Marcos Paulo de Mesquita

Matemática Intervalar: Princípios e a Ferramenta C-XSC

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências da disciplina Projeto Orientado para obtenção do título de Bacharel em Ciência da Computação.

Orientador
Jones Oliveira de Albuquerque

Lavras
Minas Gerais - Brasil
2002

Marcos Paulo de Mesquita

Matemática Intervalar: Princípios e a Ferramenta C-XSC

Monografia de Graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências da disciplina Projeto Orientado para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em *19 de Julho de 2002*

Profa. Renata Couto Moreira

Prof. Ricardo Martins de Abreu Silva

Jones Oliveira de Albuquerque
(Orientador)

Lavras
Minas Gerais - Brasil

A minha irmã Madalena e ao meu cunhado Nogueira que hoje, de longe,
e tão perto, compartilham comigo este momento de felicidade.

Agradecimentos

Agradeço a Deus por me permitir estar aqui.

Agradeço a minha mãe por toda renúncia e doação.

Agradeço aos meus irmãos Alice, Beto, Melinha,
Milene, Marialva, Maria Helena e Madalena que, com muito
amor, sempre estiveram do meu lado.

Agradeço aos meus sobrinhos Rafaella, Tiago, Daiana,
Amanda, Leonardo, Vanessa, Aline, Angélica, Andrelise,
Dalise, Daniel e Danilo que são minha motivação maior.

Agradeço a Sonila por todo carinho e dedicação.

Agradeço aos amigos Danilo, Dalton e Ana Maria
por tudo que me ensinaram.

Agradeço aos amigos Vanessa, Alisson, Deive
Paulo Sérgio, Joseane, Alessandra, Jerusa, Giselle
e Gláucia por todas as palavras de motivação.

Agradeço aos companheiros, e acima de tudo, amigos
do 107, Fábio, Samuel, Bruno, Thiago, Pablo, Shaolin e Cristiano
por toda força e companheirismo.

Agradeço a minhas amigas e amigos da PROEX por
toda demonstração de respeito e amizade.

Agradeço ao Professor Jones por todo apoio e compreensão.

Resumo

A computação científica digital consiste numa seqüência finita de operações aritméticas. Por outro lado, a solução exata de um problema matemático, na maioria da vezes, requer uma seqüência infinita de operações aritméticas exatas. A Matemática Intervalar é uma teoria matemática que se apresenta como uma solução não somente para o controle rigoroso e automático de erros de resultados das computações numéricas, mas também para o tratamento e modelagem da incerteza em computação. Este trabalho descreve os principais conceitos desta teoria e apresenta uma ferramenta computacional, C-XSC, utilizada na implementação de algoritmos intervalares.

Sumário

Lista de Figuras

Capítulo 1

Introdução

Os problemas da Computação Científica se concentram fundamentalmente em três aspectos: na criação do modelo computacional que reflita de forma mais fiel possível a realidade em questão, no controle e análise dos erros que ocorrem no processo computacional e na escolha das técnicas de programação adequadas para desenvolvimento de software científico.

Salienta-se aqui que a qualidade de um resultado, em computação científica, depende do conhecimento e do controle que se possa ter sobre seu erro. Algoritmos convencionais, normalmente utilizados em Computação Científica, chamados de algoritmos pontuais, computam uma estimativa para uma resposta, e, talvez, um erro estimado. O usuário não pode afirmar a exatidão da resposta estimada sem o auxílio de uma análise de erro, que é extensa, dispendiosa e nem sempre viável.

Existem três fontes de erros em computação numérica: (i) a propagação de erro nos dados e parâmetros iniciais, que é a mais séria, porque não é possível torná-la arbitrariamente pequena via computação adicional, (ii) o erro de arredondamento e (iii) o erro de truncamento. As técnicas intervalares consistem em uma alternativa para alcançar limites garantidos para os resultados de computações científicas, através do controle rigoroso e automático do erro do resultado.

A Análise de Intervalos, uma teoria matemática com origem na década de 60 [MY59], tem por objetivo responder à questão da exatidão e da eficiência que aparece na prática da Computação Científica. Ela está interessada em técnicas que podem ser programadas por computador, contendo em sua computação uma análise rigorosa, completa e automática dos erros de resultado. Em 1974, Leslie Fox propõem uma análise combinando diferentes áreas como análise intervalar, topologia intervalar, álgebra intervalar e outras [RIB01].

Técnicas intervalares manipulam dados e parâmetros inicias como intervalos, com o indicativo do erro máximo presente nestes valores antes que os mesmos sejam introduzidos no computador. Apresentam-se esquemas computacionais que tratam tanto do problema da propagação do erro destes dados e parâmetros iniciais ao longo do processo computacional, assim como dos erros de arredondamento e truncamento. A propagação do erro nos dados iniciais e a acumulação do erro de arredondamento em qualquer seqüência finita de operações aritméticas podem ser ambas rigorosamente controladas simplesmente pela utilização de aritmética de máquina.

Desta forma algoritmos intervalares, em contraste com os algoritmos pontuais, computam um intervalo como solução, com a garantia de que a resposta pertence a este intervalo [HE01]. Portanto, resultados intervalares carregam sempre consigo a segurança de sua qualidade e o grau de sua incerteza, pois o diâmetro de um intervalo solução é um indicativo da influência do erro do dado de entrada no erro do resultado final obtido. Este é um tipo de análise de sensibilidade, que pode substituir execuções de simulação repetidas e dispendiosas.

Atingir uma solução intervalar significativa requer uma fundamentação matemática cuidadosa de todos os estágios do desenvolvimento do algoritmo e sua implementação. Os algoritmos a serem desenvolvidos devem ser algoritmos intervalares, e não versões intervalares de algoritmos pontuais..

O uso de ambientes de programação que suportem representação intervalar para as operações de cálculo científico favorece o controle automático de erros através de métodos auto-validáveis (métodos que se encarregam de verificar e garantir a exatidão dos cálculos efetuados). Foram utilizados neste trabalho a linguagem para a computação científica *C-XSC* e o conjunto de ferramentas para resolução de problemas numéricos com verificações dos resultados *CToolbox* para estabelecer alternativas de resolução de problemas de computação científica.

Capítulo 2

A Aritmética Intervalar

O presente capítulo visa a apresentação das principais definições que se fazem necessárias no estudo da Matemática Intervalar, tais como: a noção de intervalo, a definição do conjunto de intervalos de números reais IR e suas correspondentes operações aritméticas. Aqui também serão destacados alguns aspectos topológicos deste conjunto e uma breve descrição dos conceitos de funções intervalares. As definições aqui apresentadas estão mais detalhadas em [MOO62], [SUN58] e [CRU01].

2.1 Breve Histórico da Aritmética Intervalar

Um dos primeiros relatos da utilização de intervalos como estimadores formais de uma grandeza é o algoritmo de Arquimedes para a estimar o valor de π. Através de uma sucessão de polígonos inscritos e circunscritos com número de lados crescente, Arquimedes conseguiu gerar uma seqüência convergente de intervalos para estimar essa constante. Por outro lado, os primeiros estudos da aritmética intervalar como ramo da Computação Científica aparecem na década de 1950, através de alguns estudos isolados e que pouco a pouco passaram a requisitar a atenção de um número maior de pesquisadores. É nesse contexto que se apresenta o trabalho de Sunaga [SUN58], no qual são investigadas as regras que definem as operações aritméticas entre intervalos. Neste trabalho são definidos vetores e matrizes intervalares, juntamente com as operações correspondentes, e são esboçados exemplos de aplicações da aritmética intervalar para a determinação de soluções intervalares para raízes de funções e para integrais. Porém, somente com o primeiro livro sobre análise intervalar, publicado por Moore [MOO62], tais resultados passaram a receber mais atenção da comunidade científica.

Um dos fundamentos que motivaram o desenvolvimento da aritmética intervalar foi o desenvolvimento de algoritmos numéricos para Computação Científica. Neste contexto, a compreensão dos efeitos da existência de uma aritmética de ponto flutuante de precisão finita (associada à noção de erro de arredondamento) aliada à necessidade de truncamento de certos métodos iterativos impulsionou o desenvolvimento de algoritmos cuja a saída fosse capaz de garantir a proximidade entre a solução exata e as respostas produzidas. Nesse sentido Rump [RUM 88] (conforme [VAC01]) apresenta uma interessante análise comparativa do lugar da abordagem intervalar frente a outras abordagens de solução de problemas do ponto de vista computacional, tais como algoritmos algébricos e algoritmos numéricos.

Durante as últimas três décadas o lugar dos intervalos compactos como objetos independentes tem crescido continuamente na análise numérica, na verificação ou determinação de soluções de vários problemas matemáticos ou na prova de que tais problemas não possuem solução em um domínio particular. Diversas áreas de aplicação foram exploradas através da abordagem intervalar: problemas em engenharia (estrutural, química, mecânica, elétrica), robótica, controle, economia, etc [KEA97] e [DD01]. Do ponto de vista matemático pode-se citar problemas associados à solução de sistemas lineares ou não lineares, otimização (restrita ou global), determinação de valores e vetores próprios, solução de problemas de contorno e de equações diferenciais, entre outros. Isto foi possível através da compreensão de intervalos como extensões de números reais ou complexos, da introdução de funções intervalares e de aritméticas intervalares.

2.2 Intervalo de Números Reais

Um intervalo de reais, ou simplesmente intervalo, é denotado pelo par ordenado de números reais, $[x_1; x_2]$, com $x_1 \leq x_2$, e representa o conjunto de todos os números reais x tais que $x_1 \leq x \leq x_2$.

São exemplos de intervalos: [7;12], [-5;-2], [9;9]. Note que o intervalo [9;9] corresponde ao próprio número real 9, recebendo a denominação de *intervalo pontual*.

2.3 O Conjunto IR

Define-se IR como sendo o conjunto de todos os intervalos de reais, isto é, IR= $\{[x_1; x_2] | x_1; x_2 \in I\!R, \ x_1 \leq x_2\}$. Assim, vale a seguinte cadeia de inclusões

$I\!N \subseteq Z\!\!\!Z \subseteq Q\!\!\!\!I \subseteq I\!R \subseteq II\!R$.
Acompanhe na Figura 2.1 que associando-se a cada intervalo $[x_1; x_2] \in$ IR, um ponto $(x_1, x_2) \in I\!R^2$, obteremos uma representação geométrica para IR.

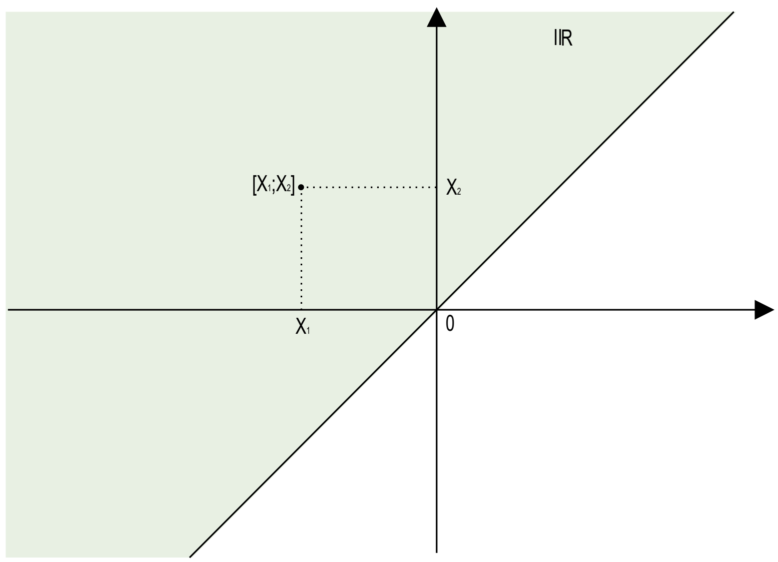

Figura 2.1: Representação geométrica de IR

2.4 Operações aritméticas em IR

Sejam $A, B \in$ IR dois intervalos de reais. As operações de soma, subtração, multiplicação e divisão em IR são definidas por $A * B = \{a * b | a \in A, b \in B\}$, onde $* \in \{+, -, ., /\}$ é quaisquer uma das quatro operações aritméticas. Se ω é uma operação unária, então ωX é definida por $\omega X = \omega(X) = \{\omega(x) | x \in X\} = [min\{\omega(X) | x \in X\}; max\{\omega(X) | x \in X\}]$
Para a operação de divisão, devemos assumir que $0 \notin B$ para que a operação seja bem definida.

2.4.1 Soma Intervalar

Sejam $A, B \in$ IR dois intervalos de reais, com $A = [a_1; a_2]$ e $B = [b_1; b_2]$. Define-se a soma de A com B como sendo:

$$A + B = [(a_1 + b_1); (a_2 + b_2)]$$

A Figura 2.2 ilustra um exemplo de soma de dois intervalos.

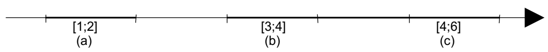

Figura 2.2: Intervalos na reta real R: (a) Intervalo $A = [1; 2]$; (b) intervalo $B = [3; 4]$ e (c) intervalo $A + B = [4; 6]$

Sejam $A, B\, e\, C \in$ IR. As seguintes propriedades algébricas se aplicam à soma de intervalos em IR:

Fechamento: Se $A \in$ IR e $B \in$ IR então $A + B \in$ IR;

Associatividade: $A + (B + C) = (A + B) + C$;

Comutatividade: $A + B = B + A$;

Elemento Neutro: $\exists, 0 = [0; 0] \in$ IR tal que $A + 0 = 0 + A = A$.

Observações:

- O conjunto IR não possui inverso aditivo, ou seja, nem sempre pode-se achar um intervalo $-A$ tal que $A + (-A) = 0$.
- Seja A um intervalo. Então $0 \in A - A$.

2.4.2 Pseudo Inverso Aditivo Intervalar

Seja $A \in$ IR um intervalo de reais, com $A = [a_1; a_2]$. Então:

$$-A = [-a_2; -a_1].$$

Exemplo: Seja $A = [-1; 2]$. Temos $-A = [-2; 1]$.

2.4.3 Subtração Intervalar

Sejam dois intervalos de reais $A, B \in$IR, com $A = [a_1; a_2]$ e $B = [b_1; b_2]$. Define-se a subtração de A com B como sendo

$$A - B = A + (-B) = [(a_1 - b_2); (a_2 - b_1)]$$

Exemplo: Sejam $A = [-1; 5]$ e $B = [2; 7]$. Temos $A - B = [(-1) - 7;\ 5 - 2] = [-8; 3]$.

2.4.4 Multiplicação Intervalar

Sejam dois intervalos de reais $A, B \in$IR, com $A = [a_1; a_2]$ e $B = [b_1; b_2]$. Define-se a multiplicação de A com B como sendo:

$$A.B = [min\{a_1.b_1, a_1.b_2, a_2.b_1, a_2.b_2\}; max\{a_1.b_1, a_1.b_2, a_2.b_1, a_2.b_2\}]$$

Exemplo: Sejam $A = [-2; 3]$ e $B = [4; 5]$. Temos $A.B = [min\{(-2).4, (-2).5, 3.4, 3.5\}; max\{(-2).4, (-2).5, 3.4, 3.5\}] = [-10; 15]$.

Sejam A, B *e* $C \in$ IR. As seguintes propriedades algébricas se aplicam à multiplicação de intervalos em IR:

Fechamento: Se $A \in$ IR e $B \in$ IR então $A.B \in$ IR;

Associatividade: $A.(B.C) = (A.B).C$;

Comutatividade: $A.B = B.A$;

Elemento Neutro: $\exists$, $1 = [1; 1] \in$ IR tal que $A.1 = 1.A = A$.

Subdistributividade: $A.(B + C) \subseteq (A.B) + (A.C)$.

Observações:

- O conjunto IR não possui inverso multiplicativo, ou seja, nem sempre pode-se achar um intervalo A^{-1} tal que $A.A^{-1} = 1$;
- Seja A um intervalo tal que $0 \notin A$. Então $1 \in \frac{A}{A}$.

Para fins de implementação em computadores, pode-se otimizar os cálculos feitos nos caso da multiplicação e da divisão, considerando-se os sinais dos extremos dos intervalos, que nos levam a analisar os seguintes nove casos:

1. $a_1 \geq 0$ *e* $b_1 \geq 0 \rightarrow A.B = [a_1.b_1; a_2.b_2]$;
2. $a_1 \geq 0$ *e* $b_1 < 0 \leq b_2 \rightarrow A.B = [a_2.b_1; a_2.b_2]$;
3. $a_1 \geq 0$ *e* $b_2 < 0 \rightarrow A.B = [a_2.b_1; a_1.b_2]$;
4. $a_1 < 0 \leq a_2$ *e* $b_1 \geq 0 \rightarrow A.B = [a_1.b_2; a_2.b_2]$;
5. $a_1 < 0 \leq a_2$ *e* $b_1 \leq 0 \leq b_2 \rightarrow A.B = [min\{a_1.b_2, a_2.b_1\}; max\{a_1.b_1, a_2.b_2\}]$;

6. $a_1 < 0 \leq a_2 \; e \; b_2 < 0 \rightarrow A.B = [a_2.b_1; a_1.b_1]$;

7. $a_2 < 0 \; e \; b_1 \geq 0 \rightarrow A.B = [a_1.b_2; a_2.b_1]$;

8. $a_2 < 0 \; e \; b_1 < 0 \leq b_2 \rightarrow A.B = [a_1.b_2; a_1.b_1]$;

9. $a_2 < 0 \; e \; b_2 < 0 \rightarrow A.B = [a_2.b_2; a_1.b_1]$;

2.4.5 Pseudo Inverso Multiplicativo Intervalar

Seja $A \in$ IR um intervalo de reais, com $A = [a_1; a_2]$ e $0 \notin A$. Então:

$$A^{-1} = 1/A = [\tfrac{1}{a_2}; \tfrac{1}{a_1}].$$

Exemplo 1: Seja $A = [3; 4]$. Temos $A^{-1} = [\frac{1}{4}; \frac{1}{3}]$.
Exemplo 2: Seja $A = [-4; -2]$. Temos $A^{-1} = [\frac{1}{-2}; \frac{1}{-4}] = [-\frac{1}{2}; -\frac{1}{4}]$.

2.4.6 Divisão Intervalar

Sejam dois intervalos de reais $A, B \in$IR, com $A = [a_1; a_2]$ e $B = [b_1; b_2]$ e $0 \notin B$. Define-se a divisão de A com B como sendo:

$$\frac{A}{B} = A.B^{-1} = [min\{\tfrac{a_1}{b_2}, \tfrac{a_1}{b_1}, \tfrac{a_2}{b_2}, \tfrac{a_2}{b_1}\}; max\{\tfrac{a_1}{b_2}, \tfrac{a_1}{b_1}, \tfrac{a_2}{b_2}, \tfrac{a_2}{b_1}\}], \text{ com } 0 \notin [b_1; b_2].$$

Exemplo: Sejam $A = [3; 4]$ e $B = [4; 6]$. Tem-se $A/B = [min\{3/6, 3/4, 4/6, 4/4\}; max\{3/6, 3/4, 4/6, 4/4\}] = [\frac{1}{2}; 1]$.

2.4.7 Inclusão Monotônica

Sejam $A, B, C, D \in$ IR intervalos de reais, tais que $A \subseteq C \; e \; B \subseteq D$. Então, valem:

1. $A + B \subseteq C + D$;

2. $-A \subseteq -C$;

3. $A - B \subseteq C - D$;

4. $A.B \subseteq C.D$;

5. $1/A \subseteq 1/C, sempre \; que \; 0 \notin C$;

6. $A/B \subseteq C/D, sempre \; que \; 0 \notin D$.

2.5 Definições Topológicas em IR

2.5.1 Intervalo Simétrico

Seja $A \in$ IR um intervalo. A é um *intervalo simétrico* se $-A = A$.
Exemplos: $[-1;1], [-\pi;\pi], [0;0]$.

2.5.2 Intersecção de dois Intervalos

Sejam $A = [a_1;a_2]$ e $B = [b_1;b_2]$ dois intervalos conforme Figura 2.3. Define-se a *intersecção* dos intervalos A e B como sendo o intervalo $A \cap B = [max\{a_1, b_1\};$ $min\{a_2, b_2\}]$, se $max\{a_1, b_1\} \leq min\{a_2, b_2\}$. Se $min\{a_2, b_2\} < max\{a_1, b_1\}$ então $A \cap B = \emptyset$.

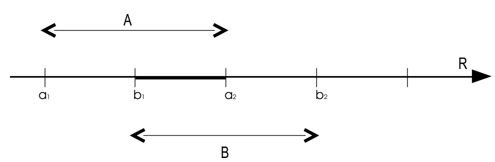

Figura 2.3: Representação geométrica da intersecção em R

2.5.3 União de dois Intervalos

Sejam $A = [a_1;a_2]$ e $B = [b_1;b_2]$ dois intervalos conforme Figura 2.4tais que $A \cap B \neq O$. Define-se a *união* dos intervalos A e B como sendo o intervalo $A \cup B = [min\{a_1, b_1\};$
$max\{a_2, b_2\}]$.

2.5.4 União Convexa de dois Intervalos

Sejam $A = [a_1;a_2]$ e $B = [b_1;b_2]$ dois intervalos quaisquer. Define-se a *união convexa* dos intervalos A e B como sendo o intervalo $\overline{A \cup B} = [min\{a_1, b_1\};$
$max\{a_2, b_2\}]$.
Observação: No caso da união convexa, a intersecção dos dois intervalos pode ser vazia. Neste caso, o intervalo resultante será o intervalo de menor diâmetro que contém, simultaneamente, ambos os intervalos operados.

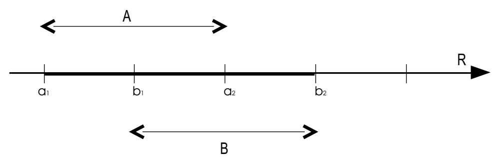

Figura 2.4: Representação geométrica do ponto médio de um intervalo em R

2.5.5 Distância entre dois Intervalos

Sejam $A = [a_1; a_2]$ e $B = [b_1; b_2]$ dois intervalos confome Figura 2.5. Define-se a *distância* de A e B como sendo o número real não-negativo $\delta = max\{|a_1 - b_1|, |a_2 - b_2|\}$.
Notação: $dist(A, B) = dist([a_1; a_2], [b_1; b_2]) = max\{|a_1 - b_1|, |a_2 - b_2|\} \geq 0$.

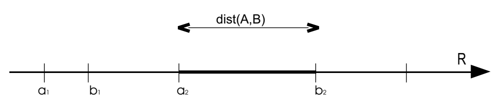

Figura 2.5: Representação geométrica da distância em R

2.5.6 Módulo de um Intervalo

Seja $A = [a_1; a_2] \in$ IR um intervalo conforme Figura 2.6. Define-se o *módulo* do intervalo A como sendo o número real não-negativo $\mu = dist(A, 0)$, que corresponde à distância de A ao zero.
Notação: $|A| = |[a_1; a_2]| = dist(A, 0) = max\{|a_1|, |a_2|\} \geq 0$.

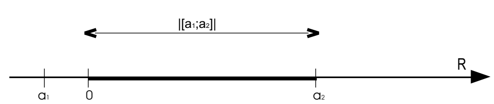

Figura 2.6: Representação geométrica do módulo de um intervalo em IR

2.5.7 Diâmetro de um Intervalo

Seja $A = [a_1; a_2] \in$ IR um intervalo conforme Figura 2.7. Define-se o *diâmetro* do intervalo A como sendo o número real não-negativo $d = a_2 - a_1$.

Notação: $diam(A) = diam([a_1; a_2]) = a_2 - a_1 \geq 0$. A Figura 2.7 exibe o diâmetro do intervalo $[-3; 4]$ que corresponde a 7 unidades.

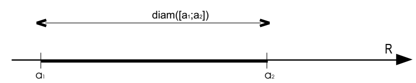

Figura 2.7: Representação geométrica do diâmetro de um intervalo em R

2.5.8 Ponto Médio de um Intervalo

Seja $A = [a_1; a_2] \in$ IR um intervalo conforme Figura **??**. Define-se o *ponto médio* do intervalo A como sendo o número real $m = \frac{a_1+a_2}{2}$.
Notação: $med(A) = med([a_1; a_2]) = \frac{a_1+a_2}{2}$.

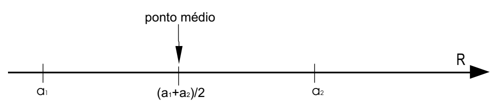

Figura 2.8: Representação geométrica do ponto médio de um intervalo em R

2.5.9 Inclusão Intervalar

Dado $x \in \mathbb{R}$, diz-se que $X \in$ IR é uma *inclusão intervalar* de x se $x \in X$.
Exemplo: O intervalo $[3; 4]$ é uma inclusão intervalar para o número π.

2.6 Funções Intervalares

Uma das mais importantes ferramentas fornecidas pela aritmética intervalar é a limitação da faixa de valores de uma função [KEA97]. Algumas definições que virão a seguir dá uma idéia de como uma função real pode ser transformada em uma função intervalar e as propriedades básicas da imagem intervalar e da avaliação intervalar de funções reais.

2.6.1 Imagem Intervalar de uma Função Real

Sejam f uma função real de variável real e X um intervalo tal que $X \subseteq Dom(f)$ e f é contínua em X. Define-se como imagem intervalar da função f em X, ou simplesmente imagem de f em X, o intervalo definido por:

$$I = \mathcal{I}(f, X) = [min\ f(x)|x \in X, max\ f(x)|x \in X].$$

Na figura 2.9 é ilustrada a imagem intervalar da função $f(x) = x^2 - x\ em\ [0; 2]$

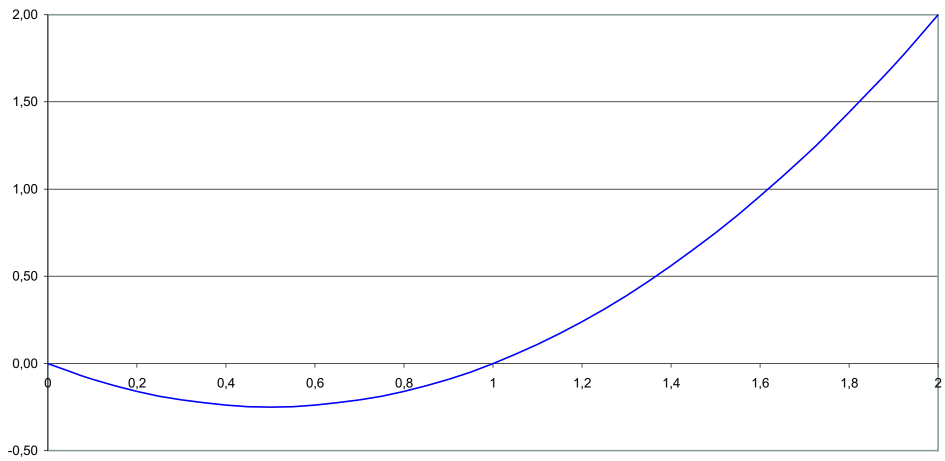

Figura 2.9: Imagem intervalar de $f(x) = x^2 - x\ em\ [0; 2]$

Nota-se que esta é uma maneira natural de se definir funções intervalares a partir de funções reais, ou seja, $Y = f(X) = \mathcal{I}(f, X)$, onde f é uma função real e X é um intervalo contido no domínio da função f.

Nota-se, também, que se $X = [x, x]$ é um intervalo pontual, então $Y = f(X)$ também é um intervalo pontual, dado por $Y = [f(x), f(x)]$. Assim, a função real está contida nesta extensão intervalar. Se $X = [a, b]$ é um intervalo com $diam(X) > 0$, então $\mathcal{I}(f, X)$ é o intervalo de menor diâmetro que contém todos os valores reais de $f(X)$, quando $x \in X$.

Exemplos:

(i) Seja $f(x) = x^2 - x\ e\ X = [0; 2] \subseteq I\!R = Dom(f)$. Assim, $\mathcal{I}(f, X) = (x^2 - x, [0; 2]) = [min\ x^2 - x|x \in [0; 2], max\ x^2 - x|x \in [0; 2]] = [\frac{-1}{4}; 2]$.

(ii) Seja $g(x) = x(x - 1)\ e\ X = [0; 2] \subseteq I\!R = Dom(g)$. Assim,$\mathcal{I}(g, X) = (x(x-1), [0; 2]) = [min\ x(x - 1)|x \in [0; 2], max\ x(x - 1)|x \in [0; 2]] = [\frac{-1}{4}; 2]$.

Observe que $\mathcal{I}(f, X) = \mathcal{I}(g, x)$, pois elas representam a mesma função real, porém escritas com expressões diferentes.

2.6.2 Avaliação Intervalar de uma Função Real

Sejam f uma função real de variável real x e X um intervalo. Define-se avaliação intervalar de f em X (ou extensão intervalar de f) como sendo a função intervalar $F(X)$, definida de maneira que cada ocorrência da variável real x é substituída pela variável intervalar X e cada operação $(+, -, ., /)$ é substitída pela respectiva operação intervalar de tal modo que, sendo $X = [x, x]$ for um intervalo pontual, então $F(X) = f(x)$.
Exemplo: Seja $f(x) = x^2 - x = x.x - x$. Então $F(X) = X.X - X$. Para $X = [0; 2]$, tem-se $F([0; 2])$=[0;2].[0;2]-[0;2]=[0;4]-[0;2]=[0;4]+[-2;0]=[-2;4].

Observação: Ao contrário da *imagem intervalar*, a *avaliação intervalar* depende da forma com que a função está expressa. Por exemplo, se tomarmos uma função *g*, tal que $g(x) = x(x - 1)$, tem-se $G(X) = X.(X - [1; 1]) = G([0; 2])$ =[0;2].([0;2]-[1;1])=[0;2].[-1;1]=[-2;2]. Assim, a avaliação da da função *f* descrita no exemplo acima é diferente da avaliação de *g*. Assim, $F(X) \neq G(X)$

Capítulo 3

A Biblioteca C-XSC

Os computadores eletrônicos possuem uma aritmética de ponto-flutuante que serve para aproximar os números reais e suas operações. A maioria das linguagens modernas de programação de alto nível realizam essas operações aritméticas através de símbolos, fazendo com que o programador seja capaz de escrever expressões simples, fórmulas ou funções da mesma forma que na notação usual científica.

Pesquisas no campo da aritmética computacional estão sendo desenvolvidas pelo Grupo de Matemática do IAM-UKA (Instituto de Matemática Aplicada da Universidade de Karlsreuhe) [HÖH97] desde os anos sessenta com o objetivo de possibilitar que computadores suportem uma aritmética de ponto-flutuante ordinária. Para tornar possível esses objetivos e algoritmos, circuitos de *hardware* mais rápidos têm sido projetados e implementados.

Algumas deficiências na linguagem de programação C fazem-na parecer imprópria para a programação de algoritmos numéricos, ela nem mesmo fornece as estruturas de dados numéricas básicas como vetores e matrizes [Ins96]. A linguagem de programação C++, uma extensão orientada a objetos de C, tornou-se mais popular devido a seus novos conceitos de estrutura de dados abstratas (classes), sobrecarga de operadores e funções, mas não se mostra melhor na programação numérica científica.

C-XSC (*C for extended for Scientific Computing*) é uma extensão da linguagem C++ para a computação científica. C-XSC provê aos programadores de C e C++ uma ferramenta para escrever algoritmos numéricos produzindo resultados confiáveis num ambiente de programação confortável. Em especial, as facilidades na manipulação do tipo de dado intervalo faz de C-XSC uma ferramenta quase que indispensável para projetos de implementações de algoritmos intervalares.

Com suas estruturas de dados abstratas, operadores e funções pré-definidas, C-XSC provê uma interface entre as linguagens C e C++ e a computação científica. Além disso, C-XSC suporta a programação de algoritmos que automaticamente incluem à solução de um dado problema matemático os limites verificados. Tais algoritmos entregam uma indicação matemática precisa a cerca da solução verdadeira.

As características mais importantes da C-XSC são:

- Os tipos real, complexos, intervalos, e aritmética intervalar complexa com propriedades matematicamente definidas;
- Matrizes e vetores dinâmicos;
- Subvetores de vetores e matrizes;
- Tipos de dados de Dotprecison;
- Operadores aritméticos pré-definidos com alta exatidão;
- Funções padrão de exatidão elevada;
- Aritmética de múltipla precisão dinâmica;
- Controle de arredondamento para dados de entrada/saída;
- Manipulação de erros;
- Biblioteca de rotinas para a solução de problemas.

3.1 Tipos de Dado Padrão, Funções e Operadores Pré-Definidos

A biblioteca C-XSC fornece os tipos de dados primitivos *real, interval, complex, e cinterval (intervalo complexo)* com suas apropridadas aritméticas, operadores relacionais e funções matemáticas. Todos os operadores são de máxima exatatidão. O arredondamentos dos operadores podem ser controlados usando os tipos *interval* e *cinterval*. Constante literais podem ser convertidas com máxima precisão. Todas as funções matemáticas são tipos de dados numéricos primitivos que podem ser chamados pelos seus nomes genéricos e, para seus argumentos possíveis, devolvem resultados com garantia de alta exatidão. Os tipos de dados escalar disponíveis são: *rvector, ivector, cvector, civector, rmatrix, imatrix, cmatrix, cimatrix*. O usuário pode alocar ou desalocar espaço dinamicamente para vetores e matrizes. Assim, sem recompilação, o mesmo programa pode usar estruturas de qualquer tamanho. A memória é usada de modo muito eficiente. Quando é acessado um componente de um vetor ou de uma matriz, o índice é verificado para aumentar a segurança, evitando acessos a endereços inválidos de memória.

3.2 Avaliação de Expressões com Alta Exatidão

Quando se avalia expressões aritméticas, a exatidão é um fator decisivo em muitos algoritmos numéricos. Mesmo se todos os operadores aritméticos e funções sejam de máxima exatidão, expressões compostas de uma série de operadores e funções não devolverão, necessariamente, resultados com máxima exatidão. Assim, métodos têm sido desenvolvidos para garantir tais resultados.

Um tipo especial dessas expressões são chamadas de *dot product expressions*, as quais são definidas como soma de expressões simples. Uma expressão simples é uma variável, uma constante, ou um produto destas duas. As variáveis podem ser escalar, vetorial, ou matriz. O resultado de tal expressão é um escalar, um vetor, ou uma matriz. Na análise numérica, *dot product expressions* são de grande importância. Por exemplo, métodos para correção de defeitos ou refinamento iterativo de problemas lineares e não-lineares são baseados em tais expressões. Uma avaliação destas expressões com máxima exatidão evitam cancelamentos.

Para avaliar com tal exatidão, C-XSC provê os tipos *dotprecision, cdotprecision, idotprecision, cidotprecision.*

3.3 Aritmética de Múltipla-Precisão Dinâmica

Além das classes *real* e *interval*, as classes dinâmicas *long real (l_real) e long interval (l_interval)* bem como as suas classes correspondentes de vetores e matrizes são implementadas incluindo toda aritmética, operadores relacionais e funções padrão de múltipla precisão. A precisão da computação pode ser controlada pelo usuário em tempo de execução. Pela substituição das declarações de *real e interval* por *l_real e l_interval* respectivamente, o usuário transforma sua aplicação em um programa de múltipla-precisão. Este conceito dá ao usuário uma ferramenta fácil e poderosa para análise de erros. Além do mais, em tempo de execução, é possível escrever programas que devolvam resultados numéricos com uma exatidão definida pelo usuário, bastanto apenas que ele indique tal precisão.

Todos os operadores para *real* e *interval* estão também disponíveis para *l_real e l_interval*. Adicionalmente, todas as possíveis combinações entre os tipos de precisão única ou múltipla estão disponíveis. Através de uma variável global pré-definida chamada *stagprec (staggred precision)* é feito o controle da precisão. Componentes de vetores e matrizes podem ter difentes níveis de precisão. Todas as operações em vetores e matrizes de múltipla-precisão são similares às de precisão simples.

3.4 Entrada e Saída em C-XSC

Usando o conceito de *stream* e sobrecarga de operadores $<<$ e $>>$ do C++, C-XSC permite arredondamentos e controle do formato durante E/S (Entrada e Saída) de dados. Parâmetros de E/S, como direção de arredondamento, tamanho dos campos, entre outros, também usam sobrecarga de operadores para manipular dados de E/S. Se um novo conjunto de parâmetros E/S está para ser usado, o conjunto de parâmetros antigos podem ser guardados numa pilha interna. Novos valores para os parâmetros podem ser então definidos. Depois de usados os novos conjuntos, os velhos poderam ser restaurados.

3.5 CToolbox

O *C++ Toolbox for Verified Computing* é um conjunto de ferramentas sofisticadas para resolução de problemas numéricos com verificação dos resultados usando características da C-XSC. Ele dispõe de funções para computação de raízes de equações, resolução de sistemas lineares, otimização de sistemas entre outras aplicações da área científica.

3.6 Como obter a C-XSC

O download do C-XSC 2.0 pode ser feito no seguinte endereço: `http://www.math.uni-wuppertal.de/wrswt/xsc/download.html`. As plataformas de desenvolvimento devem ser: PC com Linux ou Sun Solaris Workstation, com o *GNU C++ compiler gcc 2.95.2* instalado (presente na maioria das distribuições do Linux).

Capítulo 4

Estudo de Casos

Este capítulo descreve algumas aplicações da aritmética intervalar na computação numérica. Alguns casos foram estudados para ilustrar as principais funcionalidades do C-XSC e do CToolbox, bem como sua acurácia de seus resultados.

4.1 O Método de Newton Real

O Método de Newton real é um algoritmo que serve para calcular a raiz de uma dada equação, através da construção de uma sequência convergente de pontos da reta real.

Seja $f(x)$ uma função real contínua com derivada $f'(x)$ contínua num intervalo $[a;b]$ que contém a raiz real x_* de $f(x)$, de modo que $f'(x) \neq 0$. Se consideramos um ponto x_0 em $[a;b]$ e calculemos um novo ponto x_1 a partir de x_0.

Da geometria analítica, sabe-se que $\frac{y-y_p}{x-x_p} = a$ é a equação da reta que passa pelo ponto $P = (x_p, y_p)$ e que tem a inclinação a.

Da mesma forma, $\frac{y-f(X_0)}{x-x_0} = f'(x_0)$ é a equação da reta tangente ao gráfico da função f no ponto $(x_0, f(x_0))$. (Veja na figura 4.1)

Assim, $y = f'(x_0).(x - x_0) + f(x_0) = f'(x_0).x - f'(x_0).x_0 + f(x_0)$

Mas, x_1 é ponto de intersecção da reta tangente ao gráfico com o eixo das ordenadas, ou seja $y(x_1) = 0$, portanto $0 = f'(x_0).x_1 - f'(x_0).x_0 + f(x_0)$. Logo, $f'(x_0).x_1 = f'(x_0).x_0 - f(x_0)$. Assim, $x_1 = \frac{f'(x_0).x_0 - f(x_0)}{f'(x_0)} \therefore x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$ é o novo ponto da seqüência.

Chamamos de operador Newtoniano a expressão $N(x) = x - \frac{f(x)}{f'(x)}$

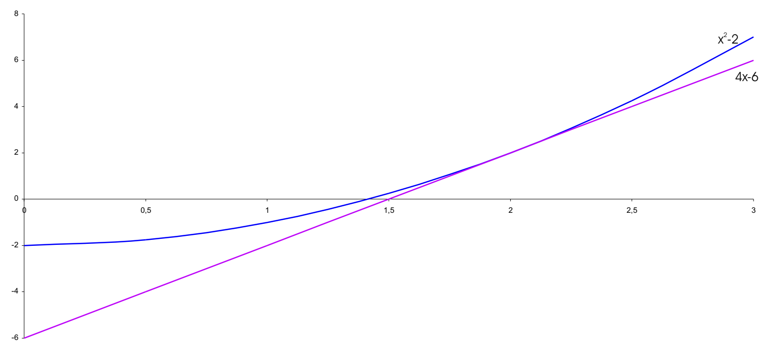

Figura 4.1: Método de Newton Real

4.2 O Método de Newton Intervalar

Uma atenção especial é dada nesta seção a fim de ressaltar o cuidado que se deve ter ao se transcrever versões intervalares de algoritmos reais. De maneira análoga ao método real, a versão intervalar do Método de Newton permite construir uma sequência convergente de intervalos, cujo limite será um intervalo que contém a raiz real da função dada.

Por ser um método autovalidável, se considerarmos um intervalo inicial que não contenha a raiz real, então, numa dada iteração, obtém-se um intervalo vazio como resultado. Caso contrário, se o intervalo inicial contém a raiz real da equação $f(x) = 0$ e, considerando que a seqüência intervalar que se obtém é de intervalos encaixados, então obtem-se como limite o intervalo de menor diâmetro possível, que ainda contém a raiz real desejada. Na prática, esta é a vantagem do uso do Método de Newton Intervalar.

A primeira idéia que se tem para definir o Método de Newton Intervalar é tomar uma extensão intervalar para o operador Newtoniano real, ou seja, definir $N(X) = X - \frac{F(X)}{F'(X)}$, onde $F(X)\ e\ F'(X)$ são extensões intervalares para as funções reais $f(x)\ e\ f'(x)$. Em [OLI97] é apresentado um contra-exemplo para esta abordagem, provando que desta forma o método sempre será divergente. Além do mais nenhum tratamento é feito a fim de se evitar divisões por zero.

Para que a seqüência de intervalos convirja para um intervalo pontual, intervalo este garantido ser a solução, é necessário que o diâmetro tenda para a largura zero. A idéia é construir uma uma seqüência de intervalos encaixados

$X_0 \supseteq X_1 \supseteq X_2 \supseteq X_3 \supseteq X_4 \supseteq ... \supseteq X_k...[x_*; x_*]$, cujos diâmetros vão diminuindo à medida que o valor de k aumenta. Para tal, avalia-se com intervalos somente a derivada, cuidando para que o intervalo resultante da avaliação intervalar da derivada não contenha o zero. Desta forma, é condição fundamental que o intervalo resultante da avaliação intervalar da derivada não contenha o zero, ou seja, é necessário que a função f não tenha pontos críticos (máximos ou mínimos locais) em X_0. Em [OLI97] é descrito o Método de Newton Intervalar convergente e o Operador Intervalar Newtoniano é apresentado da seguinte forma:

$$N(X) = med(X) - \frac{f(med(X))}{F'(X)},$$

construindo assim a sequência intervalar recursiva $X_{k+1} = X_k \cap N(X_k)$

4.2.1 Implementação do Método de Newton Intervalar

Implementação do Método de Newton para o cálculo da raiz da função $f(x) = \sqrt{x} + (x+1) * \cos(x)$ no intervalo $[2, 2.5]$
Foram utilizadas as classes *l_interval* e *l_imath* para que fosse permitido o uso de precisão-múltipla dinâmica da linguagem C-XSC.

```
#include <iostream>
#include "l_interval.hpp"         // Pacote da aritmética intervalar
#include "l_imath.hpp"            // Funções intervalares básicas
using namespace std;
using namespace cxsc;

l_interval f(const l_real& x)   // funcão f
{
  l_interval y(x);
  y = x;
  return sqrt(y) + (y+1)*cos(y);
}

l_interval deriv(const l_interval& x)  // derivada de f
{
  return (1/(2*sqrt(x))+cos(x)-(x+1)*sin(x));
}


bool criter(const l_interval& x)   // Verificando se existe a raiz no intervalo
{
  return Sup(f(Inf(x))*f(Sup(x)) ) < 0 &&  !(0 <= deriv(x));
}


int main(void)
```

```
{
  l_interval x, xold;
  stagprec=3;         // variável pré-definida que controla a precisão da computação
  x = l_interval(2,2.5);
  cout << "Intervalo inicial e [2,2.5]" << endl;
  cout << SetDotPrecision(16*stagprec, 16*stagprec-3) << RndNext;

if (criter(x))
{
  do {
      xold = x;
      cout << "Diametro do balanceamento atual: = " << real(diam(x)) << endl;
      x = (mid(x)-f(mid(x))/deriv(x)) & x;
  } while (x != xold);
      cout << "Balanceamento final da raiz: " << x << endl;
   }
    else
cout << "!Critérios não satisfeitos!" << endl;
    return 0;
}

// Saída do programa:
Intervalo inicial e [2,2.5]
Diametro do balanceamento atual: =   0.500000
Diametro do balanceamento atual: =   0.102332
Diametro do balanceamento atual: =   0.000940
Diametro do balanceamento atual: = 4.507683E-008
Diametro do balanceamento atual: = 5.714546E-018
Diametro do balanceamento atual: = 5.813014E-039
Diametro do balanceamento atual: = 5.473822E-048
Balanceamento final da raiz: [ 2.059045253415143788680636155343254522623083897,
                               2.059045253415143788680636155343254522623083898]
```

4.3 Avaliação de Expressões Aritméticas

4.3.1 Funções de Duas Variáveis

Avaliando a função abaixo para os parâmetros $x = 192119201$ e $y = 35675640$,

$$f(x, y) = \frac{1682xy^4 + 3x^3 + 29xy^2 - 2x^5 + 832}{107751}$$

o programa gera os seguintes resultados:

```
Avaliação em ponto-flutuante :    7.180560037061026E+020
Inclusão Intervalar          :   [ 1.783000000000000E+003,
                                   1.783000000000000E+003]
Correções de defeito necessárias:1
```

Observação: o resultado correto para f com tais argumentos é 1783

4.3.2 Diferencial de Segunda Ordem

Utilizando a função *Eval*, o resultado do quociente diferencial de segunda ordem

$$Df(x,h) = \frac{f(x-h)-2f(x)+f(x+h)}{h^2}$$

para a função

$$f(x) = 540\frac{x^4-23x^3+159x^2-2x+45}{x^3+18x^2+501x+20}$$

Para $h \rightarrow 0$ $Df(1,h) = 36$. Passando pra esta função um valor $h = (1.10^{-10})$, têm-se os seguintes resultados:

```
Avaliação em ponto-flutuante :    2.842170943040400E+002
Inclusão Intervalar          :  [ 3.600000000000000E+001,
                                  3.600000000000002E+001 ]
```

4.4 Zeros de Funções

Utilizando a função *AllZeros* do CToolbox, a avaliação da função:

$$e^{-3x} - sin(x)^3$$

no intervalo de entrada [0,20] com tolerância (1^{-3}), gerou os seguintes resultados:

```
Calculando todos os zeros da função  EXP(-3x)-POWER(SIN(x),3)
Intervalo de busca : [0,20]
Tolerância         : 0,001
[ 5.885327439818601E-001, 5.885327439818619E-001]
inclui um único zero local!
[ 5.885327439818601E-001, 5.885327439818619E-001]
inclui um único zero local!
[ 5.885327439818602E-001, 5.885327439818620E-001]
inclui um único zero local!
[ 5.885327439818602E-001, 5.885327439818620E-001]
inclui um único zero local!
[ 5.885327439818601E-001, 5.885327439818619E-001]
inclui um único zero local!
[ 5.885327439818601E-001, 5.885327439818619E-001]
inclui um único zero local!
[ 5.885327439818601E-001, 5.885327439818619E-001]
inclui um único zero local!
[ 5.885327439818601E-001, 5.885327439818619E-001]
inclui um único zero local!
[ 5.885327439818601E-001, 5.885327439818620E-001]
inclui um único zero local!
[ 5.885327439818601E-001, 5.885327439818620E-001]
inclui um único zero local!
[ 5.885327439818601E-001, 5.885327439818620E-001]
```

```
inclui um único zero local!
[ 5.885327439818601E-001, 5.885327439818620E-001]
inclui um único zero local!
[ 5.885327439818601E-001, 5.885327439818619E-001]
inclui um único zero local!
[ 5.885327439818601E-001, 5.885327439818620E-001]
inclui um único zero local!
[ 5.885327439818601E-001, 5.885327439818619E-001]
inclui um único zero local!
[ 3.096363932410645E+000, 3.096363932410647E+000]
inclui um único zero local!
[ 6.285049273382585E+000, 6.285049273382587E+000]
inclui um único zero local!
[ 9.424697254738520E+000, 9.424697254738522E+000]
inclui um único zero local!
[ 1.256637410168936E+001, 1.256637410168937E+001]
inclui um único zero local!
[ 1.570796311724721E+001, 1.570796311724722E+001]
inclui um único zero local!
[ 1.884955592805117E+001, 1.884955592805118E+001]
inclui um único zero local!

21 inclusão(ões) intervalar(es)
```

Capítulo 5

Conclusões

Este trabalho apresentou os conceitos mais importantes necessários no entendimento da aritmética intervalar. Mais do que uma simples teoria matemática, o conjunto de intervalos reais fornece uma metologia que se apresenta muito eficiente no projeto e implementação de algoritmos numéricos científicos.

Quando se fala em computação numérica, a exatidão dos cálculos é algo de muita importância e por isso necessária para uma solução satisfatória, sendo assim, a matemática intervalar é uma perfeita abordagem para se assegurar tal feito, uma vez que a análise de erros já vem incluída no próprio resultado da computação.

A ferramenta C-XSC demonstrou ser uma boa opção na computação de cálculos matemáticos, uma breve descrição deste ambiente foi dada visando a demonstração de uma das linguagens XSC disponíveis para a computação numérica.

Na seção 4.2 nota-se que algoritmos intervalares não é uma mera transcrição do seu algoritmo pontual correspondente o que implica em uma análise criteriosa no desenvolvimento destes.

Propostas para trabalhos futuros:

- Estudo mais aprofundado da aritmética intervalar na computação científica;
- Estudo mais detalhado do ambiente C-XSC;
- Outras áreas de aplicação da aritmética intervalar.

Referências Bibliográficas

[CM94] D. M. CLÁUDIO and J. M. MARINS. *Cálculo Numérico Computacional*, chapter Matemática Intervalar. Atlas S.A., 1994.

[CRU01] M. M. C. CRUZ. Uma relação de equivalência entre funções intervalares baseada na noção de consistência. Master's thesis, Universidade Federal do Rio Grande do Norte, 2001.

[DD01] A. M. DIAS and G. P. DIMURO. Um ambiente integrado para o ensino e pesquisa em matemática intervalar, 2001. Universidade Católica de Pelotas.

[HE01] T. HICKEY and M. H. EMDEN. Interval arithmetic: From principles to implementation. *Journal of the ACM*, 48(5), September 2001.

[HÖH97] C. L.et al. HÖHER. *Programando em Pascal XSC*. Sagra-Luzzatto, 1997.

[Ins96] Institute for Applied Mathematics. *C-XSC: A C++ Class Library for Extend Scientific Computing*, vol. 1 edition, July 1996. University of Karlsruhe.

[KEA97] R. B. KEARFOTT. Interval computations: Introduction, uses and resources, 1997. University of Southwestern Louisiana.

[MOO62] R. E. MOORE. Interval arithmetic and automatic error analysis in digital computing. Technical Report NR-044-211, Stanford University, 1962.

[MY59] R. E. MOORE and C. I. YANG. Interval analysis i. Technical Report LMSD-285875, Lockheed Aircraft Corporation, 1959.

[OLI97] P. W. et al. OLIVEIRA. *Fundamentos da Matemática Intervalar*. Sagra-Luzzatto, 1997.

[RIB01] D. R. RIBAS. Construção do espaço intervalar, 2001. Pontifícia Universidade Católica do Rio Grande do Sul.

[SUN58] T. SUNAGA. *Theory of an Interval Algebra and its Application to Numerical Analysis*, chapter Interval Calculus, p. 547-564. RAAG Memoirs, 1958.

[VAC01] G. L. R. VACCARO. *Solução de Equações Intervalares*. PhD thesis, Universidade Federal do Rio Grande do Sul, Porto Alegre, Novembro 2001.