xm. Caso xm não seja a raiz, verifica-se em qual dos subintervalos a raiz permaneceu, através do produto de f(a)*f(xm) ser menor ou não que zero, tendo assim outro intervalo menor, para o qual se repete o processo, até que um dos critérios de parada seja satisfeito, ou que se encontre a raiz real.

O método da Bissecção, quando satisfeita a condição inicial, converge, mas esta convergência é muito lenta, ganhando a cada divisão do intervalo um dígito binário de exatidão, o que resulta que tenhamos que fazer 3,3 subdivisões no intervalo para se ganhar um dígito decimal, isto se satisfeito o teorema da vizinhança.

5.3.2 Método de Newton

O método de Newton, também conhecido como Newton-Raphson, que é um método iterativo unipontual sem memória, para cálculo de zeros de funções. A função de iteração ø é dado por:

$$\phi(xn) := xn+1 := xn - \frac{f(xn)}{f'(xn)} \qquad (f'(xn) \neq 0).$$

O método necessita uma estimativa inicial xØ. Em cada iteração são avaliadas a função e a derivada, sendo relevante lembrar, que a derivada é a declividade da reta tangente no ponto e, portanto, a intersecção do eixo x com a reta tangente no ponto xn determina o ponto xn+1.

O método de Newton é de convergência quadrática para raizes simples, perdendo este desempenho para raizes múltiplas.

5.3.3 Método da Secante

O método da Secante é outro método iterativo, é um método unipontual com memória, útil para o cálculo de funções, onde o cálculo da derivada é difícil, pois é uma derivação do método de Newton, onde se substitui a derivada por sua interpretação geométrica. A declividade da reta tangente ao ponto é dado pelo quociente da variação no eixo y pela variação no eixo x, resultando ne função de iteração do método:

$$\phi(xn, xn-1) := Xn+1 := xn - \frac{(xn - xn-1)}{f(xn) - f(xn-1)} * f(xn),$$

para $f(xn) \neq f(xn-1)$. São necessários dois valores iniciais xØ e xl e a cada iteração é feita uma avaliação da função, sendo reusado o valor da iteração anterior, sendo por isto, unipontual com memória. A convergência é super linear.

5.3.4 Método de Newton para raízes múltiplas

Este método é uma modificação do método de Newton. Se conhecido que uma raiz tem multiplicidade m, a função de iteração, ou fórmula da método fica:

$$\phi(xn) := xn+1 := xn - m.\left(\frac{f(xn)}{f'(xn)}\right), \text{ para } f'(xn) \neq 0.$$

Este método pode não convergir quando usado para o cálculo de raízes simples, sendo aconselhado somente quando se conhece que uma raiz é múltipla e qual é sua multiplicidade. A convergência do método é quadrática.

5.3.5 Método híbrido C

O método híbrido C é útil ao cálculo de raízes de polinômios. Sendo f(x) uma função no intervalo X0:=[x0 ,y0], que contenha uma raiz a, e que a raiz cruze o eixo x, ou seja, f(x0)*f(y0) < 0. E ainda, que satisfaça pelo menos uma das condiçoes:

a) f'(x) > 0 e f"(x) > 0;

b) f'(x) > 0 e f"(x) < 0;

c) f'(x) < 0 e f"(x) > 0;

d) f'(x) < 0 e f"(x) < 0. (x pertencendo ao intervalo)

Em resumo, sendo X0 o intervalo inicial que contém uma raiz de f(x)=0, determina-se x1, (a extremidade inicial da próximo intervalo), a partir da intersecção da reta que passa por (x0, f(x0)) e (y0, f(y0)), com o eixo x, (ou seja, a reta secante). A extremidade final do novo intervalo y1 é dada pela intersecção da reta tangente a f(x) que passa pelo ponto (x1, f(x1)) com o eixo x.

Conforme for a concavidade de f(x) em XØ, poderá se ter que o ponto dado pela intersecção da reta que passa por (xØ, f(xØ)) e (yØ, f(yØ)) com o eixo x seja maior que o ponto dado pela intersecção da tangente a f(x) no ponto (xl,f(xl)) com o eixo x. Neste caso, troca-se xl por yl, sendo isto válido em qualquer iteração.

As fórmulas da função de iteração, são dadas por:

$$xi+1 := xi - \frac{(yi - xi)}{f(yi) - f(xi)} * f(xi)$$

$$yi+1 := xi+1 - \frac{f(xi+1)}{f'(xi+1)}$$

Observa-se ainda, se xi+1 > yi+1 então, troca-se xi+1 por yi+1. Se yi+1 > yi então, yi+1:=yi e se xi+1 < xi, então xi+1:=xi.

5.3.6 Método MIDREM

O método MIDREM é um método iterativo que determina a raiz e sua multiplicidade. Podendo ser usado para o cálculo de raizes reais de polinômios, bem como, para raizes complexas de polinômios de coeficientes reais ou complexos.

Para certo valor inicial xØ, calculado segundo a fase inicial, que consiste em:

a) Determinar um círculo centrado na origem, de raio g, que não contenha nenhuma raiz de Pn(x)=Ø, sendo

g:= max {d1, d2, d3}, onde

$$d1 := \frac{1}{\max\left\{ 1 ; \sum_{j=1}^{n} \left|\frac{aj}{a\emptyset}\right| \right\}} ;$$

$$d2 := \frac{1}{\max\left\{ \left|\frac{an}{a0}\right| ; 1+ \max\left\{ \left|\frac{aj}{a0}\right| \right\}\right\}} \quad ; \text{ para } j=1,..,n-1$$

$$d3 := \frac{|a0|}{\sqrt{\sum_{j=0}^{n} |aj|^2}} \quad ;$$

b) Determinar um circulo centrado em g, e de raio R1, que contenha pelo menos uma raiz de Pn(x), onde

$$R1 := \frac{\sqrt{n}}{\sqrt{|S2(g)|}}, \text{ onde } S2(g) := \left[\frac{P'(g)}{Pn(g)}\right]^2 - \left[\frac{P''(g)}{Pn(g)}\right]$$

c) Analogamente a b), determinar um circulo centrado em -g, de raio R2, que contenha pelo menos uma raiz de Pn(x), onde

$$R2 := \frac{\sqrt{n}}{\sqrt{|S2(-g)|}}, \text{ onde } S2(-g) := \left[\frac{P'(-g)}{Pn(-g)}\right]^2 - \left[\frac{P''(-g)}{Pn(-g)}\right]$$

d) A aproximação inicial x0, no caso de raiz real, é dado por:

```
Se R1 <= R2, ENTÃO x0:= g + R1
             SENÃO x0:= -g - R2;
```

Se S2(g) ou S2(-g) forem negativos, implica na existência de pelo menos um par de raizes complexas, e o valor inicial x0, é dado por:

```
SE R1 <= R2, ENTÃO  x0:= (1/2)(g+R1) + g i
             SENÃO  x0:= (-1/2)(g+R2) + g i.
```

O valor inicial pode ser calculado por outras fórmulas, as quais são amplamente estudadas e comparadas em [MAR82], mas, no caso do MIDREM, é aconselhável utilizar a fase inicial, pois produz a aproximação inicial mais próxima da raiz de menor módulo, trazendo vantagens ao método. A fase inicial

pode ser utilizada para determinar o valor inicial de outros métodos, como o método de Newton. Em métodos, como o da Secante, que são necessários dois valores iniciais, a fase inicial também pode ser usada, para isto, tomar-se-ia:

```
SE R1 <= R2,  ENTÂO x0:= g e x1:= g + R1
              SENÂO x0:= -g-R2 e x1:= -g
```

No método MIDREM, uma vez determinado o valor inicial x0, calcula-se uma aproximação inicial q0 da multiplicidade m desta raiz, através da fórmula:

$$qi := \frac{P'(xi)^2}{P'(xi)^2 - Pn(xi)*P''(xi)}, \text{ para } i=0,1,2,...$$

Então, calcula-se nova aproximação da raiz através da função de iteração:

$$xi+1 := xi - \frac{Pn(xi) * P'(xi)}{P'(xi)^2 - Pn(xi)*P''(xi)}, \text{ para } i=0,1,2,...$$

repete-se estes passos até que xi+1 esteja suficientemente perto da raiz de Pn(x). A multiplicidade desta raiz, é dado pelo maior inteiro que contém (qi+1 + 0.5).

Uma vez determinada uma raiz a e suma multiplicidade m, deflaciona-se esta raiz de Pn(x), obtendo um polinômio reduzido de grau n-m, caso isto seja maior que dois, reaplica-se o processo de determinação de raiz e multiplicidade, iniciando novamente na fase inicial para o polinômio Pn-m(x). Caso contrário, resolve-se a equação linear ou quadrática resultante algébricamente. Desta forma pode-se calcular todas as raizes do polinômio.

5.3.7 Método de Newton para raízes complexas

Este método é utilizado para resolver polinômios com coeficientes reais, mas que tenham raízes complexas. Estas são sempre em um número par, pois para cada raiz complexa o seu conjugado complexo, será também, uma raiz. Assim, tão logo se ache uma raiz z:=<z1+z2i> (onde i:= $\sqrt{-1}$; sabe-se também, que uma outra raiz será z':=<z1-z2i>.

A fórmula do método de Newton é dada por:

$$z_{k+1} := z_k + \frac{Pn(z_k)}{P'(z_k)}, \quad \text{onde } z_k := z1_k + z2_k\, i \quad \text{e} \quad P'(z_k) \neq 0.$$

Como se tem que Pn(x) e P'(x) são polinômios de coeficientes reais, se a estimativa inicial z0 for complexa, então as novas estimativas serão também complexas, podendo desta maneira, pela fórmula descrita acima, calcular a raiz complexa, se houver convergência.

Sendo z e z' um par de raízes conjugadas, (x-z) e (x-z') são fatores de Pn(x), pode-se então, deflacionar o produto, que corresponde ao fator quadrático $X^2 - 2z1x + (z1^2 + z2^2)$, rebaixando o grau do polinômio para n-2, podendo assim, reiniciar o processo para cálculo de novas raízes.

Resumindo o método, têm-se a escolha de uma estimativa inicial complexa z0 := <z01 + z02 i>, que pode ser fornecida pela Fase inicial, quando no caso de identificação de complexas ou por uma estimativa através do auxílio do esboço gráfico. Para o cálculo de Pn(x), se não houver disponível a aritmética complexa, pode-se utilizar o algoritmo de avaliação de Horner para o fator quadrático $x^2 - px - q$, onde p= 2z1, q= $-(z1^2 + z2^2)$, e o raciocínio é ilustrado pela figura 5.2.

	an	an-1	an-2	an-3	...	a3	a2	a1	a0
		+	+	+		+	+	+	+
p		bnp	bn-1p	bn-2p	...	b4p	b3p	b2p	Rp
			+	+		+	+	+	+
q			bnq	bn-1q	...	b5q	b4q	b3q	b2q
	bn	bn-1	bn-2	bn-3	...	b3	b2	R	S

Fig. 5.2 Esquema do algorítmo de Horner de fator quadrático

Os valores R e S são usados para o cálculo de Pn(zØ) e P'(zØ) que são dados por:

$$Pn(z\emptyset) := -R.\langle z\emptyset 1 - z\emptyset 2\ i\rangle + S = (-Rz\emptyset 1 + S) + (Rz\emptyset 2)i$$

$$P'(z\emptyset) := R + 2.z\emptyset 2\ i\ .Pn-2(z\emptyset)$$

onde Pn-2(zØ) é calculado seguindo o procedimento iniciado na figura 5.2 e mostrado pelo esquema na figura 5.3,

	bn	bn-1	bn-2	bn-3	...	b3	b2	R	S
		+	+	+		+	+		
p		cnp	cn-1p	cn-2p	...	c4p	R'p		
			+	+		+	+		
q			cnq	cn-1q	...	c5q	c4q		
	cn	cn-1	cn-2	cn-3	...	R'	S'		

Fig. 5.3 Esquema do algorítmo de HOrner para cálculo da derivada

sendo Pn-2(zØ) dado por:

$$Pn-2(z\emptyset) := -R'.\langle z\emptyset 1 - z\emptyset 2\ i\rangle + S' = (-R'.z\emptyset 1 + S') + (R'z\emptyset 2)\ i.$$

O cálculo da aproximação z1 :=<z11 +z12i>, é dado por: z1 := zØ - Pn(zØ) / P'(zØ), para isto, efetua-se as operações complexas, que são definidas para dois números complexos z e z' por: (z = a+bi z'= c+di)

$z + z' := (a+c) + (b+d)\ i$ — adição

$z - z' := (a-c) + (b-d)\ i$ — subtração

$z\ .\ z' := (ac-bd) + (ad-bc)i$ — multiplicação

$$\frac{z}{z'} := \left(\frac{ac+bd}{c^2+d^2}\right) + \left(\frac{bc-ad}{c^2+d^2}\right) i \qquad \text{divisão}$$

O mesmo procedimento é efetuado para o cálculo das demais iterações, até que seja satisfeito um dos critérios de parada.

5.3.8 Método de Bairstow

O método de Bairstow é um método mais complexo, podendo ser usado para o cálculo de raizes complexas. O método consiste em achar um fator de segundo grau, do tipo $x^2 - px - q$, que divida Pn(x). Resolvendo este fator, pode-se calcular as duas raizes reais. No caso de cálculo de raiz complexa do tipo $z=<z1+z2i>$, para z1 e z2 reais, têm-se que $P:=2z1$, $q:=-(z1^2+z2^2)$, e a avaliação de Pn(x) é dado por $-R.<z1-z2i>+S$.

Portanto o método consiste em, dado o polinômio Pn(x) de grau n com coeficientes reais, encontrar um fator $x^2 - px - q$, tal que:

$$Pn(x) = (x^2 - px - q)(bn-2x^{n-2} + \ldots + b1x + b\emptyset) + R(p,q)(x-p) + S(p,q),$$

mas para que seja um fator, o resto deve ser zero, e portanto, têm-se o sistema de equações:

$$\begin{cases} R(p,q) = \emptyset \\ S(p,q) = \emptyset \end{cases}$$

o qual pode ser resolvido através de:

$$pk+1 := pk - \frac{R.Sq - Rq.S}{Rp.Sq - Rq.Sp}$$

$$qk+1 := qk - \frac{S.Rp - R.Sp}{Rp.Sq - Rq.Sp}$$

O cálculo de R, Rp, Rq, S, Sp e Sq são calculados em função de cada pk e qk, ou seja, em cada iteração são calculados segundo o esquema da figura 5.4.

	an	an-1	an-2	an-3	...	a3	a2	a1	a0
		+	+	+		+	+	+	+
pk		bnpk	bn-1pk	bn-2pk	...	b4pk	b3pk	b2pk	Rpk
			+	+		+	+	+	+
qk			bnqk	bn-1qk	...	b5qk	b4qk	b3qk	b2qk
	bn	bn-1	bn-2	bn-3	...	b3	b2	R	S
		+	+	+		+	+	+	
pk		cnpk	cn-1pk	cn-2pk	...	c4pk	Rqpk	Rppk	
			+	+		+	+	+	
qk			cnqk	cn-1qk	...	c5qk	c4qk	Rqqk	
	cn	cn-1	cn-2	cn-3	...	Rq	Rp=Sq	Sp	

Fig. 5.4 Esquema do algoritmo de Bairstow

As iteraçoes terminam quando algum dos critérios de parada for satisfeito, como, por exemplo, o número de algarismos significativos corretos entre duas aproximações de p e q sejam maior do que um certo número D de dígitos (ASC(pk+1,pk) > D & ASC(qk+1,qk) > D).

Através da equação quadrática x -px-q, calcula-se algebricamente as duas raízes reais, ou através da relação de p e q com os termos z1 e z2 da raiz complexa, calcula-se a raiz complexa. A conjugada fica juntamente determinada, podendo,assim, deflacioná-las reduzindo o grau do polinômio e calcular as demais raízes reaplicando o algoritmo.

6 SINAI - 16 UM EXEMPLO DE IMPLEMENTAÇÃO

6.1 Introdução

O SINIAI-16, Software Interativo Numérico Aplicativo e Instrucional em micros de 16 bits, é parte de um software numérico proposto neste trabalho, para se tornar uma ferramenta útil a pessoas que tratam com problemas numéricos, tanto nas Ciências, Engenharia, Informática como no Ensino de terceiro grau.

Este software trata o problema de resolução de equações, em especial, calculando raízes de polinômios; tratando inclusive o problema de cálculo de raízes múltiplas e complexas. Este software proporciona, ainda, a visualização gráfica da função, com o qual se pode ter uma idéia do tipo das raízes, da multiplicidade das raízes e, ainda, separar raízes. Pode-se localizar as raízes através do cálculo de cotas de Cauchy e Fase Inicial, proporcionando uma boa estimativa inicial para métodos iterativos. E por fim, este software permite que se deflacione as raízes calculadas.

6.2 Ambiente computacional

O ambiente computacional para que o SINAI-16 possa ser utilizado são o microcomputador I7000PCxt, (compatível ao PC da IBM) ou qualquer compatível, que tenha como unidade central de processamento o microprocessador 8088-2 da Intel, de 16 bits e relógio de 8Mhz. De memória, o módulo básico contém 128Kbytes, tendo, entretanto, o equipamento do LAPRO da PUC, onde foi implementado, 256Kbytes. O SINAI-16 ocupa xx xxxbytes de memória. O sistema operacional SIM/DOS, compatível ao MS/DOS da Microsoft.

É necessário, ainda, um monitor de vídeo e no mínimo uma unidade de disco, para disquete de 5 1/4 polegadas. Se houver impressora gráfica, através de "hard copy" poderão ser impressos gráficos. O SINAI-16 foi programado em BASIC, com o interpretador BASICA, que é a versão 1.14, COMPAQ Personal Computer Basic, que ocupa 54 304 bytes de memória.

6.3 Finalidades

As finalidades do SINAI-16 são, como induzido no nome, Aplicativas e Instrucionais. A finalidade aplicativa destina-se ao uso científico, em centros universitários para cientistas, ou mesmo em firmas de engenharia, que se utilizem deste tipo de problema. Esta dualidade de finalidades está presente e influência desde a escolha dos métodos a serem implementados, na forma que foram implementados, utilizando dupla precisão, para obter melhor exatidão nos resultados e até a forma como o resultado é obtido, que pode tanto produzir o resultado, quanto mostrar os valores intermediários principais, em forma de tabela para propiciar o entendimento e a compreensão dos métodos.

Estas finalidades também estão presentes na documentação, onde há informações para diferentes tipos de usuários.

O projeto deste software, ou a implementação desta parte, destinou-se como ilustração, ou uma amostra do que pode ser feito na Matemática Computacional, para o desenvolvimento de softwares numéricos aplicativos, instrucionais, de uso geral ou especialista.

6.4 Estilo

O software SINAI-16 é interativo, com estilo conversacional e, portanto, de grande flexibilidade no uso deste pacote. Transfere-se ao usuário o controle da execução da solução do problema, sem perda da característica de fácil de ser usado, proporcionando a qualquer momento, troca de valores, correções de valores para que seja analisada a influência de tais mudanças nos resultados, incentivando a pesquisa. O usuário pode também, determinar a quantidade de informações que deseja no resultado, influênciando, com isto, a velocidade de resoluçao do problema.

A documentação é adequada ao estilo, tendo as opções de uso, antecipadas por menus, que compõem as telas instrucionais, as quais facilitam o uso do software, pois não há necessidade de conhecimento prévio para utilização do software. Além disto, os manuais são organizados de modo a conceder a informação desejada, por qualquer usuário, a qualquer momento.

O estilo conversacional do SINAI-16 proporciona uma maior interação entre o usuário e o computador.

6.5 Métodos implementados

Os métodos implementados nesta seção visaram atender as finalidades aplicativas e instrucionais, incluindo os métodos que são usualmente ensinados nos cursos de cálculo numérico e métodos computacionais, bem como métodos novos ou mais eficientes e complexos que têm seu valor cientifico.

A seguir, segue a relação dos métodos e funções implementados no SINAI-16.

6.5.1 Capacidade gráfica

Traçar gráficos de polinômios de grau menor ou igual a 20 com intervalo de definição variável, capaz de auxiliar a determinação da enumeração e separação de raízes.

6.5.2 Cálculo de cotas

Calcular cotas, ou intervalos que contenham todas as raízes da equação, além de proporcionar boas estimativas de valores iniciais para métodos iterativos. As cotas implementadas são: a de **Cauchy**, que, em geral, está próxima a maior raiz em módulo; e a **Fase Inicial**, que determina um círculo que contém no mínimo uma raiz, estimando um valor inicial próximo a menor em módulo, além de identificar a existência de raízes complexas.

6.5.3 Cálculo de raízes reais de equações

Os métodos implementados foram:

a) Método da Bissecção;

b) Método de Newton-Raphson;

c) Método da Secante;

d) Método de Newton para raízes complexas;

e) Método Híbrido C;

f) Método MIDREM.

6.5.4 Cálculo de raízes complexas de equações

Os métodos implementados para o cálculo de raízes complexas foram:

a) Método de Newton para raízes complexas;

b) Método de Bairstow;

6.5.5 Deflacionar de raízes calculadas

As raízes calculadas reais ou complexas podem ser deflacionadas, reduzindo o grau do polinômio, através do algoritmo de Briot-Ruffini.

6.6 Características desejáveis presentes

O SINAI-16 foi implementado no I7000PCxt da PUC, apesar das dificuldades encontradas neste equipamento, principalmente por falta de suporte. Este software é transportável a qualquer de seus compatíveis. As partes dependentes de máquina, como constantes de máquina, encontram-se isoladas no inicio do programa.

As características de exatidão e eficiência também estão presentes, pois os cálculos são feitos em dupla precisão, para se obter a melhor exatidão. Ressalta-se ainda que, para este tipo de equipamento, existe o microprocessador aritmético, que efetua com muito maior velocidade os cálculos, podendo melhorar mais a performance do software.

Este software é um pacote muito simples de ser utilizado, pois possui telas instrucionais em vários níveis, orientando a execução através de perguntas não ambíguas, que antecipam as possibilidades de respostas, para orientação do usuário. O software não requer conhecimentos de computação, nem de matemática para sua utilização, sendo que pessoas com estes conhecimentos poderão extrapolar e tirar mais vantagens. É um software bastante flexível.

A característica da modularidade é obtida parcialmente, pelo fato de que cada função ou tarefa é feita como uma

subrotina. Isto facilita a manutenção do software e permite que ele seja extendido, colocando novos métodos, como nova subrotina, sem perda de sua performance.

Em cada método são testados as situações anormais, antes que elas aconteçam, e são dadas mensagens de erro (ou orientação), devolvendo o controle ao usuário, fortalecendo ainda mais o aspecto interativo usuário-máquina. O usuário poderá fazer alterações nos dados, até mesmo, utilizar-se da capacidade gráfica do software para seus estudos.

Este software tem uma boa documentação, em dois níveis. No nível de execução e utilização e no nível de suporte, com os manuais.

6.7 Documentação

Por ser um software conversacional, o SINAI-16 é auto-documentado, contendo em seções principais telas de informações e de auxílio, descrevendo as funções implementadas, para orientar ao usuário no momento da execução e utilização do programa, bem como para instruir o usuário, suprindo assim, as finalidades aplicativas e instrucionais.

Além disto, o SINAI-16 possui dois manuais: o **Manual Instrucional do Usuário**, que contém um conjunto completo de informações do ambiente computacional, de instalação, organização e utilização do software. Neste último item, destacam-se as fichas instrucionais que, além de exemplos para ilustrar a utilização do software, contém todas as informações para pessoas que queiram utilizar métodos de resolução de equações tanto com ênfase da análise numérica; da prática de utilização do método; ou da ênfase computacional. O outro manual é denominado **Manual de**

Manutenção do Programa, que, além de um conjunto básico comum ao manual instrucional do usuário, contém informações para garantir a transportabilidade do software a outro equipamento, incluindo a listagem completa do software. Os itens destes manuais são encontrados nos apêndices B e C.

7 CONCLUSOES E SUGESTOES

7.1 Conclusões

Considerando o trabalho de pesquisa realizado, sobre o ambiente matemático-computacional, conclui-se que o capítulo intitulado **Aspectos matemáticos** se constitui de um instrumento válido para os novos cursos de bacharelado em Ciência da Computação ou Informática, bem como para pessoas interessadas a ingressarem no estudo da Matemática Numérica.

Já o capítulo **Software numérico aplicativo e instrucional** é um encorajamento ao desenvolvimento de bons softwares, pois além da descrição do processo de desenvolvimento de software numérico; da caracterização das finalidades e estilo de software; reune as principais características desejáveis de um software numérico, proporcionando com isto, subsídios e orientação para o desenvolvimento de softwares.

Neste trabalho, procura-se orientar a abordagem que possíveis usuários de computadores devem ter ao analisar seus problemas, com o tipo de equipamento, ou sistema de computador, capaz de lhe auxiliar na solução de seus problemas. São ainda, caracterizado alguns sistemas de computador e algumas das linguagens de programação disponíveis a eles.

Outra contribuição deste trabalho é a metodologia apresentada para o estudo de métodos de resolução de equações, que não só permite um estudo abrangente dos métodos existentes, como também poderá proporcionar um novo impulso no desenvolvimento de métodos híbridos. Ressalta-se ainda que o levantamento bibliográfico, na literatura disponível, será de grande ajuda a comunidade que se utiliza destes métodos.

O exemplo de implementação não só é útil como ilustração deste trabalho, mas por se ter, mesmo que uma pequena parte, um software capaz de se tornar uma ferramenta útil tanto ao ensino de Cálculo Numérico, Métodos Computacionais e Análise Computacional; como a pesquisadores que venham a necessitar solucionar equações algébricas.

7.2 Sugestões

Como sugestões deste trabalho, estão: (a) a implantação de um projeto para o desenvolvimento de um software numérico, com as características desejáveis descritas neste trabalho, abrangendo alguns dos tópicos afins da Matemática Numérica, para que venha ser útil a comunidade científica; (b) a implantação de um projeto para catalogação de métodos numéricos, através de fichas instrucionais abrangentes, iniciando com resolução de equações e, ainda, resolução de sistemas de equações, ajustamento e interpolação e de equações diferenciais.

APÊNDICE A

S I N A I - 1 6
Ficha Instrucional do método
Resolução de Equações Algébricas

1. METODO: Newton-Raphson

2. CLASSIFICAÇÃO:

... Método de Quebra
.X. Método Iterativo
... Método Intervalar
... Método Híbrido

.X. Unipontual SEM memória
... Unipontual com memória
... Multipontual sem memória
... Muitipontual com memória

3. OBJETIVOS:

.X. Cálculo de raízes reais de polinômios
.X. Cálculo de raízes reais de equações transcendentes
... Cálculo de raízes complexas
... Cálculo de raízes múltiplas

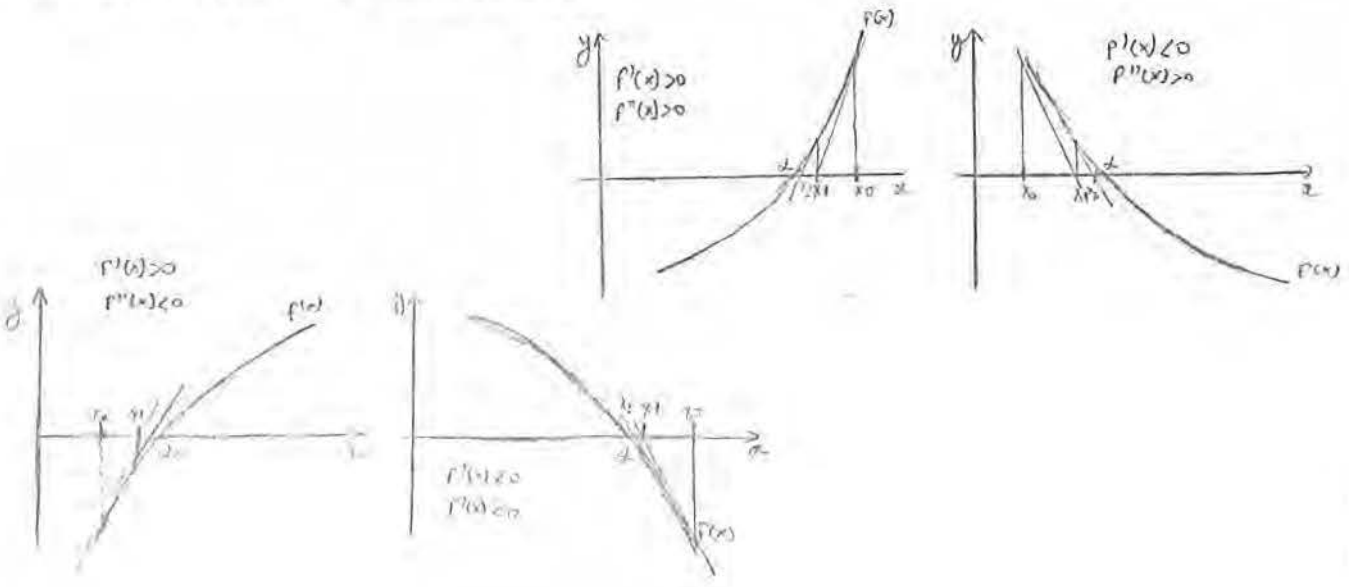
4. DESCRIÇÃO:

O método de Newton-Raphson é essensialmente um método iterativo, onde a função de iteração do método $\phi(x)$ é dada pela fórmula:

$$\phi(xn) := xn+1 := xn - f(xn)/f'(xn), \text{ com } f'(xn) \neq 0$$

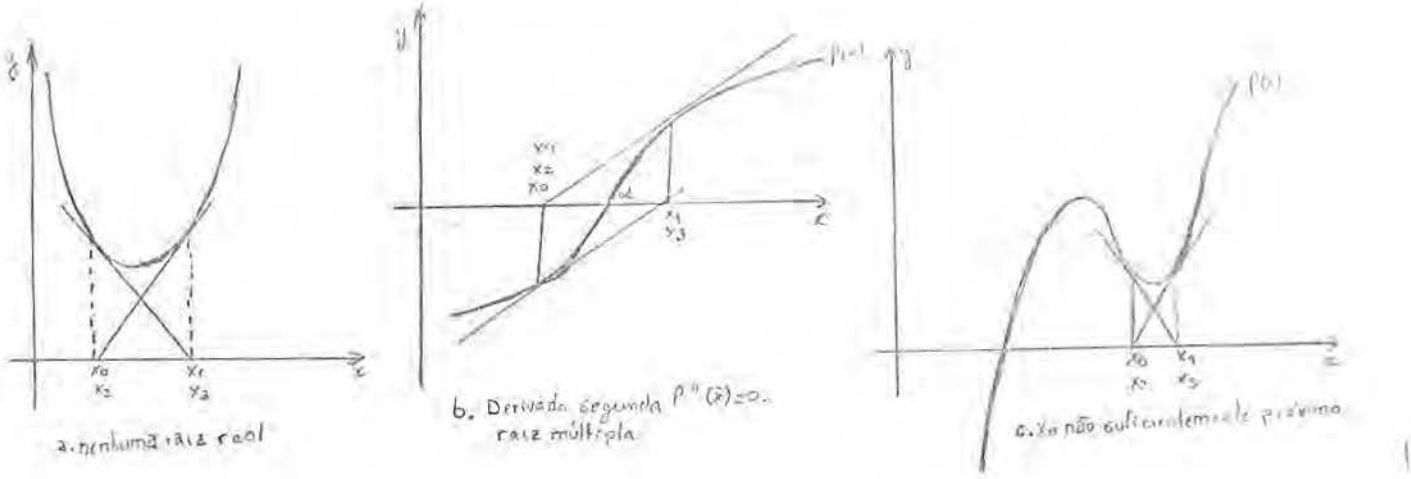
O método de Newton necessita uma estimativa inicial x0. Em cada iteração são avaliadas a função e a derivada. É bom lembrar que a derivada é a declividade da reta tangente no ponto, e portanto, a intersecção do eixo x, com a reta tangente no ponto xn, determina o ponto xn+1.

5. INTERPRETAÇÃO GEOMÉTRICA:



6. LIMITAÇÕES E RESTRIÇÕES:

Algumas vezes o método de Newton não converge, isto não ocorre somente quando f'(x)=0; mas o método não converge e fica oscilando indefinidamente. Estes casos são quando não há raiz real, ou quando há simetria de f(x) em torno da raiz a, que implica em multiplicidade impar, maior ou igual a três; ou ainda, a estimativa inicial x0 estiver tão distante da raiz, que certa parte da função "prenda" a iteração, (devido a existência de mínimo local positivo ou máximo local negativo), como mostrado nas figuras abaixo:



Para que haja convergência, tem-se ainda que, f"(x) não se torne excessivamente grande e que f'(x) muito pequena, próximo de zero.

7. ERRO DO MÉTODO:

$$e_{n+1} := \frac{f''(c)}{2.f'(xn)} * e_n^2$$, onde c está entre xn e xn+1

8. ORDEM DE CONVERGÊNCIA:

O método de Newton é de convergência quadrática, ou seja p=2 para raízes simples, perdendo este desempenho para raízes múltiplas.(Isto dentro de certa proximidade da raiz).

9. INDICE DE EFICIÊNCIA:

Indice de		valor
TRAUB	E1	1.ØØØ
OSTROWSKI	E2	1.414
FELDSTEIN	E3	1.414
PATERSON	E4	1/2 . n
KUNG-TRAUB	E5	1/4 . n

10. ALGORITMO:

```
INICIO
   K:=Ø                              (k  - número de iterações)
   LEIA( xØ , L, D )                 (xØ - valor inicial,
                                      L  - limite de iterações,
                                      D  - digitos corretos desejados)
   ENQUANTO ( ASC<D ou K≠L ou FLK<min)     (min=1Ø^-(D+1))
   FAÇA INICIO
          K:=K+1;
          FK:=f(xk);
          FLK:=f'(xk);
          SE FLK ≠ Ø
           ENTÃO xk+1:= xk - FK/FLK
                 ASC;
   FIM;
   SE FLK<min              (f'(xk)=Ø)
    ENTÃO ESCREVA ( Xk, FK, " derivada é zero")
          ESCREVA ("não dá para usar Newton com este valor");

    SENÃO  SE ASC < D
            ENTÃO  ESCREVA ( Xk+1, ASC, K, "não convergiu a",D)
            SENÃO  ESCREVA (Xk+1, ASC, K);
FIM.
```

11. IMPLEMENTAÇÕES:

Equipamento	Nome do Programa	Linguagem
HP85	TIAPOLI	BASIC
LABO8Ø37	LMCREAT	PASCAL
MAXXI	NEWTON	BASIC
I7ØØØPCxt	SINAI-16	BASIC

12. LISTAGEM DO PROGRAMA:

Veja no Manual de Manutenção do Programa do SINAI-16.

13. MENSAGENS DE ERRO:

Xk, FK, "derivada é zero" ; "não dá para usar Newton com este valor"; A derivada primeira é nula, e portanto o método não pode ser aplicado, por ocasionar divisão por zero. Se isto ocorreu com valor inicial, tem-se que entrar com outro valor inicial, mas se ocorreu durante o processo, terá que ser utilizado outro método .

Xk+1, ASC, K, "não convergiu a",D; O método não conseguiu produzir a solução com a exatidão de D algarismos significativos corretos em K iterações. É dada a melhor aproximação obtida, com sua respectiva exatidão.

14. DADOS DE ENTRADA E SAIDA:

ENTRADA: x0 - valor inicial
L - Limite de iteração
D - Dígitos corretos desejados

SAIDA: raiz procurada Xk+1
Algarismos Significativos corretos no resultado ASC
Número de iterações calculados

Saida Opcional:
Uma tabela contendo os principais valores parciais, para uma boa compreensão do método, contendo o número de iterações, a aproximação xk, o valor da função na aproximação f(xk), o valor da derivada na aproximação f'(xk) e o valor da ASC, algarismos significativos corretos.

15. EXEMPLOS DE EXECUÇÃO:

Calcular a raiz do polinômio $Pn(x)= x^4 +2x^3 -7.5x^2 -20x-11$ com o valor inicial x0=-0.5, com no mínimo 5 algarismos significativos corretos (ASC>5), e com x0= 3.0 e asc > 5.

No inicio do programa:

```
DEFINIÇAO DA FUNÇAO
3    (alterar função)

Entre com o grau do Polinômio (1 a 20):? 4

Entre com os coeficientes an...a0

a4 : ? 1
a3 : ? 2
a2 : ? -7.5
a1 : ? -20
a0 : ? -11
```

```
     Definição da função

     Função atual          1.0000000 x4
                           2.0000000 x3
                          -7.5000000 x2
                         -20.0000000 x1
                         -11.0000000 x0

     1. Função Correta

     2. Alterar Coeficientes

     3. Alterar Função

1 (Função Correta)

          Resolução de Equações Polinômiais

     1. Definição da função
     2. Gráfico da funçao
     3. Cálculos das Cotas
     4. Cálculo das raizes reais
     5. Cálculo das raizes Complexas
     6. Deflacionar raizes
     7. Voltar a tela-chave de primeiro nivel

4 (Cálculo das raizes reais)

          Cálculo das raizes reais

     A. Método da Bissecção
     B. Método de Newton
     C. Método da Secante
     D. Método de Newton para raizes múltiplas
     E. Método Híbrido C
     F. Método MIDREM
     G. Voltar a tela-Chave de segundo nivel

B (Método de Newton)

          Método de Newton

     Mostrar os resultados parciais (S/N) ? S

     Entre x0 e ASC ? -0.5,5


<resultado>
```

I	X	F(X)	F'(X)	ASC
0	-0.5000000	-3.0625000	-11.5000000	0.0
1	-0.7663043	-0.6332331	-6.7820646	0.0
2	-0.8596731	-0.0738070	-5.2119997	1.5
3	-0.8738341	-0.0016451	-4.9799620	3.1
4	-0.8741645	-0.0000001	-4.9745705	7.3

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

     Mostrar os resultados parciais (S/N) ? N

     Entre x0, ASC ? 3.,5

<resultado>

aproximação inicial: 3

Número de iterações: 3
Raiz:  3.03524499
ASC:   6.743843

Fazer outro cálculo (S/N) ? N

<volta a tela-chave de terceiro nível>

G (voltar a tela-chave de segundo nível)

7 (voltar a tela-chave de primeiro nivel)

C fim
```

16. BIBLIOGRAFIA:

[STA79] [BAR83] [RAL65] [FOR77] [DOR79] [SHO84] [TRA64]
[MAR82] [CLA83] [KRO79] [VAN78] [DEW83] [JOH82] [CON72]

17. OBSERVAÇÕES:

APÊNDICE B

MANUAL INSTRUCIONAL DO USUARIO
S I N A I - 1 6
RESOLUÇÃO DE EQUAÇOES ALGÉBRICAS

1. NOME DO SOFTWARE - nome completo do software

2. OBJETIVOS DO SOFTWARE - problemas que o software pode resolver

3. DETALHES TÉCNICOS DO AMBIENTE
   - processador
   - requisitos de memória
   - equipamentos periféricos
   - equipamento adicional
   - sistema operacional
   - linguagem e interpretador

4. ORGANIZAÇÃO, INSTALAÇÃO E UTILIZAÇÃO

5. FICHAS INSTRUCIONAIS
   - descrição das fichas
   - itens das fichas
     . nome
     . classificação
     . objetivo
     . descrição
     . interpretação geométrica
     . limitações e restrições
     . erro no método
     . ordem de convergência
     . índices de eficiência
     . algoritmos
     . implementações
     . listagem do programa
     . mensagens de erro
     . dados de entrada e saída
     . exemplos de utilização do programa
     . bibliografia
     . observações

6. INFORMAÇÕES DE VALIDAÇÃO E EXATIDÃO

7. REFERÊNCIAS A OUTROS MANUAIS

8. BIBLIOGRAFIA

9. INDICE

10. INFORMAÇÕES ADMINISTRATIVAS

11. DATA DA PUBLICAÇÃO

APÊNDICE C

MANUAL DE MANUTENÇÃO DO PROGRAMA
S I N A I - 16
RESOLUÇÃO DE EQUAÇOES ALGÉBRIAS

1. NOME DO SOFTWARE

2. OBJETIVOS DO SOFTWARE

3. DETALHES TÉCNICOS DO AMBIENTE
   - processador
   - requisitos de memória
   - equipamentos periféricos
   - equipamento adicional
   - sistema operacional
   - linguagem e interpretador

4. CONSTANTES E VARIÁVEIS DEPENDENTES DE MÁQUINA

5. OUTROS SISTEMAS COMPATÍVEIS

6. ORGANIZAÇÃO DO SOFTWARE

7. LISTAGEM COMPLETA

8. DESCRIÇÃO DOS MÉTODOS E ALGORÍTMOS

9. INFORMAÇOES DE TEMPO, EXATIDÃO E VALIDAÇÃO

10. RELAÇÃO DAS MENSAGENS DE ERRO

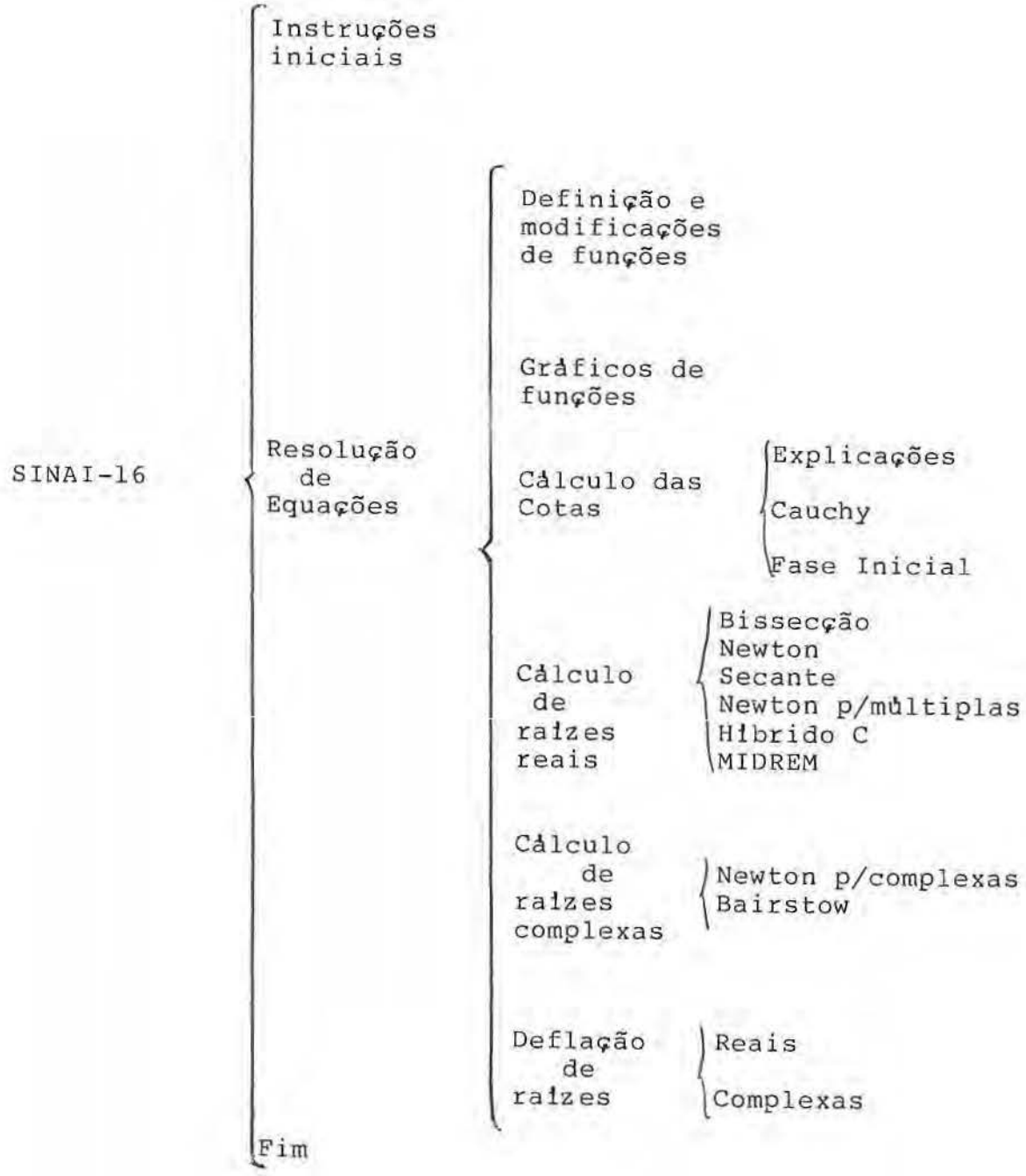
11. ÍNDICE

12. INFORMAÇÕES ADMINISTRATIVAS

13. DATA DA PUBLICAÇÃO

APENDICE D

Níveis da atividade do SINAI-16



BIBLIOGRAFIA

[ADA67] ADAMS, D.A A Stopping criterion for polynomial root finding Communication of ACM. New York, 10(10): 658-65, Oct.1967.

[ALB73] ALBRECHT, P. Análise numérica um curso moderno Rio de Janeiro, Livros Técnicos e Científicos, 1973.

[AND73] ANDERSON,N & BJÖRCK,A. A New high order method of regula-falsi type for computing a root of an equation. BIT, Copenhagen,13:253-64, 1973.

[BAR73] BARROS SANTOS,V.R Curso de cálculo numérico Rio de Janeiro, Livros Técnicos e Científicos, 1973.

[BAR83] BARROSO,L.C. et alli. Cálculo numérico São Paulo, Harper & Row, 1983.

[BER65] BEREZIN, I.S & ZHIDKOV,N.P Computing methods London, Pergamon Press, 1965.

[BRE71] BRENT,J.C.P. An Algorithms with guaranteed convergence for finding a zero functions. Computing Journal, 14:422-5, 1971.

[BUS70] BUSHNELL,R.C. User modificable software. In:RICE,J. MATHEMATICAL SOFTWARE SYMPOSIUM, Lafayete,Apr.1-3, 1970. Proceeding, New York, Academic Press,1971. p.59-66.

[CAL69] CALINGAERT,P. Princípios de computação Rio de Janeiro, Livros Técnicos e Científicos, 1969.

[CLA83] CLAUDIO,D.M & ROYO DOS SANTOS,J.A Microcomputadores e minicalculadoras progamáveis, seu uso em ciências e engenharia. São Paulo,Edgard Blücher,1983.

[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. Beriln, Springer-Verlag, 1982. p.1-23.

[CON72] CONTE,S.D & DE BOOR,C Elementary numerical analysis on algorithmic approach. New York, McGraw Hill, 1972.

[CON77] CONTE, S.D Elementos de análise numérica. Porto Alegre, Globo, 1977.

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

[DAH69] DAHLQUIST,G & BJÖRCK,A. Numerical methods. Englewood Cliffs, Prentice Hall, 1969.

[DEW83] DEW,P.M. & JAMES,K.R. Introduction to numerical computation in PASCAL. Hong Kong, Macmillan Press, 1983.

[DIA83] DIAS, S.M.V.F. Contribuições para resolução numérica de equações polinômiais. São Carlos, USP, 1983

[DIV85] DIVERIO, T.A. Dificuldades dos algoritmos iterativos na resolução de equações não lineares. Porto Alegre, CPGCC da UFRGS, 1985. (não publicado)

[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.

[DOR79] DORN,W.S & McCRACKEN,D.D. Cálculo numérico com estudos de casos em FORTRAN IV. Rio de Janeiro,Campus,1979.

[DOW71] DOWELL,M & JARRAT,P. A Modified regula-falsi method for computing the root of an equation BIT, Copenhagen, 11:168-174, 1971.

[DOW72] DOWELL,M & JARRAT,P. The Pegasus method for computing the root of an equation.,Copenhagen, BIT, Copenhagen, 12:503-8, 1972.

[EVA74] EVANS, D.J Software for numerical mathematics. London, Academic Press, 1974.

[FEL69] FELDSTEIN,A. & FIRESTONE, R.M. A Study of Ostrowski efficiency for composite iteration algorithms. Proceedings, 1969.

[FOL82] FOLEY, J.D. & VAN DAN,A . Fundamentals of interactive computer graphics. Massachusetts, Addilson Wesley, 1982.

[FOR74] FORD,B & HAGUE,S.J The Organisation of numerical algorithms libraries.In: EVANS,D.J Software for numerical mathematics, New York, Academic Press,1974.

[FOR77] FORSYTHE,G.E et alli. Computer methods for mathematical computations. Englewood Cliffs, Prentice Hall, 1977.

[FRA85] FRANCIOSI,B.R.T Introduction to numerical computation. Porto Alegre, PGCC da UFRGS, 1985. (Trab. Individual)

[FRO64] FRÖBERG,G.E. Introduction to numerical analysis. Massachusetts, Addilson Wesley, 1964.

[GEA78] GEAR,C.W Applications and algorithms in computer science. Chicago, Science Research Association, 1978.

[GEA80] GEAR,C.W Numerical Software: science or alchemy?. Advances in computer, New York, 19:229-48, 1980.

[GRO78] GROGONO, P. Programming in Pascal. London, Addilson Wesley, 1978.

[HOF85] HOFFMAN,P & NICOLOFF,T. MS-DOS: guia do usuário São Paulo, McGraw Hill, 1985.

[HOP74] HOPCROFT,J.E Complexity of computer computations.In: INFORMATION PROCESSING 74, Stockholm, Aug.5-10,1974 Proceedings Amsterdam, North-Holland, 1974. p.620-6

[HOR75] HORNBECK,R.W. Numerical Methodos. New York, Quantum Publishers, 1975.

[HOU53] HOUSEHOLDER, A.S Principles of numerical analysis. New York, McGraw Hill, 1953.

[IGA84] IGARASHI, Masao A termination criterion for iteractive methods used to find the zeros of polynomials. Mathematical of Computation. New York, 42(165): 165-71,Jan. 1984.

[JEN78] JENSEN, K & WIRTH, N. Pascal user manual an report. Berlim, Springe-Verlag, 1978.

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

[KAH63] KAHAN,W & FARKAS,I Algorithm 168 and algorithm 169. In: ACM 6 Apr.1963, p.165.

[KIN73] KING, Richard An Improved pegasus method for root finding,BIT, Copenhagen, 13:423-7,1973.

[KIN76] KING,R Methods without secant steps for findig a bracketed root. Computer Journal, New York, 17:49-57, 1976.

[KRO79] KRONSJO, Lydia Algorithms their complexity and efficiency. Chichester, John Wiley & Sons, 1979.

[KUN73] KUNG, H.T. & TRAUB, J.F Optimal order of one-point and multi-point iteration. Journal of ACM, New York, 21(4): 643-51 oct. 1974.

[LOI85] LOIOLA, C.R.A Rotinas matemáticas em BASIC para micros Rio de Janeiro, Campus, 1985.

[MAR82] MARINS, J.M. Métodos computacionais para o cálculo de raźes de equações polinomiais. Porto Alegre, PGCC da UFRGS, 1982. (dissertação).

[MOO75] MOORE,R.E Mathematical elements of scientific computing. New York, Holt Rinehart & Winston Inc, 1975.

[OST60] OSTROWSKI,A.M Solution of equations and systems of equations. New York, Academic Press, 1960.

[PAC73] PACITTI, J. & ATKINSON, C.P. Programação e métodos computacionais. Rio de Janeiro, Livros Técnicos e Científicos, 1977.

[PAT72] PATERSON, M.S. Efficient iterations for algebric numbers in complexity of computer computations London, Plenum Press, 1972.

[RAL65] RALSTON,A A First course in numerical analysis. New York, McGraw Hill, 1965.

[RAL67] RALSTON,A & WILF,H.S Mathematical methods for digital computer. New York, John Wiley & sons, 1967.

[RED61] REDISH, K.A An Introduction to computational methods. New York, John Wiley & Sons, 1961.

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

[SAC85] SACHS, Jonathan. IBM PC e compatíveis: guia do usuário São Paulo, McGraw Hill, 1985.

[SAD80] SADOSKY,M. Cálculo numérico e gráfico. Rio de Janeiro, Interciência, 1980.

[SCH86] SCHAFFER,M.I. Análise computacional de métodos iterativos pontuais e multipontuais na resolução de equações polinômiais. Porto Alegre, PGCC da UFRGS, 1986. (não publicado)

[SHO84] SHOUP, T.E Applied numerical methods for the microcomputer. Englewood Cliffs, Prentice Hall, 1984.

[STA79] STARK, Peter A. Introdução aos métodos numéricos, Rio de Janeiro, Interciência, 1979.

[TER82] TERADA,R Desenvolvimento de algoritmos e complexidade de computação. Rio de Janeiro, III Escola de Computação, 1982.

[TRA64] TRAUB,J.F. Iteractive methods for the soluction of equations. Englewood Cliffs, Prentice Hall, 1964.

[VAN78] VANDERGRAFF,J Introduction to numerical computation. New York, Academic Press, 1978.

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

[WIL63] WILKINSON,J.H. Rounding error in algebric process. New Jersey, Prentice Hall, 1963.

[WON84] WONG,P. Choices for mathematical words processing software. SIAM news, Philadelphia, 17 (6):8-9, Nov. 1984

[ZUR76] ZURMÜHL, R. Numerical analysisfor engineers and physicists. Berlin, Springer-Verlag, 1976.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

Pós-Graduação em Ciência da Computação

Software Numérico aplicativo

e instrucional

Dissertação apresentada aos Srs.

Dalcidio Pereira Claudio

Inês J. P. Claudio

Inês J. P. Claudio

J-IRG

Visto e permitida a impressão

Porto Alegre, 02/06/ 1986 .

Coordenador do Curso de Pós-Graduação
em Ciência da Computação